Novel Techniques for Robust and Generalizable Machine Learning

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von
Abdelhak Lemkhenter
von Morocco

Leiter der Arbeit:
Prof. Dr. P. Favaro
Institut für Informatik

Novel Techniques for Robust and Generalizable Machine Learning

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von
Abdelhak Lemkhenter
von Morocco

Leiter der Arbeit:
Prof. Dr. P. Favaro
Institut für Informatik

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern, 22.08.2023

Der Dekan:
Prof. Dr. M. Herwegh

Contents

List of Figures

List of Tables

Abstract

Neural networks have transcended their status of powerful proof-of-concept machine learning into the realm of a highly disruptive technology that has revolutionized many quantitative fields such as drug discovery, autonomous vehicles, and machine translation. Today, it is nearly impossible to go a single day without interacting with a neural network-powered application. From search engines to on-device photo-processing, neural networks have become the go-to solution thanks to recent advances in computational hardware and an unprecedented scale of training data. Larger and less curated datasets, typically obtained through web crawling, have greatly propelled the capabilities of neural networks forward. However, this increase in scale amplifies certain challenges associated with training such models. Beyond toy or carefully curated datasets, data in the wild is plagued with biases, imbalances, and various noisy components. Given the larger size of modern neural networks, such models run the risk of learning spurious correlations that fail to generalize beyond their training data.

This thesis addresses the problem of training more robust and generalizable machine learning models across a wide range of learning paradigms for medical time series and computer vision tasks. The former is a typical example of a low signal-to-noise ratio data modality with a high degree of variability between subjects and datasets. There, we tailor the training scheme to focus on robust patterns that generalize to new subjects and ignore the noisier and subject-specific patterns. To achieve this, we first introduce a physiologically inspired unsupervised training task and then extend it by explicitly optimizing for cross-dataset generalization using meta-learning. In the context of image classification, we address the challenge of training semi-supervised models under class imbalance by designing a novel label refinement strategy with higher local sensitivity to minority class samples while preserving the global data distribution.

Lastly, we introduce a new Generative Adversarial Networks training loss. Such generative models could be applied to improve the training of subsequent models in the low data regime by augmenting the dataset using generated samples. Unfortunately, GAN training relies on a delicate balance between its components, making it prone mode collapse. Our contribution consists of defining a more principled GAN loss whose gradients incentivize the generator model to seek out missing modes in its distribution.

All in all, this thesis tackles the challenge of training more robust machine learning models that can generalize beyond their training data. This necessitates the development of methods specifically tailored to handle the diverse biases and spurious correlations inherent in the data. It is important to note that achieving greater generalizability in models goes beyond simply increasing the volume of data; it requires meticulous consideration of training objectives and model architecture. By tackling these challenges, this research contributes to advancing the field of machine learning and underscores the significance of thoughtful design in obtaining more resilient and versatile models.

Acknowledgments

"He who travels alone travels fastest, but in the company of friends you go farther."

I am deeply grateful to my exceptional supervisor, Prof. Dr. Paolo Favaro, for his invaluable support, expertise, and mentorship throughout this thesis. His guidance and feedback have shaped my research and academic skills. Prof. Dr. Favaro's commitment to my growth and his passion for research have inspired me to strive for excellence and innovation.

A special thanks goes to Prof. Dr. Friedemann Zenke and Prof. Dr. David Bommes for serving as thesis examiners. I appreciate their valuable feedback.

I would like to thank all the members of the Computer Vision Group (CVG) in Bern: Dragana Heinzen, Simon Jenni, Givi Meishvili, Xiaochen Wang, Adrian Wälchli, Adam Bielski, Josué Page, Tomoki Watanabe, Riccardo Fantinel, Llukman Çerkezi, Aram Davtyan, Alp Eren Sari, Sepehr Sameni, Hamadi Chihaoui, Luigi Fiorillo, Renato Maria Prisco, Viktor Shipitsin, and Athanasios Charisoudis. They have greatly contributed to fostering a great working environment where we support each other and help each other break through roadblocks and see problems from a completely new perspective. I am also grateful to the research groups member under the IRC Decoding Sleep umberalla, especially to Prof. Dr. Athina Tzovara, Alnes Florence Aellen, and Sigurd Lerkerød from Cognitive Computational Neuroscience (CCN) group in Bern for the many valuable interaction and collabrations.

I would like to also thank Davide Modolo, Manchen Wang, Luca Zancato, Gurumurthy Swaminathan, Yanbei Chen, Abhay Mittal, Joe Tighe, Kaustav Kundu, and Zhenlin Xu for their mentorship during my internship at AWS.

I am also grateful to my friends for their support and encouragement throughout this journey. I appreciate their presence in my life and the moments of respite they have provided during challenging times.

Most importantly, I am most grateful to my family. Pursuing my education on a continent different from them has been an arduous journey, but their unwavering support, their sacrifices, and love have been a constant source of strength, and I am truly fortunate to have such a caring and loving family by my side. Making them proud has been and will always be one of my biggest driving forces and joys in life.

Chapter 1

Introduction

Over the last few years, machine learning models have been rapidly increasing in scale due to larger datasets and more powerful computational resources that have enabled training such models. Works such as Fedus et al. [37] can train a trillion-parameter model on corpora such as C4 [117]. For reference, a large computer vision dataset, such as JFT [141] contains 700 million images. Despite these impressive figures, they pale in comparison to the amount of data posted daily on the Internet. An estimated 400 hours of video are uploaded every minute to YouTube and around 100 million new images are posted on Instagram daily[1]. Therefore, the reliability and generalizability of machine learning models is an important issue to tackle, especially when one cannot hope that the scale of training data will one day catch up to the data generation rate. Indeed a useful machine learning model is one such that it is able to handle various distributions shifts present in the wild and capture robust features and patterns across settings.

1.1 Neural Networks as Universal Approximators

The Universal Approximation Theorem states that one can approximate a given function between two Euclidean spaces up to an arbitrary level of precision using feedforward neural networs [52]. It was later shown that this result holds for a three-layer multilayer perceptron (MLP) [92]. This has propelled neural networks to the forefront of machine learning since they are able to learn complex mappings between various modalities. However, this function approximation capability is a double-edge sword, as the neural network could learn arbitrary spurious correlations. This is called overfitting. Addressing this failure mode requires a careful design of the training scheme and the benchmarks used to evaluate such models.

[1] https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day/

1.2 Cases of Overfitting

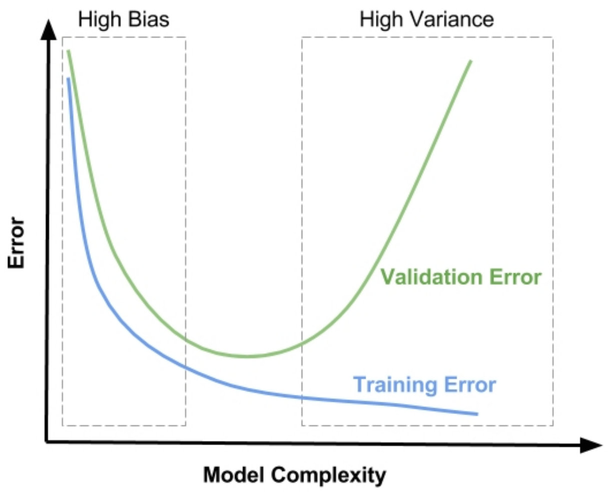


Figure 1.1: **Bias-variance trade off**. Illustration of bias-variance trade off in terms of the model capacity when comparing the training and validation errors[2].

The classic machine learning literature tackles the risk of overfitting as a bias-variance trade-off. This interpretation is rooted in the literature of statistical estimators and is often illustrated using Figure 1.1. Nevertheless, overfitting is not a one-note phenomenon, but rather a combination of different training failure cases such as:

1.2.1 Learning Spurious Correlation

Given a finite set of training data, it is often possible to solve the training task without the model learning a robust set of features. Let us consider an example of a data set that contains images of cats and dogs. Depending on how the dataset was collected, it might be enough for the model to learn to detect whether the photo was taken indoor or outdoor to solve the classification task. In fact, if the training set is not carefully curated, images of dogs tend to be taken in parks, while images of cats are predominantly taken inside the house. Obviously, a model that relies solely on this spurious correlation is not a robust cat/dog detector. Such spurious correlations are always present in the training data to various degrees and can be easily captured by neural networks. Beyond biases present in the training data, the per-sample noise component, such as JPEG compression artifacts or additive noise, could be used as a

[2] https://learnopencv.com/bias-variance-tradeoff-in-machine-learning/

unique sample identifier, allowing the neural network to simply memorize the training data. To address this risk, different image augmentation strategies have become standard practice in computer vision [23, 24].

1.2.2 Learning under Label Noise

Outside of synthetic datasets, data labels are obtained using human annotators. Services such as Amazon Mechanical Turk[3] have allowed for an unprecedented scale of data annotation. However, human annotations are not perfect. Due to the focus on efficiency and the ambiguity associated with more complex tasks, the agreement between different human annotators is almost always below 100%. This setting is referred to as label noise (LN). A common workaround to this issue is to use multiple annotators and use the majority vote as the ground-truth label. However, this is not a perfect solution. The source of disagreement between annotators can be systematic; *e.g.* the image contains multiple objects, but only one class label can be selected, or the dataset was acquired over an extended period of time during which the annotation protocol changed, *e.g.* medical recordings. Therefore, a proper model should learn the more robust label mapping instead of perfectly memorizing the noisy labels present in the training set, *i.e.* it should prioritize generalization over perfectly solving the training task. Indeed, under random label noise, the best performing model on a properly designed test set is the one able to capture the underlying robust label mapping.

1.2.3 Learning Domain Specific Features

As stated previously, the scale of the data used to train a model is significantly smaller than the data available in the wild, which it will be deployed on. This difference in scale is not just a difference in the quantity of data but also a difference in the diversity of the data. The data in the wild may belong to a different domain; *i.e.* there is a distribution shift between the training and test data. Examples of such domain gaps include:

- medical datasets where the patterns differ slight from subject to subject;
- differences in the data acquisition setting, *e.g.* low light setting vs. well lit setting;
- *Sim2Real* where the training data was obtained using a simulation, *e.g.* a 3D rendering engine, and the model is deployed on real data.

Similarly to the previous two cases, the model could learn to solve the training task by relying on features specific to the training domain that do not generalize beyond it, *e.g.* textures specific to the 3D rendering engine.

[3] `https://www.mturk.com/`

1.2.4 Memorization in Deep Neural Networks

Although overfitting is often depicted as a training failure mode, building more generalizable models does not imply eliminating overfitting completely. Arpit et al. [5] has shown that sample memorization is an intrinsic part of neural network training, where the network first focuses on the easier samples than switches to memorizing the more challenging ones. Carlini et al. [19] further shows that even if these challenging samples are removed from the training data, the network will identify another set of the next inline challenging samples to memorize. Therefore, generalizable models should account for this phenomenon during their training.

1.3 Learning to Generalize

This thesis addresses the problem of building more robust and generalizable models. Under this broad umbrella, each chapter tackles the generalization problem in a different context.

1.3.1 Learning to Ignore Noise

As noted in Section 1.2, a given training dataset contains a mixture of spurious and robust features. Differentiating these features is a non-trivial task. In the supervised setting, one can rely on a diverse and large enough collection of samples. The model can learn to extract robust features based on the assumption that annotations and spurious features are not causally related. Even then, such spurious correlations might still be learned by the model, especially in the presence of hidden confounding variables; *i.e.* scaling up the training data does not necessarily solve this issue. For example, [42] showed that CNN models trained on ImageNet tend to focus on texture information. Identifying robust features is even more challenging in the unsupervised setting, where no label information is available to serve as a source of counterexamples for spurious correlations. Instead, an alternative approach is to incorporate prior knowledge of some properties of the robust features into the unsupervised training method. For instance, works such as Caron et al. [20] and Chen et al. [21] make the assumption that the robust image feature should be invariant w.r.t. image augmentations such as color jittering, blur, *etc.* In Chapter 3, we introduce an unsupervised training method for medical time series. The type of medical time series we work with is often characterized by a low signal-to-noise ratio and a high degree of interpatient and intercohort variability, both being two major challenges for training robust models as stated before. The main intuition behind our proposed method is that robust patterns are more structured than spurious ones. By incentivizing our model to focus on such patterns and ignore the noisier part of the signal, we greatly improve its generalization to new subjects across different data regimes.

1.3.2 Learning to Generalize

When no additional source of information is available, the previous approach that relies on prior knowledge can be the most that can be done to extract features that generalize well across settings. However, one often has access to more information for each sample in the form of ground-truth labels and/or meta-data. This is especially true for sequential data, where each sequence could be considered as a separate domain. On top of leveraging prior knowledge, as described above, one can also explicitly constrain the model to learn features that generalize well to other sequences in the dataset. In other words, our model is tasked with identifying patterns that are considered robust not only for the current sequence, but also for other sequences w.r.t. a defined supervised training task. One way to go about imposing such a constraint is to cast the training scheme as a meta-learning problem where the goal is to learn robust features that generalize well across settings. We explore this paradigm for sleep scoring in Chapter 4 where the goal is to learn to generalize across datasets.

1.3.3 Learning to Propagate Labels

When aiming to learning robust data representation, in the supervised setting spurious correlation can be more easily identified thanks to the access to the label information, while in the unsupervised setting the model has access to a significantly larger collection of training samples as unlabeled data is usually cheaper to acquire and overall more readily available. In between lays the semi-supervised setting where both unlabeled and labeled training sets are available. Semi-supervised learning is a great fit for tackling model generalization, as it can benefit from both unlabeled data to learn useful data representation while leveraging the labeled data to identify potential erroneous correlations. Not only that, but model generalization and robustness is an important consideration to take into account when designing semi-supervised methods. Indeed, semi-supervised training relies mostly on propagating the annotation information from the labeled set to the unlabeled counterpart. Doing so is prone to propagating different biases and arbitrary correlations or introducing new ones, especially since the labeled set is often very restricted in size, *e.g.* 2 samples per class. Thus a robust semi-supervised method should be carefully designed to minimize failure cases such as confirmation bias when incorporating the unlabeled set into the training scheme.

1.3.4 Learning to Generate Additional Data

An alternative approach to learning robust feature representations when training samples are scarce is to learn to generate additional data. For certain tasks, synthetic annotated data is easier to obtain. One such example is the FlyingChairs dataset [31] for optical flow which consists of 3D chair models moving against a static image background. The FlyingChairs has been a standard inclusion in most modern optical flow training schemes [142, 136]. However, such synthetic processes are not always available. Instead, in the more general case, one can train a generative model as a means to sample more observations from the data distributions. One such model is Generative Adversarial Networks (GANs)[44], which have seen great success in augmenting training data with high-quality generated samples and thus improving the overall performance of subsequent trained models, especially in the computer vision [53] and medical imaging literature [15]. However, training GANs remains a challenging task and an open research question, as they often suffer from failure modes such as mode collapse where the data distribution is learned partially.

1.4 Thesis Contributions

This thesis tackles the challenge of training more robust models in multiple settings and under different interpretations of the term *robustness*. In the unsupervised setting, we derive a biologically inspired self-supervised learning task for training robust feature representations for physiological time series. Robustness here is interpreted as the ability to generalize to new subjects. We then extend such features to the more challenging task of cross-dataset generalization using meta-learning. When comparing recordings from different datasets, many factors can differ, such as data acquisition hardware and protocol and the demographic of the subjects. By tackling such variability that has a direct impact on the performance of trained deep learning models, we provide an important step forward towards building more reliable deep learning models for medical time series. In the semi-supervised setting, we tackle the problem of training a semi-supervised model under different degrees of class imbalance. Robustness in this setting refers to the ability of the model to avoid being biased towards the majority class when the class imbalance is present in either the labeled or the unlabeled sets, or both. To fulfill this goal, we design a new label propagation strategy based on Gaussian Processes that is locally sensitive to nearby minority samples while remaining faithful to the global structure of the data distribution. Lastly, we introduce a new generative loss inspired by the contrastive learning literature. On top of providing a more principled loss that takes the form of a statistical divergence regardless of the optimality of the discriminator, our loss has better gradients that allow it to actively seek missing mode in the distribution of the generated data.

1.4.1 Thesis Outline

Chapter 2: Background We provide an overview of key topics that are relevant for the later chapters. First, we introduce the polysomnography data used in Chapters 3 and 4, its applications, properties, and relevant deep learning methods. We then introduce the different learning paradigms we touch upon in this thesis, including: Self-supervised learning, Meta-learning, and Semi-Supervised Learning. Lastly, we introduce Generative Adversarial Networks along with their applications and current limitations.

Chapter 3: Learning the Phase-Amplitude Coupling We introduce *Phase Swap* a new biologically inspired self-supervised task for physiological time series. By training a model to detect whether the phase and amplitude information of a given signal match or not, we are able to extract deep features that generalize well to new subjects for different downstream classification tasks. The robustness of our *Phase Swap* features is especially prominent in the low data regime, where only a few subjects are available for training.

This work is associated with the publication: Lemkhenter, Abdelhak, and Paolo Favaro. *Boosting generalization in biosignal classification by learning the phase-amplitude coupling.* Pattern Recognition: 42nd DAGM German Conference, DAGM GCPR 2020, Tübingen, Germany, September 28–October 1, 2020, Proceedings 42. Springer International Publishing, 2021.

Chapter 4: Self-Supervised Meta-Learning for Sleep Scoring We extend our *Phase Swap* method beyond generalizing to new subjects. Using meta-learning, we are able to explicitly constrain our new training scheme *S2MAML* to generalize better across datasets. Our formulation favors self-supervised features extracted on a given dataset and that also generalize well to another randomly sampled dataset w.r.t. to a given supervised task. We show that S2MAML outperforms other approaches for different data splits and configurations.

This work is associated with the publication: Lemkhenter, Abdelhak, and Paolo Favaro. *Towards Sleep Scoring Generalization Through Self-Supervised Meta-Learning.* 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE, 2022.

Chapter 5: Imbalanced Semi-Supervised Learning Using Gaussian Processes Self-learning based semi-supervised methods are prone to confirmation bias. This risk is more prominent when training data is skewed and contains significant class imbalances. To address this issue, we introduce SemiGPC, a drop-in extension for existing semi-supervised methods derived from the posterior mean of a Gaussian

Process (GP). Our GP-based label refinement allows our method to better deal with class imbalances by showing a better local sensitivity to minority class samples while preserving the global structure of the data distribution. We show that our method SemiGPC produces state-of-the-art results on standard artificially imbalanced datasets such as CIFAR-LT as well as on other more challenging semi-supervised fine-grained visual classification benchmarks.

Chapter 6: Kernel Density Discrimination GAN GAN training requires a delicate balance between the discriminator and the generator networks and is often prone to converging to suboptimal solution due to mode collapse. To address these issues, we introduce a novel GAN loss based on kernel density discrimination (KDD GAN). By defining our loss as statistical divergence between the kernel density estimates of the real and generated distributions in feature space regardless of the optimality of the discriminator, we obtain better training gradients that encourage the generator to seek missing modes in its distribution. This results in quantitatively better generative models.

Chapter 2

Background

In this chapter, we provide an overview of the polysomnography (PSG) and Electroencephalography (EEG) data used in both Chapter 3 and Chapter 4. We present the acquisition and formatting of this data modality, as well as the different applications in which it is used, and the domain shifts associated with it. We also introduce relevant elements of the Self-Supervised Learning, Semi-Supervised Learning, Meta-Learning, and Generative Modeling literatures.

2.1 Polysomnography

2.1.1 PSG Recordings

Polysomnography [59] is a study of sleep that involves multiple physiological measurements over time. These signals include:

Electroencephalograms (EEG) which measure the synchronized postsynaptic excitation of a population of cortical neurons [58]. Each neurons can be seen as a dipole due to extracellular voltage caused by the negative charge near the neural dendrites, *cf.* Figure 2.1.

These measurements of brain activity are obtained using different electrodes placed on the surface of the head following the 10-20 system as shown in Figure 2.2. Electrode placements require an exact measurement of the dimensions of the skull and are generally laid out to cover five main regions, each associated with a specific prefix: pole (FP), frontal (F), central (C), parietal (P), and occipital (O). Electrodes on the right/left side of the head are designated with an even/left number, respectively. Not all sleep studies include all electrodes, but all sample at least the frontal, central, and occipital regions, as they exhibit K complexes, sleep spindles, and delta activity (0.5Hz-4Hz)[59] respectively, which are specific patterns of brain activity associated

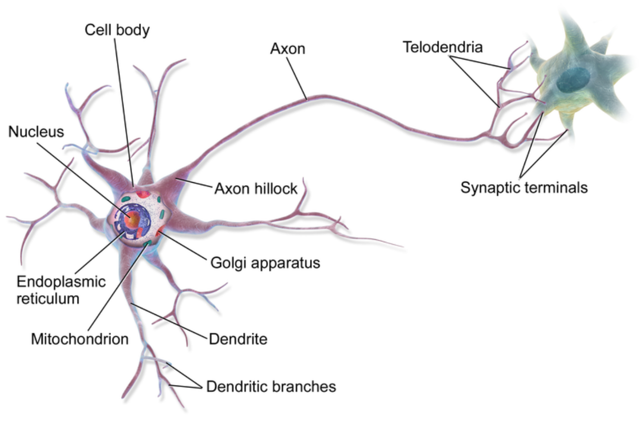


Figure 2.1: **Structure of a neuron**. Diagram representing the structure a neuron and its components[1].

with different stages of sleep, *cf.* Figure 2.3.

EEG measurements are computed relative to the average electrode or computed as the difference between two specific electrodes, *i.e.* a derivation. Recommended derivations are F4-M1, C4-M1, O2-M1 or alternatively, FZ-CZ, CZ-OZ and C4-M1 with FPZ, C3, O1, and M2 as backup electrodes according to the American Academy of Sleep Medicine (AASM) manual [13].

Electro-oculograms (EOG) which measure eye movements. This relies on the dipole formed by the cornea and retina which results in eye movements being recorded as deflections in the electric signal. The presence of eye movements is one of the factors used to distinguish different stages of sleep.

Electromyograms (EMG) which measure muscle activity. EMG electrodes are typically placed on the submental triangle and on the leg during PSG. The former allows for characterizing muscle tone for sleep staging, while the latter can be used to detect disorders such as periodic limb movement (PLM).

Airflow which is measured using a nasal pressure monitor or an oronasal thermal sensor. PSG can also include a measurement of respiratory effort by tracking the movement of the rib cage using inductance plethysmography. These signals can be used to detect disorders such as sleep apnea.

Electrocardiograms (ECG) which measure the electical activity associated with cardiac cycles. This signal can be used to derive the heart rate and detect different

[1]`https://en.wikipedia.org/wiki/Neuron`

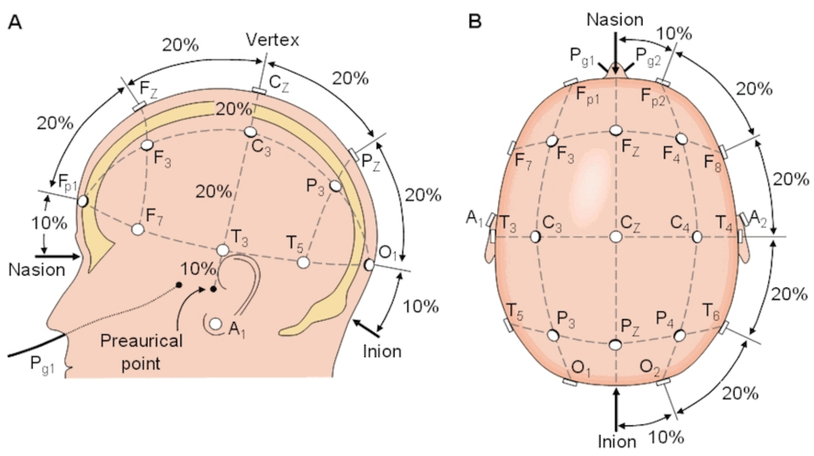


Figure 2.2: **The 10-20 EEG system**. Illustration of the 10-20 EEG system with a 10% and 20% distance between electrodes [128].

arrhythmias.

Oxygenation which measure the oxygen level in the blood using pulse oximetry.

2.1.2 Sleep Scoring

According to the AASM Manual for the Scoring of Sleep and Associated Events [13], sleep scoring focuses on assigning epochs defined as 30 second segments into one of the following stages:

Wakefulness (Stage W) which is characterized by the presence of alpha activity (8-12Hz) in the posterior regions on the EEG for more than 50% of the epoch. However, when the alpha rhythm is not discernible, which is the case in 10% to 20% of healthy subjects, wakefulness can be detected using the EOG through: Blinking (0.5-2Hz), Reading eye movement consisting of a slow phase followed by a rapid movement in the opposite direction, and irregular eye movement with high muscle tone.

Stage NREM1 Sleep which is characterized by slow eye movement, lower muscle tone compared to stage W, and low-amplitude activity focused in the 4-8Hz range. Detecting this stage can be challenging in subjects who do not exhibit an alpha rhythm.

Stage NREM2 Sleep which is characterized by the presence of K complexes, *i.e.* a specific biphasic pattern lasting more than 0.5 seconds, without arousals, *i.e.* the subject wakes up or the presence of a train of waves longer than 0.5 seconds within the 11-16Hz or 12-14Hz frequency bands called sleep spindles. This stage ends when

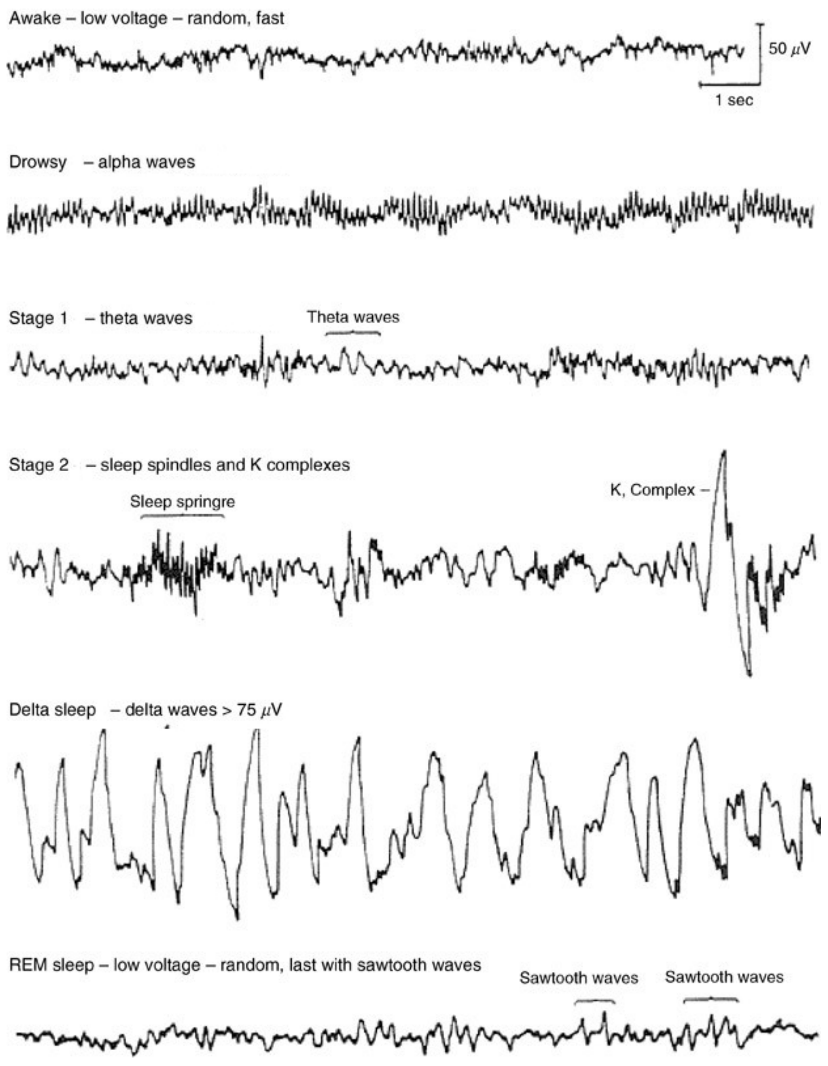


Figure 2.3: **EEG sleep patterns**. Illustration of the EEG patterns associated with the different sleep stages. alpha, theta, and delta waves correspond to the 8-12 Hz, 4-8 Hz, and 0.5-4 Hz frequency bands.

Stage W, REM, or N3 are detected. Sleep apnea can induce K complexes due to frequent arousals that should not be scored as N2.

Stage NREM3 Sleep which is characterized by a predominant slow wave activity for more than 20% of the epoch. Slow waves have a frequency between 0.5Hz and 2 Hz and a peak-to-peak amplitude greater than 75mV. Detecting these waves is of great relevance in the context of cognitive neuroscience [155].

Rapid Eye Movement Sleep (REM) which is characterized by an EEG activity similar to N1, a low muscle tone and irregular and fast eye movements with deflections shorter than 0.5 seconds. Once REM sleep is detected, all subsequent epochs should be scored as such until a specific of transition criteria is met.

Detecting the different stages is useful both in clinical and research settings. Through analyzing the sleep patterns of a given subject, healthcare professionals can detect different abnormalities and pathologies of sleep [58]. In research, studying the brain dynamics during specific stages of sleep sheds light on different cognitive processes such as memory consolidation [70] or creative thinking [33]. Currently, the golden standard for sleep staging requires physicians to manually annotate each 30 seconds epoch in a recording of PSG 8 hours. In order to make this process more efficient, different automatic sleep scoring methods have been proposed [8, 113, 114]. However, such methods are still not widely adopted due to the challenging nature of the PSG data and the different sources of variability that a reliable automatic scoring method should account for.

2.1.3 Domain Shifts

In order to automate the detection of the different sleep stages, one needs to design an algorithm that accounts for the variability present in the PSG data due to various factors including:

Electrode Choice: This is most prominent for the EEG data, but also for EMG, EOG where different PSG recordings contain different EEG electrodes within the PSG database. Depending on the condition of the subject in a clinical setting or the experimental design in a research setting, a set of electrodes may be adopted, and it is often impractical and/or redundant to record all possible electrodes in the 10-20 system. Therefore, an automated algorithm should not overspecialize for a specific choice of input electrodes and instead be flexible in that regard.

Physiological Differences: PSGs can be recorded from subjects with different conditions and states. Many factors can affect brain activity. For example, as mentioned above, 10-20% of the population do not exhibit an alpha rhythm. Electrical activity is also known to decrease with age [3]. Furthermore, different diseases can have a direct impact on the EEG signal such as the presence of legions or epilepsy [135].

Imperfect annotation: When following the guidelines specified in the AASM [13], different experts are reported to achieve an average agreement of 82%. Certain epochs are ambiguous making the detection of the exact onset of different sleep stages a non-trivial task. Furthermore, the PSG database considered may contain recordings that use a different set of sleep scoring criteria if they were processed before the AASM was issued.

To account for these factors, a reliable automatic sleep scoring method should not overfit to the different noise sources in its training database and instead capture features that generalize the most.

2.1.4 Deep Learning for Bio-signals.

Similarly to many other fields, biosignals analysis has also seen the increase in popularity of deep learning methods for both classification [54] and representation learning [9].The literature review [123] showcases the application of deep learning methods to various EEG classification problems such as brain computer interfaces, emotion recognition and seizure detection.

2.1.5 Automatic Sleep Scoring

Recent methods, such as SeqSleepNet [114], have focused on exploiting the context of the data by staging sequences rather than single epochs, or aimed at reducing the model parameters and introducing an estimate of the prediction uncertainty [39]. An important limitation that has emerged is the lack of generalization, *i.e.* , the drop in performance when trained models are applied to new data. As mentioned in the Introduction, this phenomenon is currently attributed to the large diversity of the data across subjects/patients and sessions. To address this problem, U-Sleep [113] introduces a u-net architecture for high-frequency sleep staging. However, generalization across datasets remains an open problem.

Beyond supervised methods, the work by Banville *et.al.* [9] leverages self-supervised tasks based on the relative temporal positioning of pairs/triplets of EEG segments to learn a useful representation for a downstream sleep staging application.

2.2 Self-Supervised Learning

In this section, we introduce self-supervised learning and present two main classes of methods in this field: Invariance-based methods and equivariant-based methods.

2.2.1 Overview

Self-supervised learning is a machine learning paradigm in which one learns to extract a useful feature representation of the data by solving a pretext task. Such pretext tasks are defined without using human annotations. The utility of the learned feature representation and the patterns it captures are determined by the choice of the pretext task. Therefore, one can influence the robustness and generalizability of the learned features by carefully designing the pseudo-task.

Self-supervised learning has become the goto practice in recent years for extracting useful feature representation from a growing amount of available unlabeled data and for training machine models that generalize to a large set of downstream tasks based on different data modalities including, but not limited to: images, videos, natural language, medical recordings, graphs, audio, *etc.*

In the following sections, we cover two main classes of tasks used in the self-supervised learning literature. At a high level, both classes of methods can be summarized using Equation (2.1)

$$\mathcal{L}_{ssl}(x) = d(t(x), y(x)) \tag{2.1}$$

that defines the training loss for a sample x, where t, y and d are two transformations and a metric or loss function, respectively. In invariance-based methods, y and t are independent while $y(x)$ is determined by the choice of t for the equivariance-based ones.

2.2.2 Learning to be invariant

The first class of self-supervised learning tasks is designed to encourage the obtained feature representation to be invariant with respect to a known noise factor. One of the most prominent paradigms that falls under this definition is self-supervised contrastive learning. Contrastive learning methods aim to maximize the similarity between pairs of positive samples, *e.g.* , different augmentations of the same image, while minimizing the similarity between pairs of negative samples, *e.g.* , augmentations of different images, with applications ranging from metric learning to self-supervised training [77]. One of the most commonly used losses in this context is the Normalized Temperature-scaled Cross Entropy Loss (NT-Xent) [21], also referred to as Information Noise Contrastive Estimation loss (InfoNCE). [148] shows that this loss can be split into two terms: an alignment term and a uniformity term. The former encourages the alignment of positive pairs of samples, whereas the latter encourages the feature representation to be uniformly distributed on the unit-sphere. The uniformity constraint stems from maximizing the entropy of the distribution of the features, which is computed through kernel density estimation using the von Mises-Fisher kernel on the unity sphere. This interpretation of the uniformity constraint is experimentally validated by replacing the entropy term with other statistical divergences between the features distribution and a uniform one [22].

Other works such as Caron et al. [20] and Oquab et al. [109] forgo the need for negative samples relying solely on augmentation-based positive samples in what is referred to as non-contrastive methods.

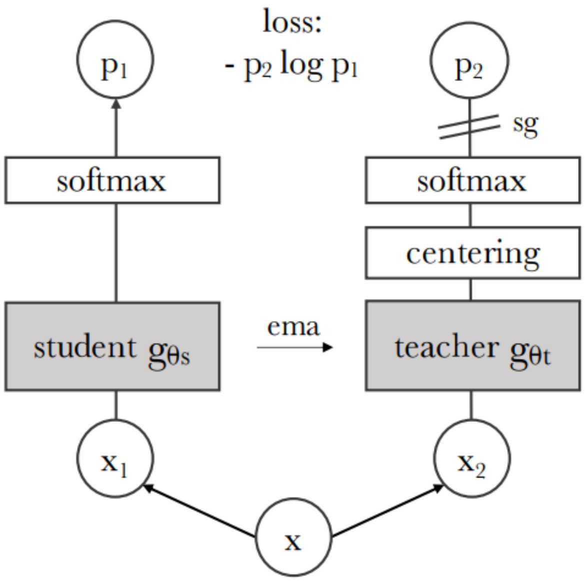


Figure 2.4: **Outline of DINO self-supervised training.** Given two random augmentations x_1 and x_2 of the same image x, the student network is tasked with matching its output p_1 given x_1 to that of the teacher p_2 obtained after a centering operation given x_2. The teacher model is defined as a exponential moving average of the student.

We illustrate this using DINO [20] as an example. The general outline of the method is shown in Figure 2.4. It consists of two models, a student network and a teacher network defined as the exponential moving average (EMA) of the student. Two image augmentations x_1 and x_2 are generated from the same original image x, the student network learns to match the prediction of the teacher network. For DINO, the matching criterion is defined as the cross-entropy between the softmax predictions of the student and teacher networks. The feature representation obtained using DINO generalizes extremely well to computer vision tasks such as segmentation, depth estimation, image classification, *etc.*

Other self-supervised methods define different criteria, such as the Euclidean distance or the InfoNCE[108] loss, where the model is trained in contrastive fashion, *i.e.* to distinguish between augmentations of the same image (positive samples) and augmentations of other images (negative samples). Contrastive methods include works such as Chen et al. [21] and He et al. [47]. The main underlying hypothesis of such methods is that random augmentations do not alter the unknown label. [126] argues that a stronger assumption can be made where they postulate the existence of intra-class connectivity, *i.e.* there exists a path consisting in a sequence of augmentations that connect two samples from the same class.

For temporal data such as time series or videos, one can use temporal jittering as data augmentation. By learning to encode temporal neighbors using similar features, the model is able to learn a state-like representation of the signal. Works such as Lorre et al. [90] follow this approach.

2.2.3 Learning to solve pseudo-tasks

The second major class of self-supervised tasks consists of defining pseudo-tasks for which labels can be generated automatically. The model learns the underlying concepts present in the data by solving the pretext task, and thus the design of the task should be carefully considered. Let us consider the example of RotNet [72]. Given an image x, t in Equation (3.2) is defined as a random rotation with an angle $\theta \in \{0°, 90°, 180°, 240°\}$. The network is trained to predict θ from $t(x)$. For example, if the image x is that of a dog, the model needs to learn what a dog is to determine whether the image is upright or rotated. Another example of such methods is masked auto-encoders (MAE) [48] where t randomly masks patches in x and the model is training to inpaint the missing patches, *cf.* Figure 2.5. Due to the high masking ratio, the model is forced to learn to extract the objects present in the image in order to generate the missing patches.

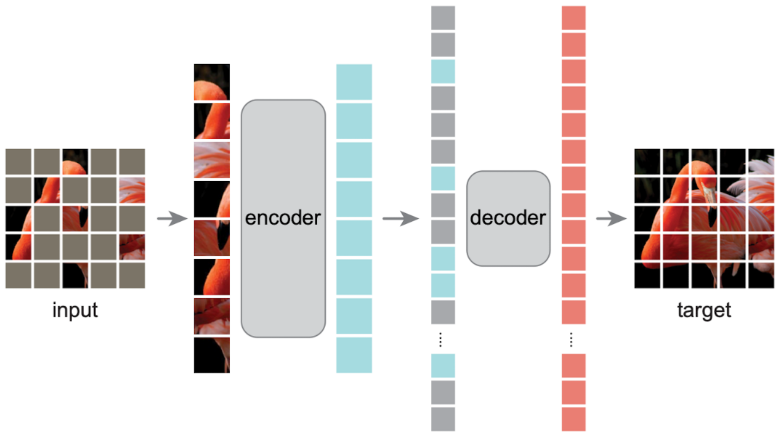


Figure 2.5: **Outline of a Masked Auto Encoder**. The decoder network is tasked with reconstructing the random masked patches in the input to the encoder.

2.2.4 Self-Supervised Learning as a Scalable learning paradigm

One of the main advantages of self-supervised learning is its scalability to larger unlabeled datasets. Indeed, with the gradual shift towards larger and more diverse training sets such as ImageNet-21k [26] and towards larger models, self-supervised methods exhibit a promising scaling trend. A recent example of this is apparent when comparing DINO [20] to DINOv2 [109] where one of the main differences was the dataset scale and quality that allowed DINOv2 to achieve impressive results in a wide range of downstream tasks.

2.3 Meta-Learning

In this section, we will cover the concept of meta-learning. First, we formalize it as a bilevel optimization procedure. We then introduce MAML [38], a pioneering work in the field, as well as other meta-learning methods derived from it.

2.3.1 Problem Formulation

Meta-Learning is defined as learning to learn. For a parametric model $\mathcal{M}(., \Theta)$, where Θ represents its set of parameters, and a learning objective $\mathcal{L}$, the meta-learning problem can be formalized as a bilevel optimization

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{L}(\mathcal{D}_{val}, \hat{\Theta}) \tag{2.2}$$

$$\text{s.t. } \hat{\Theta} = \underset{\Theta'}{\operatorname{argmin}}\, \mathcal{L}(\mathcal{D}_{train}, \Theta') \tag{2.3}$$

$$\text{and } \Theta = \hat{\Theta}.$$

$\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ denote the meta-training and meta-validation sets respectively. Note that the upper optimization is in Θ. In other words, the meta-learning objective aims at finding the optimal set of parameters on the meta-training set while ensuring that they generalize to the meta-validation set. Different meta-learning methods have been proposed to solve Equation (2.2), each adopting a different link between the outer optimization loop in Equation (2.2) and the inner optimization loop in Equation (2.3). Gradient-based methods such as MAML [38] rely on the alignment between the gradients the two optimization problems while Prototypical Networks [132] and memory-based models such as Neural Turing Machine [45] cast it as a metric learning problem where the decision on the meta-validation set is based on the most similar samples in the meta-training set, the set of prototypes and/or the memory bank. In the context of this thesis, we focus on the literature on gradient-based meta-learning.

2.3.2 Gradient-based Meta-Learning

In gradient-based meta-learning, the generalization from meta-training to meta-validation sets is characterized by the alignment between the gradients of Equation (2.2) and Equation (2.3). In other words, the optimal meta-parameters are such that they can be quickly adapted to new task using a limited set of the new data. One prominent gradient-based methods is Model Agnostic Meta-Learning (MAML) [38] in which Equations (2.2) and (2.3) are rewritten as

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{L}(\mathcal{D}_{val}, \hat{\Theta}) \tag{2.4}$$

$$\hat{\Theta} = \text{GradientDescent}(\mathcal{L}, \mathcal{D}_{train}, \Theta, k, \lambda) \tag{2.5}$$

where k and λ are the total number of gradient descent steps and the learning rate respectively. When $k = 1$, Equation (2.5) is given by

$$\hat{\Theta} = \Theta - \lambda \nabla_\Theta \mathcal{L}(\mathcal{D}_{train}, \Theta). \tag{2.6}$$

In other words, a gradient step w.r.t. to the outer optimization problem requires computing the gradient of the GradientDescent operation w.r.t. Θ, *i.e.* computing the second-order derivatives of the model. To address the computational overhead associated with higher-order derivative, different approximations such as FOMAML [38] and REPTILE [104] have been proposed. FOMAML or First-Order MAML consists in a first-order approximation of MAML where the second-order term is neglected, while REPTILE defines the meta-gradient as the difference $\hat{\Theta} - \Theta$. Other works such as COMLN [25] replace the iterative gradient descent in Equation (2.5) with a continuous-time process that allows for a more exact formulation.

Most variants of MAML are based on the same underlying principle that consists of re-weighting the contribution of each sample in the outer optimization level based on the alignment of its gradient with the gradients of the inner optimization, *i.e.* for $x \in \mathcal{D}_{val}$ its weight is a function of $< \nabla_\Theta \mathcal{L}(\{x\}, \Theta), \nabla_\Theta \mathcal{L}(\mathcal{D}_{train}, \Theta) >$ where $< ., . >$ is the dot product [118].

2.3.3 Applications of Meta-Learning

The general purpose nature of meta-learning makes it applicable to a large range of tasks. One such task is few-shot learning where the model is presented with a limited set of examples, through the meta-training set, in order to solve a novel task, represented by the meta-validation set. Meta-learning can also be used for Domain Adaptation [116] and efficient Reinforcement Learning [51].

2.4 Semi-Supervised Learning

In this section, we provide an overview of the semi-supervised literature. This includes the general principle of propagating the label information using the learned data structure as well as introducing the paradigm of self-learning.

2.4.1 Problem Formulation

Given a labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n_l}$ and an unlabeled set $\mathcal{U} = \{x_j\}_{j=1}^{n_u}$, the goal of semi-supervised learning is to learn a representation of $p(x)$ using both $\mathcal{D}$ and $\mathcal{U}$ that enables the propagation of the label information $p(y|x)$ from $\mathcal{D}$ to $\mathcal{U}$. In practice, unlabeled data is significantly less costly to acquire and more readily available, resulting in n_l being significantly smaller than n_u. This makes semi-supervised learning a more cost-effective and scalable learning paradigm. Moreover, partial annotations allow for greater flexibility. Let us consider the following scenario describing the scalability of the annotation cost of an object detection dataset both in the supervised and semi-supervised settings when samples from new classes are added. Given an image dataset $\mathcal{D}_1$ of 1000 different objects/classes each annotated using a bounding box if present in a given image and $\mathcal{D}_2$ a set of new samples pertaining to 100 new classes not present in $\mathcal{D}_1$. We consider the annotation cost of combining $\mathcal{D}_1$ and $\mathcal{D}_2$ into a superset $\mathcal{D}_3$ with 1100 classes. In the supervised setting, combining $\mathcal{D}_1$ and $\mathcal{D}_2$ would require the reannotation of $\mathcal{D}_1$ and $\mathcal{D}_2$ completely. Indeed, given an image from $\mathcal{D}_1$, its corresponding original annotation only takes into account the original 1000 classes and contains no information on the presence or absence of any of the new 100 classes from $\mathcal{D}_2$. Merging $\mathcal{D}_1$ and $\mathcal{D}_2$ without a reannotation effort will introduce many false negatives to the data. On the other hand, creating $\mathcal{D}_3$ in the semi-supervised learning setting is trivial, as both $\mathcal{D}_1$ and $\mathcal{D}_2$ can be considered as partially annotated datasets w.r.t. to all 1,100 classes. Indeed, partial annotations offer more flexibility and can be used to specify new tasks in a scalable fashion. In the rest of this section, we provide an overview of graph-based semi-supervised learning in order to build an intuition of the label propagation principle, then we introduce consistency-based semi-learning in the context of the modern semi-learning literature.

2.4.2 Graph-based Semi-Superived Learning

Based on the key smoothness assumption that similar samples should have similar labels, *e.g.* belong to the same class, graph-based semi-supervised methods aim to learn a meaningful representation of the data distribution $p(x)$ in the form of a graph structure $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ and $\mathcal{E}$ are the set of nodes and edges, respectively, and each node represents a data point. The goal is to take advantage of the geometric structure of the graph in order to propagate the label information from the labeled set $\mathcal{D}$ to the unlabeled set $\mathcal{U}$.

Earlier works such as Belkin et al. [11] focused on regularizing the label propagation process through a fixed graph structure, while more recent works such as Dornaika et al. [30] jointly learn the graph structure and solve the semi-supervised learning problem.

2.4.3 Self-training

An orthogonal approach for semi-supervised learning to the one presented above is self-learning, where a model is trained using the labeled set $\mathcal{D}$ and as it gets better, a subset of its own predictions is used a pseudo-labels on the unlabeled set. The key assumption is that a fraction of the model predictions on the unlabeled set $\mathcal{U}$ are correct, so by deriving a reliable heuristic to identifying them, one expands the labeled set and trains the model further.

Naturally, using the model predictions as pseudo-labels introduces confirmation bias as an inherent risk of such training procedures. To mitigate this risk, various methods have been proposed that introduce additional regularization, *e.g.* consistency across augmentations, or refine the predicted pseudo-labels. Semi-supervised learning methods such as FixMatch [133], ReMixMatch [14], SimMatch [167] and FreeMatch [151] share the common design choice of enforcing the consistency of the model predictions across augmentations of different strength levels on the unlabeled samples. FixMatch [133] enforces this consistency as a cross-entropy loss applied using the one-hot encoding of the model predictions on weakly augmented unlabeled samples as pseudo-labels for their strongly augmented counterpart. Weak augmentations consist of random image flipping and translation while strong augmentation combine AutoAugment [23] and Cutout [28]. The consistency loss is only applied on high confidence predictions where the predicted probability value is higher than a fixed threshold. ReMixMatch [14] instead opts for using temperature sharpened model predictions as pseudo-labels for its consistency regularization in addition to a rotation prediction regularization loss. FreeMatch [151] introduces a per-class confidence threshold update rule based on model predictions combined with an entropy-based diversity loss, making it more suitable for imbalanced settings. Methods such as CoMatch [84] and SimMatch [167] choose to enforce consistency across additional data representations. In particular, CoMatch [84] encourages consistency between pseudo-labels and embeddings obtained using similarity graphs, while SimMatch [167] encourages consistency between semantic-level and instance-level pseudo-labels. Both methods choose to smooth the predicted pseudo-labels based on a similarity-based aggregate of a memory buffer of samples in order to mitigate confirmation bias. However, these pseudo-label refinement strategies fail to eliminate data biases w.r.t. the class balance.

2.5 Generative Adversarial Networks

Generative modeling refers to a family of machine learning methods in which the goal is to learn to sample from the data distribution. Broadly speaking, given a data distribution $p_{\mathcal{D}}(x, y)$, where x is the random variable that represents each sample and y its corresponding label, generative modeling can be cast as learning the joint distribution $p_{\mathcal{D}}(x, y)$ while discriminative modeling aims at learning $p_{\mathcal{D}}(y|x)$. In the context of this thesis, we provide an overview of Generative Adversarial Networks as our generative method of choice and briefly discuss their potential applications to the incomplete data setting.

2.5.1 Overview

Generative Adversarial Networks (GANs) [44] are a prominent generative learning method in which a generator network is used to map samples from a known distribution, typically a multivariate Gaussian, to samples from the data distribution. Training is based on an adversarial min-max game, where a discriminator is trained to distinguish real samples from fake ones while the generator is trained to fool the discriminator. The link to the data distribution is implicitly defined through the saddle points of the min-max optimization, where the loss of the generator corresponds to a known statistical divergence, *e.g.* the Jensen-Shannon Divergence (JSD) [44].

2.5.2 Applications

GANs have been widely adopted across a wide range of fields. In addition to their ability to generate high-quality samples [63], GANs can be especially useful in various incomplete data settings. Generated samples can be used as data driven augmentations [53], to upsample minority classes [125] or to generate paired samples across modalities. *e.g.* Sim2Real [120], 2D to 3D [85]. Furthermore, discriminative losses adopted by GANs can be used beyond generative modeling. For example, adversarial domain adaptation methods [165] leverage adversarial losses to ensure invariance across domains, resulting in more generalizable features.

2.5.3 Limitations

Despite their many advantages, GANs in general and adversarial losses in particular can be challenging to train. Due to the two-player game, GAN training is often unstable and sensitive w.r.t. to various training hyperparameters. Moreover, they are susceptible to mode collapse where the model converges to a partial version of the data distribution and is unable to cover all its modes. Lastly, the evaluation of such models is still an ongoing research area. Different metrics such as FID [50] for images or the BLUE Score [111] for natural language can only provide a partial view of the performance of such generative models given that the underlying data distribution is often restricted to a low-dimensional manifold embedded in a high-dimensional space [115].

Ever since their inception, GANs have gone through various iterations and improvements. Works such as Arjovsky et al. [4] and Nowozin et al. [107] focus on training the generator to minimize other statistical divergences that exhibit better properties compared to the JSD in the original work. One complementary line of research explores additional regularization terms such as using a gradient penalty for the discriminator [96], consistency regularization [162], or differentiable augmentations [164] to various degrees of success. A recent addition to this list are methods that capture more structure into the latent representation of the discriminator through the use of Contrastive Learning [62, 60, 159]. One such example is ContraGAN [62], where the authors introduce a new regularization term, called 2C loss, based on the NT-Xent loss [21] used commonly in Contrastive Learning. The introduced loss term aims at capturing the data-to-data and data-to-class relations in the dataset.

Instance Selection for GANs. Training GANs on large scale datasets remains a challenging task. State-of-the-art models such as BigGAN [17] require a substantial amount of compute resources and many of them require post hoc processing to reduce spurious samples. [29] proposes to address both issues by filtering the dataset using instance selection. They argue that the model's capacity is wasted on low density regions of the empirical distributions of the data. Their results show that instance selection allows to train better GAN models using substantially fewer parameters and training time.

Chapter 3

Boosting Generalization in Bio-signal Classification by Learning the Phase-Amplitude Coupling

This chapter was published as Lemkhenter, A., Favaro, P. (2021). Boosting Generalization in Bio-signal Classification by Learning the Phase-Amplitude Coupling. In: Akata, Z., Geiger, A., Sattler, T. (eds) Pattern Recognition. DAGM GCPR 2020. Lecture Notes in Computer Science, vol 12544. Springer, Cham. https://doi.org/10.1007/978-3-030-71278-5_6. ©2021 Springer Nature Switzerland AG.

Various hand-crafted feature representations of biosignals rely primarily on the amplitude or power of the signal in specific frequency bands. The phase component is often discarded, as it is more sample specific and thus more sensitive to noise than the amplitude. However, in general, the phase component also carries information relevant to the underlying biological processes. In this chapter we show the benefits of learning the coupling of both phase and amplitude components of a biosignal. We do so by introducing a novel self-supervised learning task, which we call *Phase Swap*, that detects if biosignals have been obtained by merging the amplitude and phase from different sources. In our evaluation, we show that neural networks trained on this task generalize better across subjects and recording sessions than their fully supervised counterparts.

Biosignals, such as electroencephalograms and electrocardiograms, are multivariate time series generated by biological processes that can be used to assess seizures, sleep disorders, head injuries, memory problems, heart diseases, just to name a few [102].

Although clinicians can successfully learn to correctly interpret such biosignals, their protocols cannot be directly converted into a set of algorithmic rules that yield comparable performance. Currently, the most effective way to transfer this expertise into an automated system is to gather a large number of examples of biosignals with the corresponding labels provided by a clinician and to use them to train a deep neural network. However, collecting such labeling is expensive and time consuming. In contrast, biosignals without labeling are more readily available in large numbers.

Recently, self-supervised learning (SelfSL) techniques have been proposed to limit the amount of required labeled data. These techniques define a so-called *pretext* task that can be used to train a neural network in a supervised manner on data without manual labeling. The pretext task is an artificial problem where a model is trained to output the transformation that was applied to the data. For example, a model could be trained to output the probability that a time series had been time-reversed [154]. This step is often called pre-training and it can be carried out on large datasets as no manual labeling is required. The pre-trained neural network is then adapted with a smaller learning rate on the small target dataset, where labels are available. This second step is called *fine-tuning*, and it produces a substantial boost in performance [105]. Thus, SelfSL can be used to automatically learn physiologically relevant features from unlabeled biosignals and improve classification performance.

SelfSL is most effective if the pretext task focuses on features that are relevant to the target downstream task. Typical features work with the amplitude or power of biosignals, but, as shown in the literature, the phase carries information about the underlining biological processes [18, 103, 89]. Thus, in this chapter, we propose a new pretext task to learn the coupling between the amplitude and the phase of biosignals, which we call *Phase Swap* (PS). The objective is to predict whether the phase of the Fourier transform of a multivariate physiological time-series segment was swapped with the phase of another segment.

We show that features learned through this task help classification tasks generalize better, regardless of the neural network architecture.

Our contributions are summarized as follows

- With phase swap, we demonstrate experimentally the importance of incorporating the phase in biosignal classification;
- We show that the learned representation generalizes better than current state of the art methods to new subjects and to new recording sessions;
- We evaluate the method on four different datasets and analyze the effect of various hyper-parameters and of the amount of available labeled data on the learned representations.

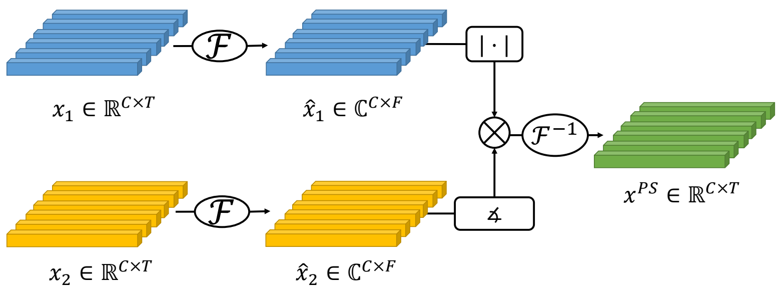

$\mathcal{F}$
$x_1 \in \mathbb{R}^{C \times T}$
$\hat{x}_1 \in \mathbb{C}^{C \times F}$
$|\cdot|$
$\mathcal{F}^{-1}$
$x^{PS} \in \mathbb{R}^{C \times T}$
$\mathcal{F}$
$\measuredangle$
$x_2 \in \mathbb{R}^{C \times T}$
$\hat{x}_2 \in \mathbb{C}^{C \times F}$

Figure 3.1: **Illustration of the phase-swap operator Φ.** The operator takes two signals as input and then combines the amplitude of the first signal with the phase of the second signal in the output.

3.1 Related Work

We provide an overview of the relevant deep learning applications to PSG in general and EEG recordings in particular, the signals they include, and the related self-supervised literature in Sections 2.1, 2.1.4 and 2.2. Different prior works have analyzed the phase component of biosignals. Busch *et.al.* [18] show a link between the phase of the EEG oscillations, in the alpha (8-12Hz) and theta (4-8Hz) frequency bands, and the subjects' ability to perceive flashes of light. The phase of the EEG signal has also been shown to be more discriminative for determining the firing patterns of neurons in response to certain types of stimuli [103]. López et al. [89], highlights the potential link between the phase of the different EEG frequency bands and cognition during proactive control of task switching.

3.2 Learning to Detect the Phase-Amplitude Coupling

In this section, we define the *Phase Swap* operator and the corresponding SelfSL task, and present the losses used for pre-training and fine-tuning.

Let $D_{i,j}^{W} = \{(x^{i,j,k}, y^{i,j,k})\}_{k=1}^{N}$ be the set of samples associated with the i-th subject during the j-th recording session. Each sample $x^{i,j,k} \in \mathbb{R}^{C \times W}$ is a multivariate physiological time series window where C and W are the number of channels and the window size, respectively. $y^{i,j,k}$ is the class of the k-th sample. Let $\mathcal{F}$ and $\mathcal{F}^{-1}$ be the Discrete Fourier Transform operator and its inverse, respectively. These operators will be applied to a given tensor x. In the case of multivariate signals, we apply these operators channel-wise.

For the sake of clarity, we provide the definitions of the absolute value and the phase

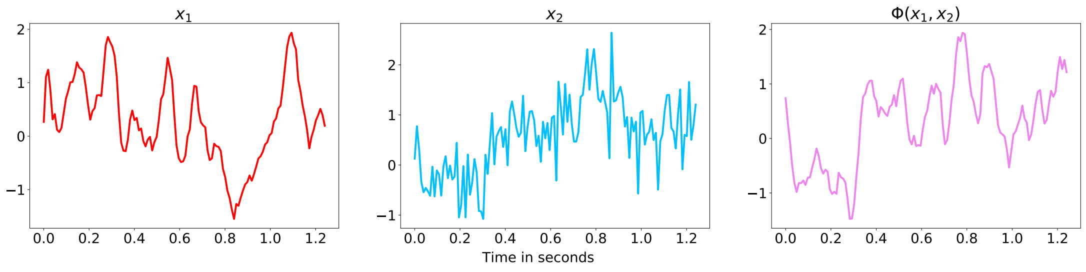


Figure 3.2: **Visualization of a phase-swaped sample.** Illustration of the PS operator on a pair of 1.25 seconds segments taken from the Fpz-Cz channel in the SC dataset [100]. The amplitude and phase information are taken from the signals and x_1 and x_2, respectively.

element-wise operators. Let $z \in \mathbf{C}$, where $\mathbf{C}$ denotes the set of complex numbers. Then, the absolute value, or *magnitude*, of z is denoted $|z|$ and the phase of z is denoted $\measuredangle z$. With such definitions, we have the trivial identity $z = |z| \measuredangle z$.

Given two samples $x^{i,j,k}$, $x^{i,j,k'} \in D_{i,j}^W$, the *Phase Swap* (PS) operator Φ is

$$\Phi\left(x^{i,j,k}, x^{i,j,k'}\right) \doteq \mathcal{F}^{-1}\left[\left|\mathcal{F}\left(x^{i,j,k}\right)\right| \odot \measuredangle \mathcal{F}\left(x^{i,j,k'}\right)\right] = x_{swap}^{i,j,k}, \tag{3.1}$$

where $\odot$ is the element-wise multiplication (see Figure 3.1). Note that the energy per frequency is the same for both $x_{swap}^{i,k}$ and $x^{i,k}$ and that only the phase, *i.e.* , the synchronization between the different frequencies changes. An example of phase swapping is shown in Figure 3.2. Notice how the shapes of the oscillations change drastically when the PS operator is applied, and no trivial shared patterns seem to emerge.

The PS pretext task is defined as a binary classification problem. A sample belongs to the positive class if it is transformed using the PS operator, otherwise it belongs to the negative class. In all of our experiments, both inputs to the PS operator are sampled from the same patient during the same recording session. Because the phase is decoupled from the amplitude of white noise, our model has no incentive to detect noise patterns. In contrast, it will be encouraged to focus on the structural patterns in the signal in order to detect whether the phase and magnitude of the segment are coupled or not.

We use the FCN architecture proposed by Wang *et.al.* [152] as our core neural network model $E : \mathbb{R}^{C \times W} \to \mathbb{R}^{H \times W/128}$. It consists of 3 convolutional blocks using a Batch Normalization layer [55] and a ReLU activation followed by a pooling layer. The output of E is then flattened and fed to two Softmax layers C_{Self} and C_{Sup}, which are trained on the self-supervised task and the supervised task, respectively. Instead of a global pooling layer, we used an average pooling layer with a stride of 128. This allows us to keep the number of weights of the supervised network $C_{Sup} \circ E$ constant when

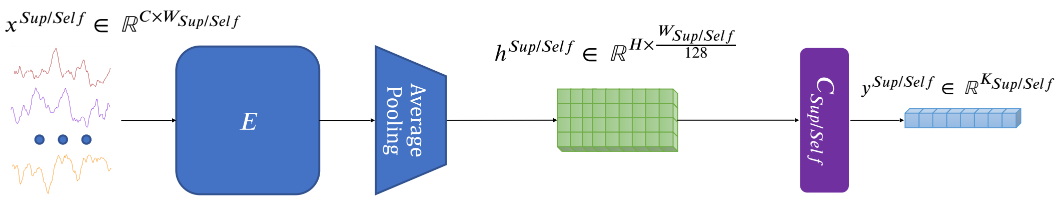

$x^{Sup/Self} \in \mathbb{R}^{C \times W_{Sup/Self}}$
E
Average Pooling
$h^{Sup/Self} \in \mathbb{R}^{H \times \frac{W_{Sup/Self}}{128}}$
$C_{Sup/Self}$
$y^{Sup/Self} \in \mathbb{R}^{K_{Sup/Self}}$

Figure 3.3: **Training Outline**. The encoder network is first E is trained with C_{Self} using the self-supervised loss. C_{Self} is then replaced with C_{Sup} for the supervised training.

the self-supervised task is defined on a different window size. The overall framework is illustrated in Figure 3.3. Note that the encoder network E is the same for both tasks.

The loss function for training on the SelfSL task is the cross-entropy

$$\mathcal{L}_{Self}\left(y^{Self}, E, C_{Self}\right) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K_{Self}} y_{i,k}^{Self} \log\left(C_{Self} \circ E(x_i)\right)_k, \tag{3.2}$$

where $y_{i,k}^{Self}$ and $(C_{Self} \circ E(x_i))_k$ are the one-hot representations of the true SelfSL pretext label and the predicted probability vector, respectively. We optimize Equation (3.2) with respect to the parameters of both E and C_{Self}. Similarly, we define the loss function for the (supervised) fine-tuning as the cross-entropy

$$\mathcal{L}_{Sup}\left(y^{Sup}, E, C_{Sup}\right) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K_{Sup}} y_{i,k}^{Sup} \log\left(C_{Sup} \circ E(x_i)\right)_k, \tag{3.3}$$

where $y_{i,k}^{Sup}$ denotes the label for the target task. The vectors $y_{i,k}^{Sup}$ and $y_{i,k}^{Self}$ are in $\mathbb{R}^{N \times K_{Sup}}$ and $\mathbb{R}^{N \times K_{Self}}$, where N, K_{Sup} and K_{Self} are the number of samples and classes, respectively. In fine tuning, E is initialized with the parameters obtained from the optimization of Equation (3.2) and C_{Sup} with random weights, and then both networks are updated to minimized Equation (3.3), but with a smaller learning rate.

3.3 Experiments

3.3.1 Data Sets

In our experiments, we use the expanded SleepEDF datasets [100, 64, 43], CHB-MIT [127] and ISRUC-Sleep [66] datasets as they contain recordings from multiple patients. This allows us to study the generalization capabilities of the learned feature representation for new recording sessions and new patients. The Expanded SleepEDF database contains two different sleep scoring datasets.

Sleep Cassette Study (SC) [100]. Collected between 1987 and 1991 to study the effect of age on sleep. It includes 78 patients with two recording sessions each (3 recording sessions were lost due to hardware failure).

Sleep Telemetry Study (ST) [64]. Collected in 1994 as part of a study of the effect of Temazepam on sleep in 22 different patients with 2 recordings sessions each.

Both datasets define sleep scoring as a 5-way classification problem. The 5 classes in question are the sleep stages: Wake, NREM 1, NREM 2, NREM 3/4, REM. NREM 3 and 4 are merged into one class due to their small number of samples (these two classes are often combined in sleep studies).

The third dataset that we use in our experiments is the CHB-MIT dataset [127] recorded at the Children's Hospital Boston from pediatric patients with intractable seizures. It includes multiples recording files for 22 different patients. We retain the 18 EEG channels that are common to all recording files. The sampling rate for all channels is 256 Hz. The target task defined in this dataset is to predict whether a given segment is a seizure event or not, *i.e.* , a binary classification problem.

The last dataset that we use is ISRUC-Sleep [66], for sleep scoring as a 4-way classification problem. We use the 14 channels extracted in the Matlab version of the dataset. This dataset consists of three subgroups: subgroups I and II contain, respectively, recordings from 100 and 8 subjects with sleep disorders, whereas subgroup III contains recordings from 10 healthy subjects. This allows us to test the generalization from diagnosed subjects to healthy subjects.

For all datasets, the international 10-20 system [93] was adopted for the choice of the position of the EEG electrodes. For the SC, ST, and ISRUC-sleep datasets we resample the signals to 102.4Hz. This resampling allows us to simplify the neural network architectures we use, because in this case most window sizes can be represented by a power of 2, *e.g.* , a window of 2.5sec corresponds to 256 samples. We normalize each channel per recording file in all datasets to have a zero mean and a standard deviation of one.

3.3.2 Training Procedures and Models

In the supervised baseline (respectively, self-supervised pre-training), we train the randomly initialized model $C_{Sup} \circ E$ (respectively, $C_{Self} \circ E$) on the labeled dataset for 10 (respectively, 5) epochs using the Adam optimizer [68] with a learning rate of 10^{-3} and $\beta = (0.9, 0.999)$. We balance the classes present in the dataset using resampling (no need to balance classes in the self-supervised learning task). In fine-tuning, we initialize E with the weights obtained from the SelfSL training and then train $C_{Sup} \circ E$ on the labeled dataset for 10 epochs using Adam [68], but with a learning rate of 10^{-4} and $\beta = (0.9, 0.999)$. As in the fully supervised training, we also balance the classes using resampling. In all training cases, we use a default batch size of 128.

We evaluate our self-supervised framework using the following models

- *Phase Swap*: The model is pre-trained on the self-supervised task and fine-tuned on the labeled data;

- **Supervised**: The model is trained solely in a supervised fashion;
- **Random**: C_{Sup} is trained on top of a frozen randomly initialized E;
- **PSFrozen**: We train C_{Sup} on top of the frozen weights of the model E pre-trained on the self-supervised task.

3.3.3 Evaluation Procedures

We evaluated our models using different train/validation/test splits in our experiments. In total we use at most 4 sets, which we refer to as the training set, the Validation Set, the Test set A and the test set B. The validation set, test set A, and training set share the same patient identities, while B contains recordings from new subjects. The validation set and test set A use distinct recording sessions. The validation set and the training set share the same patient identities and recording sessions with a 75% (for the training set) and 25% (Validation Set) split. We use each test set for the following purposes:

- **Validation Set**: this set serves as a validation set;
- **Test set A**: this set allows us to evaluate the generalization error on new recording sessions for patients observed during training;
- **Test set B**: this set allows us to evaluate the generalization error on new recording sessions for patients not observed during training.

We use the same set of recordings and patients for both self-supervised and supervised tasks training. For the ST, SC and ISRUC datasets we use class re-balancing only during the supervised fine-tuning. However, for the CHB-MIT dataset, the class imbalance is much more extreme: The dataset consists of less than 0.4% positive samples. Because of that, we undersample the majority class both during the self-supervised and supervised training. This prevents the self-supervised features from completely ignoring the positive class. Unless specified otherwise, we use $W_{Self} = 5sec$ and $W_{Sup} = 30sec$ for the ISRUC, ST and SC datasets, $W_{Self} = 2sec$ and $W_{Sup} = 10sec$ for the CHB-MIT dataset, where W_{Self} and W_{Sup} are the window size for the self-supervised and supervised training, respectively. For the ISRUC, ST and SC datasets, the choice of W_{Sup} corresponds to the granularity of the provided labels. For the CHB-MIT dataset, although the labels are provided at a rate of 1Hz, the literature in neuroscience generally defines a minimal duration of around 10 seconds for an epileptic event in humans [40], which motivates our choice of $W_{Sup} = 10sec$.

Evaluation Metric. As an evaluation metric, we use the balanced accuracy

$$Acc^{Balanced}(y, \hat{y}) = \frac{1}{K} \sum_{k=1}^{K} \frac{\sum_{i=1}^{N} \hat{y}_{i,k} y_{i,k}}{\sum_{i=1}^{N} y_{i,k}}, \tag{3.4}$$

which is defined as the average of the recall values per class, where K, N, y and $\hat{y}$ are respectively the number of classes, the number of samples, the one-hot representation of true labels and the predicted labels.

3.3.4 Generalization on the Sleep Cassette Data Set

We explore the generalization of the self-supervised trained model by varying the number of different patients used in the training set for the SC dataset. r_{train} is the percentage of patient identities used for training, in the Validation Set and in the Test Set A. In Table 3.1, we report the balanced accuracy on all test sets for various values of r_{train}. Self-supervised training was carried out using a window size of $W_{Self} = 5sec$. We observe that the *Phase Swap* model performs the best for all values of r_{train}. We also observe that the performance gap between the *Phase Swap* and *Supervised* models is narrower for larger values for r_{train}. This is to be expected since including more identities in the training set allows the *Supervised* model to generalize better. For $r_{train} = 100\%^*$, we use all recording sessions across all identities for the training set and in the Validation Set (since all identities and sessions are used, the test sets A and B are empty). The results obtained for this setting show that there is still a slight benefit with the *Phase Swap* pre-training even when labels are available for most of the data.

3.3.5 Generalization on the ISRUC-Sleep Data Set

Using the ISRUC-Sleep dataset [66], we aim to evaluate the performance of the Phase Swap model on healthy subjects when it was trained on subjects with sleep disorders. For self-supervised training, we use $W_{Self} = 5sec$. The results are reported in Table 3.2. Note that we combined the recordings of subgroup II and the ones not used for the training from subgroup I into a single test set since they are both from subjects with sleep disorders. We observe that for both experiments $r_{train} = 25\%$ and $r_{train} = 50\%$, the *Phase Swap* model outperforms the supervised baseline for both test sets. In particular, the performance gap on subgroup III is greater than 10%. This can be explained by the fact that sleep disorders can drastically change the sleep structure of affected subjects, which in turn leads the supervised baseline to learn features that are specific to the disorders/subjects present in the training set.

Table 3.1: **Comparison of the performance of the *Phase Swap* model on the SC dataset**. We evaluate both models for various values of r_{train}. For $r_{train} = 100\%^*$ we use all available recordings for the training and the Validation sets. Results with different r_{train} are not comparable.

r_{train}	Experiment	Validation Set	Test set A	Test set B
20%	*Phase Swap*	**84.3%**	**72.0%**	**69.6%**
20%	**Supervised**	79.4%	67.9%	66.0%
50%	*Phase Swap*	**84.9%**	**75.1%**	**73.3%**
50%	**Supervised**	81.9%	71.7%	69.4%
75%	*Phase Swap*	**84.9%**	**77.6%**	**76.1%**
75%	**Supervised**	81.6%	73.7%	72.8%
100%*	*Phase Swap*	**84.3%**	-	-
100%*	**Supervised**	83.5%	-	-

Table 3.2: **Comparison of the performance of the *Phase Swap* model on the ISRUC-Sleep dataset**. We evaluate both models for various values of r_{train}.

r_{train}	Model	Validation set	Test set B (subgroup I + II)	Test set B (subgroup III)
25%	Phase Swap	75.8%	**67.3%**	**62.8%**
25%	Supervised	**75.9%**	63.1%	47.9%
50%	Phase Swap	**76.3%**	68.2%	**67.1%**
50%	Supervised	75.5%	**68.3%**	57.3%

3.3.6 Comparison to the Relative Positioning Task

The Relative Positioning (RP) task was introduced by Banville *et.al.* [9] as a self-supervised learning method for EEG signals, which we briefly recall here. Given x_t and $x_{t'}$, two samples with a window size W and starting points t and t', respectively, the RP task defines its labels as

$$C_{Self}(|h_t - h_{t'}|) = \mathbb{1}\left(|t-t'| \leq \tau_{pos}\right) - \mathbb{1}\left(|t-t'| > \tau_{neg}\right), \tag{3.5}$$

$$\text{where } h_t = E(x_t), h_{t'} = E(x_{t'}). \tag{3.6}$$

$\mathbb{1}(\cdot)$ is the indicator function and τ_{pos} and τ_{neg} are predefined quantities, while $|\cdot|$ denotes the element-wise absolute value operator. Pairs with $C_{Self} = 0$ are discarded.

We compare our self-supervised task to the RP task [9]. For both settings, we use $W_{Self} = 5sec$ and $r_{train} = 20\%$. For the RP task, we choose $\tau_{pos} = \tau_{neg} = 12 \times W_{Self}$. We report the balanced accuracy for all test sets on the SC dataset in Table 3.3. We observe that our self-supervised task outperforms the RP task. This means that the features learned through the PS task allow the model to perform better on unseen data.

Table 3.3: **Comparison between the PS and RP pre-training on the SC dataset.** The SelfSL Acc is the accuracy of the self-supervised task computed on the validation set.

Pre-training	Validation Set	Test set A	Test set B	SelfSL Acc
Supervised	79.4%	67.9%	66.0%	-
PS	**84.3%**	**72.0%**	**69.6%**	86.9%
RP	80.3%	66.2%	65.4%	56.9%

3.3.7 Results on the Sleep Telemetry and CHB-MIT Data Sets

In this section, we evaluate our framework on the ST and CHB-MIT datasets. For the ST dataset, we use $W_{Self} = 1.25sec$, $W_{Sup} = 30sec$, and $r_{train} = 50\%$. For the CHB-MIT dataset, we use $W_{Self} = 2sec$, $W_{Sup} = 10sec$, $r_{train} = 25\%$ and 30 epochs for supervised fine-tuning / training. As shown in Table 3.4, we observe that for the ST dataset, the features learned through the PS task produce a significant improvement, especially on Test Sets A and B. For the CHB-MIT dataset, the PS does not provide the performance gains observed for the previous two datasets. We believe that this is due to the fact that the PS task is too easy on this particular dataset: Notice how the validation accuracy is above 99%. With a trivial task, self-supervised pretraining fails to learn any meaningful feature representations.

In order to make the task more challenging, we introduce a new variant, which we call **PS + Masking**, where we randomly zero out all but six randomly selected channels for each sample during the self-supervised pretraining. The model obtained through this scheme performs best on sets A and B and is comparable to the **Supervised** baseline on the validation set. As for the reason why the PS training was trivial on this particular dataset, we hypothesize that this is due to the high spatial correlation in the CHB-MIT dataset samples. This dataset contains a high number of homogeneous channels (all of them are EEG channels), which in turn result in a high spatial resolution of the brain activity. At such a spatial resolution, the oscillations due to brain activity show a correlation in both space and time [57]. However, our PS operator ignores the spatial aspect of the oscillations. When applied, it often corrupts the spatial coherence of the signal, which is then easier to detect than the temporal phase-amplitude incoherence. This hypothesis is supported by the fact that random

Table 3.4: **Evaluation of the *Phase Swap* model on the ST and CHB-MIT datasets**. The SelfSL Acc is the accuracy of the self-supervised task computed on the validation set.

Dataset	Experiment	Val. Set	Test set A	Test set B	SelfSL Acc
ST	**Supervised**	69.2%	52.3%	46.7%	-
ST	**Phase Swap**	**74.9%**	**60.4%**	**52.3%**	71.3%
CHB-MIT	**Supervised**	**92.6%**	89.5%	58.0%	-
CHB-MIT	**Phase Swap**	92.2%	86.8%	55.1%	99.8%
CHB-MIT	**PS+Masking**	91.7%	**90.6%**	**59.8%**	88.1%

Table 3.5: **Analysis of the effect of the window size W_{Self}.** We compare the performance of the *Phase Swap* model on the SC dataset for various values of the window size W_{Self}.

W_{Self}	Experiment	Validation Set	Test set A	Test set B
1.25sec	**Phase Swap**	84.3%	72.0%	69.6%
2.5sec	**Phase Swap**	**84.6%**	71.9%	70.0%
5sec	**Phase Swap**	83.4%	**72.5%**	**70.9%**
10sec	**Phase Swap**	83.6%	71.6%	69.9%
30sec	**Phase Swap**	83.9%	71.0%	69.2%
-	**Supervised**	79.4%	68.1%	66.1%

channel masking, which in turn reduces the spatial resolution during self-supervised training, yields a lower training accuracy *i.e.* , it is a nontrivial task.

3.3.8 Impact of the Window Size

In this section, we analyze the effect of the window size W_{Self} used for self-supervised training on the final performance. We report the balanced accuracy on all our test sets for the SC dataset in Table 3.5. For all these experiments, we used 20% of the identities in the training set. The capacity of the **Supervised** model $C_{Sup} \circ E$ is independent of W_{Self} (see Section 3.2), and so is its performance. We observe that the best-performing models are the ones that use $W_{Self} = 2.5sec$ for the Validation Set and $W_{Self} = 5sec$ for sets A and B. We argue that the features learned by the self-supervised model are less specific for larger window sizes. The PS operator drastically changes structured parts of the time series, but barely affects pure noise segments. As discussed in Section 3.2, white noise is invariant with respect to the PS operator. With smaller window sizes, most of the segments are either noise or structured patterns, but as the window size grows, its content becomes a combination of the two.

3.3.9 Frozen vs Fine-tuned Encoder

Table 3.6: **Comparison of the four training variants**. We report the balanced accuracy on the SC dataset for the four training variants.

Experiment	Validation Set	Test set A	Test set B
Supervised	79.4%	67.9%	66.0%
Phase Swap	**84.3%**	**72.0%**	**69.6%**
PSFrozen	75.2%	68.1%	67.1%
Random	70.1%	62.1%	63.9%

In Table 3.6, we analyze the effect of freezing the weights of E during supervised fine-tuning. We compare the performance of the four variants described in Section 3.3.2 on the SC dataset. All variants use $W_{Self} = 5sec$, $W_{Sup} = 30sec$, and $r_{train} = 20\%$. As expected, we observe that the *Phase Swap* variant is the best performing one since it is less restricted in terms of training procedure than **PSFrozen** and **Random**. Moreover, the **PSFrozen** outperforms the **Random** variant on all test sets and is on par with the **Supervised** baseline on the Test set B. This confirms that the features learned during pre-training are useful for downstream classification even when the encoder model E is frozen during fine-tuning.

The last variant, **Random**, allows us to disentangle the contribution of the self-supervised task from the prior imposed by the architecture choice for E. As we can see in Table 3.6, the performance of the *Phase Swap* variant is significantly higher than that of the former variant, confirming that the self-supervised task chosen here is the main factor behind the performance gap.

3.3.10 Architecture

Most of the experiments in this chapter use the FCN architecture [152]. In this section, we illustrate that the performance boost of the Phase Swap method does not depend on the neural network architecture. To do so, we also analyze the performance of a deeper architecture in the form of the Residual Network (ResNet) proposed by Humayun *et.al.* [54]. We report in Table 3.7 the balanced accuracy computed using the SC dataset for two choices of $W_{Self} \in \{2.5sec, 30sec\}$ and two choices of $r_{train} \in \{20\%, 100\%^*\}$. The table also contains the performance of the FCN model trained using the PS task as a reference. We do not report the results for the RP experiment using $W_{Self} = 30sec$ as we did not manage to make the self-supervised pre-training

Table 3.7: **Evaluation of the *Phase Swap* model using the ResNet architecture on the SC dataset**. Values denoted with a * are averages across two runs.

r_{train}	W_{Self}	Architecture	Experiment	Val. Set	Test set A	Test set B
20%	5sec	FCN	FCN + PS	84.3%	72.0%	69.6%
20%	5sec	ResNet	Phase Swap	82.1%	**72.5%**	69.6%
20%	5sec	ResNet	RP	72.3%	67.4%	65.9%
20%	-	ResNet	supervised	79.1%*	70.0%*	66.5%*
20%	30sec	ResNet	Phase Swap	**83.6%**	70.7%	**69.3%**
100%*	5sec	FCN	FCN + PS	84.3%	-	-
100%*	5sec	ResNet	Phase Swap	81.2%	-	-
100%*	5sec	ResNet	RP	79.1%	-	-
100%*	-	ResNet	supervised	**84.2%***	-	-
100%*	30sec	ResNet	Phase Swap	**84.2%**	-	-

converge. All ResNet models were trained for 15 epochs for supervised fine-tuning. For $r_{train} = 20\%$, we observe that pretraining ResNet on the PS task outperforms both supervised and RP pretraining. We also observe that for this setting, the model pre-trained with $W_{Self} = 30sec$ performs better on both the validation set and test set B compared to the one pre-trained using $W_{Self} = 5sec$. Nonetheless, the model using the simpler architecture still performs the best on those sets and is comparable to the best performing one on set A. We believe that the lower capacity of the FCN architecture prevents the learning of feature representations that are too specific to the pretext task compared to the ones learned with the more powerful ResNet. For the setting $r_{train} = 100\%^*$, the supervised ResNet is on par with a model pre-trained on the PS task with $W_{Self} = 30sec$. Recall that $r_{train} = 100\%^*$ refers to the setting in which all recording sessions and patients are used for the training set. Based on these results, we can conclude that there is a point of diminishing returns in terms of available data beyond which the self-supervised pretraining might even deteriorate the performance of the downstream classification tasks.

3.4 Discussions

We have introduced the *Phase Swap* pretext task, a novel self-supervised learning approach suitable for biosignals. This task aims to detect when biosignals have mismatching phase and amplitude components. Since the phase and amplitude of white noise are not correlated, the features learned with the *Phase Swap* task do not focus on noise patterns. Moreover, these features exploit signal patterns present both in the amplitude and phase domains. We demonstrate the benefits of learning features from the phase component of biosignals in several experiments and comparisons with competing methods. Most importantly, we find that pretraining a neural network with limited capacity on the *Phase Swap* task builds features with strong generalization capabilities between subjects and recording sessions. Indeed, by designing a pretext task that actively encourages the features to be based on the structured part of the signal and ignores the noise component that might lead to spurrious correlation, we obtain a model that generalizes across subjects even in the low data regimes. One extension of this work that we explore in the next chapter is to incorporate the generalizability constraint explicitly into the training logic of our model.

Chapter 4

Towards Sleep Scoring Generalization Through Self-Supervised Meta-Learning

In this chapter, we introduce a novel meta-learning method for sleep scoring based on self-supervised learning. Our approach aims to build models for sleep scoring that can generalize across different patients and recording facilities, but do not require a further adaptation step to the target data. Towards this goal, we build our method on top of the Model Agnostic Meta-Learning (MAML) framework by incorporating a self-supervised learning (SelfSL) stage and call it S2MAML. We show that S2MAML can significantly outperform MAML. The gain in performance comes from the SelfSL stage, which we base on a general purpose pseudo-task that limits the overfitting to

[1]©2022 IEEE Reprinted, with permission, from A. Lemkhenter and P. Favaro, "Towards Sleep Scoring Generalization Through Self-Supervised Meta-Learning", 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Glasgow, Scotland, United Kingdom, 2022, pp. 2961-2966.

the subject-specific patterns present in the training dataset. We show that S2MAML outperforms standard supervised learning and MAML on the SC, ST, ISRUC, UCD, and CAP datasets.

Sleep is known to play an important role in the mental and physical health of an individual [129] and therefore the development of tools to diagnose sleep quality and common sleep pathologies is fundamental. Sleep is monitored by polysomnography (PSG), *i.e.* , electrical biosignal analysis, such as electroencephalogram (EEG), electromyograph (EMG), electrooculograph (EOG), and electrocardiograph (ECG). The recorded biosignals are split into 30-second intervals (epochs) and annotated by clinicians into several categories, such as wake (W), non-rapid eye movement (NREM: N1, N2, and N3), and rapid eye movement (REM). This task is difficult and time-consuming. Therefore, the ability to do so consistently and on a large scale through automation has an important impact on medical research and clinical practice [156].

Toward this purpose, machine learning methods have been introduced as a way to achieve automatic sleep scoring [39, 114, 65]. However, these methods are still not widely adopted among sleep practitioners. One limitation is that current methods for sleep stage classification typically experience a decrease in performance on data obtained from new cohorts of patients. The main reason for this decay is the large variability of biosignals between subjects and sessions. This variability stems from experimental factors such as differences in the recording equipment and protocol (e.g., the number and placement of the electrodes) or physiological factors such as age, prognosis, and medication. While this problem is commonly known among practitioners through direct experience, we illustrate it quantitatively in detail in our experimental analysis.

An approach to overcome this limitation is transfer learning, in which a classification model is adapted to the target cohort through further training. This approach also motivated the recent work MetaSleepLearner [8], which builds on meta-learning in the case of few-shot adaptation. MetaSleepLearner requires only a small set of annotations of the target dataset and a limited amount of training when new data become available.

However, as we show in our experiments, the scheme used in MetaSleepLearner, *i.e.* , MAML, still runs the risk of overfitting even if the adaptation is performed on a large dataset. Moreover, we find the few-shot learning scenario, or more in general, the transfer learning case, not practical because practitioners would need further training and/or to provide annotation to adapt a classifier to new target data. Thus, in this chapter, we propose to build a single sleep staging model and then use it "as is" on new data. To avoid overfitting, we combine the Model Agnostic Meta-Learning (MAML) framework with self-supervised learning (SelfSL) [80]. SelfSL has the advantage of not requiring annotation and it can be designed to train models that overfit less to the training data. With a slight abuse of notation, we refer to the proposed setting as *zero-shot learning*, to emphasize that no new training or annotation is needed with new data. To the best of our knowledge, the zero-shot learning scenario has not been explored so far in the literature for sleep scoring. We test our proposed method on several datasets and find that the use of SelfSL with MAML yields state-of-the-art performance in zero-shot learning.

4.1 Related Works

In Sections 2.2 and 2.3, we introduce useful concepts from the self-supervised learning and meta-learning literature. Relevant to this work are the deep learning works for sleep scoring covered in Section 2.1.4, and applications of MAML [38] to EEG classification tasks such as sleep scoring [8], brain-computer interface [83] and emotion prediction [97].

Since the main focus of this work is to show the potential of combining the *Phase Swap* introduced in Chapter 3 and meta-learning [38], we adopt an existing baseline, DeepSleepNet-Lite [39], as our network architecture of choice, as it is efficient to train and acheives competitive performance on the considered benchmark. Indeed, our proposed S2MAML method is agnostic to the choice of the used architecture, so we focus our experimental effort on showcasing its generalizability across datasets instead of performing an exhaustive neural architecture search.

4.2 Methods

4.2.1 Datasets

In this work, we used five different sleep scoring datasets.

Sleep Cassette (SC). It is a subset of the Expanded Sleep-EDF Database [65] . It contains PSG sleep recordings obtained between 1981 and 1991. It includes recordings from 78 healthy subjects between the age of 25 and 101, with two recordings per person for most of them.

Sleep Telemetry (ST). It is another subset of the Expanded Sleep-EDF Database [65]. It was collected as part of a 1994 study on the effect of temazepam on sleep. It contains PSG recordings from 22 subjects with one session per individual.

Old datasets like SC & ST allows us to investigate generalization from/to recordings with different signal quality.

ISRUC. It is a publicly available sleep dataset [66]. It consists of PSG recordings obtained at the Sleep Medicine Center of the Hospital of Coimbra University (CHUC) between 2009 and 2013. This database has three different subsets:

- Subgroup-I contains one recording per subject for 100 individuals with sleep disorders;
- Subgroup-II contains two recordings per subject for 8 individuals with sleep disorders;
- Subgroup-III contains one recording per subject for 10 healthy individuals.

University College Dublin Sleep Apnea Database (UCD) It is a 2011 database collected at St. Vincent's University Hospital [49]. It contains one PSG recording per subject for 25 individuals with suspected sleep disorder breathing.

Cyclic Alternating Pattern (CAP) Sleep Database It is a collection of one recording per subject for 108 individuals with varying conditions. It contains 10 healthy subjects, 40 diagnosed with NFLE, 22 affected by RBD, 10 with PLM, 9 insomniac, 5 narcoleptic, 4 affected by SDB and 2 by bruxism [143]. It was published in 2001.

4.2.2 Data Preprocessing

Out of all the signals available in each recording, we keep the EEG , EMG and EOG channels. All signals are re-sampled at 102.4Hz. This allows us to represent a 30sec epoch with 3072 time points, which is more compact and closer to the original sampling frequency compared to the commonly adopted 128Hz. This is sufficient since most spectral features classically used for sleep scoring are at lower frequency bands.

Since the convolutional architecture we adopted in our experiments requires a constant number of channels as input, we fix that number to 9. If the recording contains more than 9 channels, which is the case for ISRUC and CAP, we randomly select a subset of them. Otherwise, if the recording does not have enough channels, we add dummy ones that are all zeros. The channels are shuffled before being fed to the model. We normalize each channel to have zero mean and unit standard deviation.

4.2.3 Data Split

To evaluate our models, we choose two different train/evaluation splits. We first split each dataset by **subjects**, then we randomly split each recording into samples of 3×30 seconds. This allows us to have an evaluation set that contains subjects that were **seen**

Table 4.1: **Diagram illustrating our evaluation sets**. In a setting with 4 subjects and 4 samples per subject, we define 3 sets: Train, Eval. Seen and Eval.Unseen.

	Sample 1	**Sample 2**	**Sample 3**	**Sample 4**
Subject 1	Train	Eval. Seen	Train	Train
Subject 2	Train	Train	Train	Eval. Seen
Subject 3	Train	Train	Train	Eval. Seen
Subject 4	-	-	-	Eval. Unseen

during training and another evaluation set containing **unseen** ones. Both splits follow a 75%-25% ratio. An illustration of both splits is shown in Table 4.1 (©2022 IEEE).

4.2.4 Notation

We define the mapping $E : x \mapsto h$ as the encoding of the input signal x into a feature vector h. The associated trainable parameters are denoted by Θ^E. We denote the mapping from the feature vector h to the predicted class label $\hat{y}$ for the supervised and self-supervised settings as C^{SL} and C^{SelfSL}, respectively. Their associated trainable parameters are Θ^C_{SL} and Θ^C_{SelfSL}, respectively. The predicted labels of the model are therefore

$$\hat{y}^{SL/SelfSL} = C^{SL/SelfSL}(E(x)). \tag{4.1}$$

In all experiments, models are trained by minimizing the average cross-entropy loss given by

$$\mathcal{L}(T, \Theta^E, \Theta^C) = \frac{1}{|T|} \sum_{t \in T} \frac{1}{|t|} \sum_{(x,y) \in t} - \sum_{c=1}^{N_c} y_c log(\hat{y}_c), \tag{4.2}$$

where y, $\hat{y}$, N_c and T are respectively the true labels, the model predictions, the numbers of classes and a set of tasks t consisting of signal-label pairs. Note that we represent the true labels as a one-hot encoding vector. v_c refers to the c-th entry in the vector $v \in \mathbb{R}^{N_c}$. We frame the sleep scoring problem as a five-way classification with the five classes being: Wake (W), N1, N2, N3, and REM.

4.2.5 Self-Supervised MAML (S2MAML)

Model Agnostic Meta-Learning (MAML) is a meta-learning algorithm, where a given model is trained on a large variety of tasks with the goal of generalizing to novel tasks through fast-adaptation, *i.e.* few-shot learning or with no adaptation, *i.e.* zero-shot learning. In this work, we investigate the benefit of using meta-learning jointly with self-supervised learning to improve generalization to unseen subjects and datasets.

The problem that our proposed model solves can be described using the following bilevel formulation

$$\begin{aligned} \Theta^{*E}, \Theta^{*C} &= \underset{\Theta^E, \Theta^C_{SL}}{\operatorname{argmin}} \mathcal{L}(T_{SL}, \hat{\Theta}^E, \Theta^C_{SL}) \\ \text{s.t. } \hat{\Theta}^E, \hat{\Theta}^C_{SelfSL} &= \underset{\Theta^E, \Theta^C_{SelfSL}}{\operatorname{argmin}} \mathcal{L}(T_{SelfSL}, \Theta^E, \Theta^C_{SelfSL}), \end{aligned} \tag{4.3}$$

where the model E is optimized to learn useful self-supervised representations of the set of tasks T_{SelfSL}. The bilevel optimization favors representations that generalize well to the supervised outer problem on tasks T_{SL}.

Algorithm 1 outlines our adaptation of MAML, which we call **S2MAML**. Given K datasets $\{D_k\}_{k=1}^K$, we randomly sample n_{tasks} tasks from each one. Each task $t = \{(x_j, y_j)\}_{j=1}^{N_s}$ is defined as a set of signal and label pairs belonging to the same subject in a given dataset. The total set of tasks T is then split into a meta-training set T^{tr} and a meta-validation set T^{val}. Each MAML iteration consists of an inner and an outer optimization problem. In the inner problem, Θ^E_{in} is initialized with the values of Θ^E. Both Θ^E_{in} and Θ^C_{SelfSL} are optimized for n_{in} iterations with respect to the self-supervised loss $\mathcal{L}^{in}$ computed on the meta-training set. To do this, we need to generate a set of self-supervised tasks T^{SelfSL} based on T^{tr}. The details of this step are described in Section 4.2.6.

The weights Θ^E are then updated in the outer problem by minimizing the supervised loss $\mathcal{L}^{out}$ (see Algorithm 1) computed on the meta-validation set. The gradient for the Θ^E update is calculated at Θ^E_{in}, and not at Θ^E, because we use the first-order approximation version of MAML [38].

The goal of this design is to encourage the model to learn self-supervised general purpose features in the inner problem that would generalize well to the outer supervised problem computed on novel tasks, *i.e.* unseen subjects.

4.2.6 PhaseSwap

For our self-supervised training, we choose *Phase Swap* (PS) introduced in Chapter 3. Our choice is motivated by two reasons. First, PS has been shown to improve generalization to unseen subjects, making it a strong candidate for our approach. Second, PS can be defined on the same time scale as the supervised task. In fact, other self-supervised methods, such as relative positioning (RP) or contrast positional coding (CPC) [10], require a longer temporal context, which would complicate the training loop. Since the main focus of this work is to highlight the potential of using self-supervised learning in a meta-learning setting, we opted for the simplest self-supervised loss.

Algorithm 1 S2MAML (©2022 IEEE)

```
Require: ${D_k}_{k=1}^{K}, \Theta^E, \Theta^C_{SelfSL}, \Theta^C_{Sup}, \lambda_{in}, \lambda_{out}$
  while not converged do
    $T \leftarrow \{\}$
    for $k$ in 1..$K$ do
      for $i$ in 1..$n_{tasks}$ do
        $t \leftarrow$ sample_task($D_k$)
        $T \leftarrow T \cup \{t\}$
      end for
    end for
    $T^{tr}, T^{val} \leftarrow$ split($T$)
    $\Theta^E_{in} \leftarrow \Theta^E$
    for $i$ in 1..$n_{in}$ do
      $T^{tr}_{SelfSL} \leftarrow$ generate_ssl_task($T^{tr}$)
      $\mathcal{L}^{in} \leftarrow \mathcal{L}(T^{tr}_{SelfSL}, \Theta^E_{in}, \Theta^C_{SelfSL})$
      $\Theta^E_{in} \leftarrow \Theta^E_{in} - \lambda_{in} \nabla_{\Theta^E_{in}} \mathcal{L}^{in}$
      $\Theta^C_{SelfSL} \leftarrow \Theta^C_{SelfSL} - \lambda_{in} \nabla_{\Theta^C_{SelfSL}} \mathcal{L}^{in}$
    end for
    $\mathcal{L}^{out} \leftarrow \mathcal{L}(T^{val}, \Theta^E_{in}, \Theta^C_{Sup})$
    $\Theta^E \leftarrow \Theta^E - \lambda_{out} \nabla_{\Theta^E_{in}} \mathcal{L}^{out}$
    $\Theta^C_{Sup} \leftarrow \Theta^C_{Sup} - \lambda_{out} \nabla_{\Theta^C_{Sup}} \mathcal{L}^{out}$
  end while
```

More specifically, PS is defined as a binary classification problem, where a model is trained to distinguish between samples x and x_{PS} defined as

$$x_{PS} = \mathcal{F}^{-1}\left[\left|\mathcal{F}(x)\right| \odot \measuredangle\mathcal{F}\left(x'\right)\right], \tag{4.4}$$

where x and x' are two different samples. For a complex scalar $z \in \mathbf{C}^*$, the absolute value $|.|$ and angle $\measuredangle$ operators are defined such that $z = |z|e^{i\measuredangle z}$.

In Algorithm 1, $T^{tr} = \{t_i\}_{i=1}^{K \times n_{tasks}/2}$ is a set of supervised tasks. For each task $t \in T^{tr}$, the function **generate_ssl_task** generates a new task t_{SelfSL} to be included in T^{tr}_{SelfSL}. For each signal-label pair $(x, y) \in t$, t_{SelfSL} includes $(x, y_{SelfSL} = 0)$ and its phase-swapped counterpart $(x_{PS}, y_{SelfSL} = 1)$.

4.2.7 Architecture Choice

For our experiment, we use DeepSleepNet-Lite [39] as our architecture of choice. It consists of two parallel convolutional neural networks using sets of small and large filters for the first layer, respectively. The output of the two networks is concatenated into a single vector h and fed into a softmax layer that maps it to the predicted class. The input x of the network is a segment of 90 seconds, *i.e.* three consecutive epochs of 30 seconds each. We chose this architecture for its simplicity, its shorter temporal context, and the fact that it does not require the power spectrum as input.

4.2.8 Baselines and Training Hyper-parameters

In all experiments, we compare the performance of our S2MAML model to two other baselines: A supervised classification model without meta-learning and a MAML based training similar to ours, but where we replace the self-supervised problem in the inner loop with a supervised one. We refer to these models as **SL** and **MAML** respectively.

Unless stated otherwise, each task t contains 8 samples from the same subjects. n_{tasks}, n_{in}, λ_{out} and λ_{in} are set to 32, 1, 10^{-4} and $5 \cdot 10^{-5}$ respectively and each model is trained for 20 full iterations in all databases considered. We use Adam [69] as our optimizer with its default hyperparameters. Our models are implemented using Pytorch[2] and run on a single NVIDIA 1080Ti GPU. We observe no significant differences between the computation times of all models both in inference and in training.

We adopt the same label smoothing regularization as used by [39] with their suggested tuning.

4.2.9 Evaluation Metrics

We use macro F1 (MF1) as an evaluation metric for our experiments. Macro F1 is defined as

$$\mathrm{MF1} = \frac{1}{N_c}\sum_{c=1}^{N_c} \mathrm{F1}_c = \frac{1}{N_c}\sum_{c=1}^{N_c} \frac{2\mathrm{P}_c \times \mathrm{R}_c}{\mathrm{P}_c + \mathrm{R}_c} \tag{4.5}$$

where N_c, P_c and R_c are, respectively, the number of classes, the precision and recall for the class c. It is the average F1 score per class, where the F1 score is defined as the harmonic mean of precision and recall. We choose MF1, instead of the classic F1 score, as it is a better metric when the dataset has a significant class imbalance, as is the case for sleep scoring. All reported MF1 scores are averaged across a 4-way cross-validation split.

Table 4.2: **Cross-validation MF1 Scores for the 3 vs 5 setting on seen subjects.** Avg(S) refers to the average MF1 across all seen evaluation sets (©2022 IEEE).

Run	**CAP**	**ST**	**ISRUC**	**Avg(S)**
S2MAML	**68.8**	74.8	**74.7**	**72.8**
MAML	66.4	71.3	73.3	70.3
SL	55.0	**75.5**	66.7	65.7

Table 4.3: **Cross-validation MF1 Scores for the 3 vs 5 setting on unseen subjects**. Avg(U1) and Avg(U2) refer to the averages MF1s for unseen subjects across seen (CAP, ISRUC, ST) and unseen (UCD, SC) datasets respectively. Avg(U) is the average MF1 across all unseen sets (©2022 IEEE).

Run	**CAP**	**ST**	**ISRUC**	**Avg(U1)**	**SC**	**UCD**	**Avg(U2)**	**Avg**
S2MAML	**56.5**	65.2	**70.3**	**64.0**	**41.1**	**43.7**	**42.4**	**55.4**
MAML	55.0	**65.3**	68.9	63.1	34.4	42.1	38.2	53.1
SL	46.4	63.2	63.0	57.5	30.3	37.9	34.1	48.1

4.3 Results

4.3.1 Generalization to Novel Databases: 3 vs 5

In this set of experiments, we compare the performance of S2MAML to the two baselines when training on 3 of 5 of the considered databases. This allows us to evaluate the performance of our model on completely unseen cohorts of subjects belonging to different databases (see Section 4.2.3). More specifically, we train using ST, CAP, and ISRUC and evaluate on all five datasets. We report the performance of all models on both evaluation sets with **seen** and **unseen** subjects in Tables 4.2 and 4.3.

For seen subjects, we observe that our model outperforms the two baselines (supervised and MAML training) on average, as well as on CAP and ISRUC. On average the performance gap is 2.5% compared to MAML and 7.1% compared to supervised training. This shows that meta-learning-based methods are generally better suited for overcoming intra-subject variability, and that self-supervision is a powerful tool to further reduce that performance gap.

For unseen subjects, we observe that our model outperforms both baselines on most datasets and on average. We also find that the meta-learning-based models outperform the supervised baseline, similar to the results on seen subjects. More importantly, the performance gap between our S2MAML and MAML is wider on held-out datasets. Although MAML generalizes better to unseen subjects from the databases used for

[2]https://pytorch.org/

training compared to the supervised baseline, it generalizes less to held out databases compared to our model. In other words, our S2MAML is not only better suited to deal with variability between subjects, but it is also better suited to deal with variability between cohorts. We discuss the low performance on ST in Section 4.3.3.

4.3.2 Generalization in a Data Abundant Setting: All vs All

In this set of experiments, we compare the performance of S2MAML to our two baselines when trained on all databases jointly. This allows us to highlight the benefit of our algorithm in a setting where a large quantity of labeled recordings are available. The MF1 scores of all models on both seen and unseen subjects are reported in Table 4.4.

Table 4.4: **Cross-validation MF1 Scores for the All vs All setting**. We also report the average MF1 across all databases (©2022 IEEE).

Run	Subjects	CAP	ST	ISRUC	SC	UCD	Avg
S2MAML	Seen	**82.1**	**85.0**	**88.8**	**86.3**	**90.4**	**86.5**
MAML	Seen	80.2	81.5	86.1	84.1	89.4	84.3
SL	Seen	59.3	83.2	71.2	82.1	68.6	72.9
S2MAML	Unseen	**67.9**	**73.7**	**82.7**	**83.8**	**70.7**	**75.8**
MAML	Unseen	65.7	69.2	80.8	80.8	69.8	73.3
SL	Unseen	50.8	70.0	67.2	79.5	55.0	64.5

We observe that our model outperforms both baselines on all datasets as well as on average for both evaluation settings. This shows that the generalization advantage of our model does not disappear when scaling up the amount of available data. In the deep learning literature, scaling up the amount of training data is a common practice used to improve the generalization of artificial neural networks. This relies on the implicit assumption that with enough data, one is able to obtain a training set that is similar in distribution to the evaluation set and contains most sources of variability that can be encountered. However, in the case of physiological signals, this assumption may not hold as well. A new individual will always have subject-specific sleep patterns, and the inter-dataset variability will always remain a challenge as long as hardware/software recording pipelines keep evolving.

In the previous section, we have split the data by subjects, which is not a common practice. We did so to illustrate a more extreme setting for generalization. In this section, we also obtain performance on seen subjects, as done in the literature, so that it is easier to compare it with prior work. Although the main focus of our work is to reduce the generalization gap between subjects and datasets, the MF1s reported on seen subjects are comparable or better than state-of-the-art methods in the literature.

Our model achieves an MF1 score of 86.3% and 85.0% compared to 79% and 76% for U-Sleep [113] in SC and ST, respectively. However, the numbers are not directly comparable due to the difference in the randomness of the splits. For this reason, and in order to keep our results focused on the generalization problem, we chose to omit the numbers reported by other prior works from our tables.

4.3.3 Disparity Between Datasets: One vs All

In this set of experiments, we compare the performance of the different models on unseen databases when trained only on a single one. This represents a worst-case scenario, where one has access to a very limited number of subjects, and therefore learning to generalize becomes much more challenging. Since the different databases considered in this study have different sizes, we choose to equalize experiments by training for a fixed number of gradient updates 5000, instead of looping through the training set 20 times. The goal of these experiments is to gauge how similar or dissimilar the databases considered in this work are. In other words, our goal is to confirm that generalizing from one set to the others is indeed a challenging task and that each database has its particularities.

We report the MF1 scores obtained in Table 4.5. For all datasets and the three models considered, we observe that the performance drops significantly on unseen datasets. One additional noteworthy observation is that out of all combinations, models trained on SC/ST and tested on others and vice versa seem to generalize the least. On the other hand, generalizing between ST and SC seems more feasible. This may be due to the fact that ST/SC were collected a few decades ago or to the fact that they include EEG electrodes that are not common in the other three databases. We believe that this observation may explain why both meta-learning models struggle compared to the SL baseline on ST as reported in Tables 4.2 and 4.3.

Overall in this setting, the performance across different methods does not indicate a clear winner. Given the restricted number of subjects per dataset, all methods struggle to learn features that generalize well to new cohorts. However, on average across all possible combinations, S2MAML and MAML are slightly above with 29.8% and 29.9%, respectively, compared to 29.4% for the SL baseline.

4.3.4 Effect of λ_{in}

In this section, we study the effect of λ_{in} on our model and the MAML baseline. We train both our model and the MAML baseline in the **3 vs 5** setting described in Section 4.3.1 for $\lambda_{in} \in \{10^{-3}, 5 \cdot 10^{-5}\}$. Tables 4.6 and 4.7 report the obtained MF1 scores for seen and unseen subjects, respectively. We observe that while the value of λ_{in} has little effect on the performance of our model, setting it at 10^{-3} greatly

Table 4.5: **Cross-validation MF1 Scores for models trained on one dataset.** Each row block corresponds to models trained on a single dataset and evaluated on unseen subjects from the same or other datasets (©2022 IEEE).

Train \ Test	Model	ISRUC	SC	ST	CAP	UCD
ISRUC	S2MAML	**76.0**	27.4	9.7	**25.8**	12.2
	MAML	74.7	**24.5**	**10.3**	23.3	15.0
	SL	71.7	17.7	10.3	21.6	**16.1**
SC	S2MAML	7.2	74.1	41.5	6.3	7.4
	MAML	8.1	75.2	**44.3**	**19.4**	7.5
	SL	**10.1**	**75.5**	43.7	6.6	**8.2**
ST	S2MAML	14.9	28.7	67.7	17.2	**20.2**
	MAML	19.3	**28.9**	68.2	**18.4**	19.2
	SL	**21.7**	28.3	**68.3**	18.2	19.6
CAP	S2MAML	28.8	**27.4**	13.9	**58.1**	**48.5**
	MAML	28.5	24.6	13.2	54.3	46.4
	SL	**29.8**	25.4	**14.4**	49.0	41.7
UCD	S2MAML	23.2	18.6	4.5	24.2	**62.2**
	MAML	20.0	**18.5**	4.9	20.9	59.4
	SL	29.9	18.5	**6.1**	**25.7**	57.7

reduces the performance of the MAML baseline. By setting λ_{in} to a higher value, we put more emphasis on the convergence in the meta-train set, *i.e.* , in the inner loop. This confirms that using *Phase Swap* as a self-supervised task in the inner loop, *i.e.* , in the meta-train set, is less prone to learning subject-specific features and thus generalizes better compared to its supervised counterpart. Additionally, this shows that our method is more robust to the choice of the hyper-parameter λ_{in}.

4.4 Discussions

With the increasing popularity of deep learning methods, more and more artificial neural network architectures have been proposed for automatic sleep scoring. Reliable automatic sleep scoring models have the potential to speed up sleep research and make it more accessible by reducing the cost of manual annotations and enabling a more advanced closed-loop system. However, one important requirement for such models is that they should maintain their level of performance across sessions, subjects, and hardware/software recording settings. Our work positions itself as a step forward toward achieving this goal. By leveraging both meta-learning and self-supervised

Table 4.6: **Analysis of the effect of λ_{in} for unseen subjects**. We report the cross-validation MF1 Scores on unseen subjects for models trained in the 3 vs 5 setting for different values of λ_{in} (©2022 IEEE).

Run	λ_{in}	**CAP**	**ST**	**ISRUC**	**Avg**	**SC**	**UCD**	**Avg**
S2MAML	10^{-3}	60.0	**70.2**	67.4	65.9	**34.3**	**49.0**	**41.7**
S2MAML	$5 \cdot 10^{-5}$	**61.9**	70.1	**68.4**	**66.8**	32.6	46.9	39.8
MAML	10^{-3}	23.1	25.4	33.1	27.2	15.9	10.4	13.2
MAML	$5 \cdot 10^{-5}$	**59.2**	**67.9**	**65.4**	**64.2**	**25.9**	**49.4**	**37.7**

Table 4.7: **Analysis of the effect of λ_{in} for seen subjects**. We report the cross-validation MF1 Scores on seen subjects for models trained in the 3 vs 5 setting for different values of λ_{in} (©2022 IEEE).

Run	λ_{in}	**CAP**	**ST**	**ISRUC**	**Avg**
S2MAML	10^{-3}	68.3	70.8	73.4	71.0
S2MAML	$5 \cdot 10^{-5}$	**68.8**	**74.8**	**74.7**	**72.8**
MAML	10^{-3}	16.6	24.5	20.5	20.5
MAML	$5 \cdot 10^{-5}$	**66.4**	**71.3**	**73.2**	**70.8**

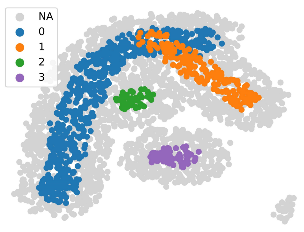
learning, our S2MAML is able to reduce the performance drop associated with both intrasubject variability, *i.e.* unseen subjects from seen datasets, and intra-database variability, *i.e.* on unseen datasets. Addressing the generalizability of such models is an important milestone towards the wider adoption of such models, especially in a medical context. Training data will always lag behind that of collected patient data in both scale and diversity. Therefore, it is of the utmost importance to ensure the robustness and generalizability of such models across settings.

Chapter 5

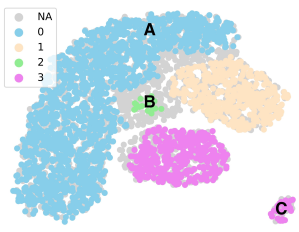
Distribution-Aware Label Refinement for Imbalanced Semi-Supervised Learning

In this chapter, we introduce SemiGPC, a distribution-aware label refinement strategy based on Gaussian Processes where the predictions of the model are derived from the labels posterior distribution. Unlike other buffer-based semi-supervised methods such as CoMatch [84] and SimMatch [167], our SemiGPC includes a normalization term that addresses imbalances in the global data distribution while maintaining local sensitivity. This explicit control allows SemiGPC to be more robust w.r.t. confirmation bias, especially under class imbalance. We show that SemiGPC improves performance when paired with different Semi-Supervised methods such as FixMatch [133], ReMixMatch [14], SimMatch [167] and FreeMatch [151] and different pre-training strategies including MSN [6] and Dino [20]. We also show that SemiGPC achieves state-of-the-art results under different degrees of class imbalance on standard CIFAR10-LT/CIFAR100-LT especially in the low data regime. Using SemiGPC also results in an increase in the average accuracy of approximately 2% compared to a new competitive baseline on the more challenging benchmarks SemiAves, SemiCUB, SemiFungi [140] and Semi-iNat [139].

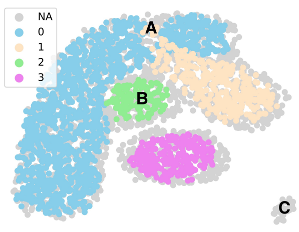
Semi-Supervised Learning offers a more cost effective alternative to fully supervised learning when scaling up the data collection process. Current state the of the art semi-supervised methods rely on self-learning by generating pseudo-labels for the unlabeled samples. However, pseudo-labels can also hurt the final performance when they introduce persistent incorrect predictions, a problem known as confirmation bias.



(a) Initial labels



(b) Similarity-based predictions



(c) SemiGPC predictions

Figure 5.1: **SemiGPC pseudo-labeling with class imbalance.** (a) Four-class dataset containing unlabeled (gray) and labeled samples (in color). (b) and (c) show the labels propagated according to the similarity-based aggregate [84] and SemiGPC methods. (c) In A, the initial labels are mixed, so SemiGPC is more conservative there (many samples are not pseudo-labeled at the current threshold level). In B, SemiGPC is able to propagate the labels of the minority green class despite being surrounded by the majority blue class. In C, SemiGPC assigns low confidence to the set of outliers (labels are not propagated). On the contrary, the similarity-based approach expands the majority classes at the expense of the minority classes, *cf.* B and C.

In particular, self-learning can bias the label distribution if the data is imbalanced. To address this, recent works such as CoMatch [84] and SimMatch [167] rely on a buffer of samples to refine the predicted pseudo-labels. However, no countermeasure is adopted to globally balance the data in the memory bank. As such, the resulting refined pseudo-labels are plagued by the class imbalances present in the unlabeled data. To overcome these limitations, we introduce SemiGPC, a novel semi-supervised learning method that generates pseudo-labels using a distribution-aware label-refinement strategy. This distribution awareness stems from the use of Gaussian Processes, which account for local data concentration disparities and counteracts them. This results in more robust pseudo-labels especially for minority classes and outliers as shown in Figure 5.1. In particular, SemiGPC correctly assigns nearby points to the minority classes despite the larger count of the majority class at a bigger scale, i.e. it has a better local sensitivity, while remaining faithful to the global data distribution. SemiGPC is flexible and can be used on top of previous label-refinement schemes on other semi-supervised methods. Furthermore, we show that the similarity-based pseudo-labels heuristics used in SimMatch and CoMatch can be cast as a special case of SemiGPC. To improve the computational efficiency of our method and allow for fast batched updates essential for Semi-Supervised learning methods, we pair SemiGPC with a batched online update rule that significantly reduces its forward pass cost ($\times$7.5 speed-up). We show the benefit of using SemiGPC on top of semi-supervised algorithms such as Fix-

Match [133], ReMixMatch [14], SimMatch [167] and FreeMatch [151] ($\sim 0.8\%$ average improvement) and different self-supervised pre-training strategies such as MSN [6] and Dino [20] resulting in a $\sim 1.3\%$ average improvement. This highlights the general purpose nature of SemiGPC as a relevant extension for semi-supervised methods based on label refinement strategies. We experimentally show that SemiGPC is capable of achieving state of the art results on CIFAR10-LT ($\geq +\ 7.65\%$) and CIFAR100LT ($\geq +\ 1.84\%$) as well as the more challenging semi-supervised benchmarks SemiAves, SemiCUB, SemiFungi, and Semi-iNat ($\sim +1.92\%$ compared to our baseline and $\sim +20.52\%$ compared to the numbers reported in the literature [139, 140]). We also show that SemiGPC is able to narrow the gap between the high and low data regimes with 10/100x fewer labeled samples as we report a 45% and 32% relative improvement over the baseline across regimes for CIFAR10-LT and CIFAR100-LT, respectively.

5.1 Related Works

We provide an overview of the relevant semi-supervised literature in Section 2.4. Gaussian Processes (GPs) are a class of non-parametric function approximation methods fully characterized by their mean and kernel functions. Given a set of observations and their corresponding measurements, GPs define a posterior distribution over the measurements for new observations. Their ability to explicitly model uncertainty makes them a natural fit for Semi-supervised learning. Early works such as Lawrence and Jordan [75] introduce GPs in the context of semi-supervised learning by assuming that the data density in regions between the class-conditional densities should be low while [130] leverages GPs to model the relationship between labeled and unlabeled samples by incorporating the geometry of the latter in the construction of the global kernel function. However, such early works are limited to toy datasets due to the computational cost of GPs. The more recent UaGGP work [88] proposed to address uncertainty caused by erroneous neighborhood relationships in the context of graph-based semi-supervised learning by leveraging the ability of GPs to generalize well from few samples. NP-Match [146] proposed Neural Processes instead of GPs as a probabilistic model for uncertainty estimation which, in turn, allows for better computational efficiency compared to MCDropout [41]. Beyond Semi-Supervised Learning, Gaussian Processes have been used alongside neural networks in multiple other fields. DGPNet [61] relies on GPs in the context of dense few-shot segmentation to capture complex appearance distributions while [79] leverages GPs for fast and accurate uncertainty estimates in robotics systems. Furthermore, different works draw parallels between Gaussian Processes and Neural Networks by interpreting the activation functions of the latter as interdomain inducing features [35] or by proving a correspondence between the two classes of models [157].

5.2 SemiGPC

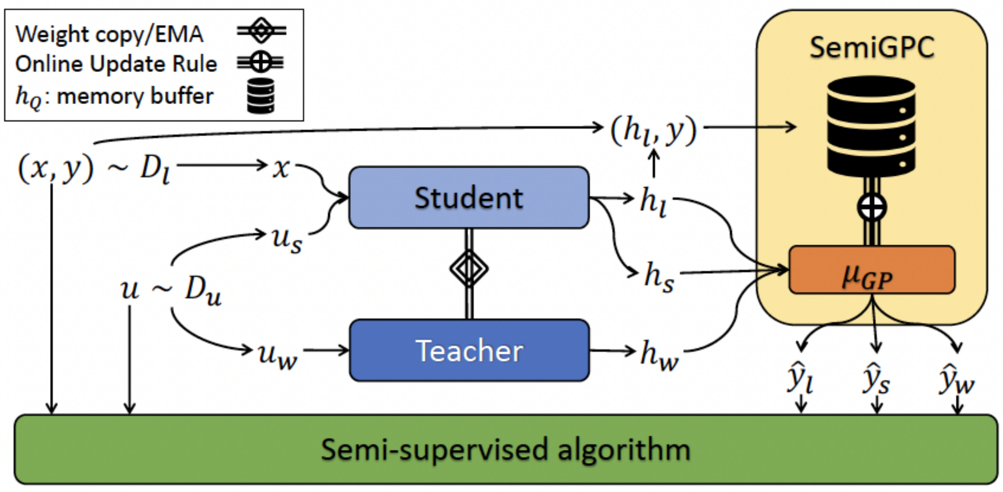


Figure 5.2: **SemiGPC Outline.** x, u_s, u_w, h and y are the labeled, strongly/weakly augmented unlabeled samples, their feature vector and the ground truth labels respectively. The SemiGPC buffer is used to derive the model predictions $\hat{y}$.

In this section, we present the basic framework of consistency-based semi-supervised learning, how it can be extended with a memory buffer, and analyze where confirmation bias comes into play. We then introduce our Gaussian Process-based classifier, SemiGPC, and highlight its advantages over other classifiers using a toy example. The general outline of SemiGPC is shown in Figure 5.2. In the following, we denote the labeled dataset with $\mathcal{D}_l = \{(x_i, y_i)\}_{i=1}^{n_l}$ and the unlabeled one with $\mathcal{D}_u = \{(x_i)\}_{i=1}^{n_u}$ where $x \in X$ are RGB images and $y \in \mathbb{R}^C$ are labels belonging to a fixed set of concepts C. We indicate a feature extractor with $h : X \to Z$ where $Z = \mathbb{R}^d$ and d is the dimension of the feature space, and call the classification head $g : Z \to \mathbb{R}^C$. The model predictions are defined as

$$\hat{y}(x) = g \circ h(x). \tag{5.1}$$

5.2.1 Consistency-based Semi-Supervised Learning

Given labeled and unlabeled datasets $\mathcal{D}_l$ and $\mathcal{D}_u$, consistency-based semi-supervised methods [133, 84, 14, 167, 151] rely on the labeled set and high-confidence pseudo-labels computed on the unlabeled set. Model predictions are computed for strongly

$\text{aug}_s(x)$ and weakly $\text{aug}_w(x)$ augmented views of a given image x, as follows:

$$\hat{y}^s(x) = \hat{y}(\text{aug}_s(x)), \qquad \hat{y}^w(x) = \hat{y}(\text{aug}_w(x)). \tag{5.2}$$

We refer to [133, 84, 167] for typical strong and weak data augmentations. In this work, we adopt those used in FixMatch [133]. More precisely, the labeled and unlabeled losses are defined as

$$\mathcal{L}_l = H(\hat{y}^w(x), y); \qquad (x, y) \in \mathcal{D}_l \tag{5.3}$$

$$\mathcal{L}_u = \mathbb{1}[conf(x) > \tau] H(\hat{y}^s(x), f(\hat{y}^w(x))); \qquad x \in \mathcal{D}_u \tag{5.4}$$

$$\text{with } conf(x) = \max[softmax(\hat{y}^w(x))], \tag{5.5}$$

where H is the cross-entropy loss, τ is the confidence threshold specifying which unlabeled samples to use and $f : \mathbb{R}^C \to \mathbb{R}^C$ is a label refinement function (*e.g.* see [14]). Model confidence is defined as the maximum of the softmax vector. Different choices of f include the identity function (no refinement), onehot encoding (hard pseudo-labels used in FixMatch [133]), temperature sharpening [14], etc. For a linear classification head, such pseudo-labels are sensitive to outliers in the sense that a new unlabeled sample located far from the labeled data can have high confidence, *cf.* Figure 5.3a. To overcome such limitation, works such as SimMatch [167] and CoMatch [84] propose to ground their pseudo-labels using a memory buffer during training. The buffer, (h_Q, y_Q), is a set of N_Q feature vectors of weakly augmented labeled samples using aug_w, *i.e.*

$$h_Q := \{h(\text{aug}_w(x_l)); x_l \; sim \mathcal{D}_l\} \in \mathbb{R}^{n \times d}\}, \tag{5.6}$$

and their associated labels. Then, the smoothed pseudo-label (output of f) for any given input x is defined as

$$f(\hat{y}(x)) = (1 - \alpha)\hat{y}(x) + \alpha \hat{y}^{sim} \tag{5.7}$$

$$\text{with } \hat{y}^{sim} = k(h(x), h_Q) y_Q \tag{5.8}$$

where α, k and $h(x)$ are a smoothing factor, a kernel similarity function and the feature representation of the input x. The pseudo-labels $\hat{y}^{sim}$ introduced in Equation (5.8) closely reflect the data distribution. However, biases present in the data, if not addressed, could be amplified due to the unweighted kernel average (*e.g.* by favoring the majority classes *cf.* Figure 5.3b over the minority classes). In this work, we address the short-comings of previous approaches by introducing a normalization term, computed leveraging Gaussian Processes, that automatically counteracts class imbalances in the data.

5.2.2 Gaussian Processes-based Label Refinement

We introduce SemiGPC, our label refinement strategy based on Gaussian Processes (GPs). Our key motivation is that the normalized kernel similarity used in the GP posterior mean helps address the class imbalance by equalizing the local contribution of each sample in the buffer for each class population. Similarly to previous methods [133, 84, 167], SemiGPC aggregates global information for the refinement of the input location of each pseudo-label. However, SemiGPC retains local sensitivity by favoring minority classes when appropriate, despite the global aggregate favoring majority classes, as shown in Figure 5.1.

Given a memory buffer containing features and labels (h_Q, y_Q), we define SemiGPC refined pseudo-labels as

$$\hat{y}^{GP} = \lambda \mu^{GP}(h(x)) = \lambda k(h(x), h_Q) K^{-1} y_Q \tag{5.9}$$

$$\text{with } K = k(h_Q, h_Q) + \sigma^2 I \tag{5.10}$$

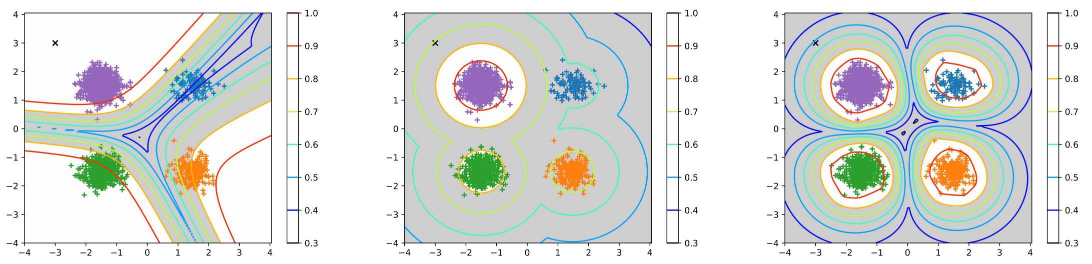
where μ^{GP} is the posterior mean of the GP, λ is the logit scaling factor, σ a regularization parameter of the GP which represents how much we trust labels in the memory bank and k is the GP kernel function (*e.g.* the RBF kernel). By comparing Eqs. (5.8) and (5.9) we see that the GP approach aggregates all labels in the memory bank and reweighs them according to the inverse covariance matrix K^{-1}. Such a normalization is particularly useful to counteract class imbalance, as we show in Figure 5.3. In the following, we use the RBF kernel defined as

$$k(x, y) = \eta \exp\left(-\frac{\|x - y\|^2}{2l^2}\right) \tag{5.11}$$

where η and l are the kernel scale factor and length scale respectively. Note that Equation (5.9) characterizes the posterior mean of a GP whose likelihood function is Gaussian. Other non-Gaussian options are available and are typically applied to build GP based classifiers [121]. However, when non-Gaussian likelihoods are used, there is no closed-form solution and approximation schemes that entail higher computational costs are required [121]. Thus, we choose to refine pseudo-labels by directly regressing the logits $\hat{y}$ using a Gaussian likelihood. Connection with other label-refinement methods. Equation (5.9) can also be rewritten as

$$\mu^{GP}(h(x)) = k(h(x), h_Q) y_K, \tag{5.12}$$

a similarity-based aggregation of y_K, the propagated version of y_Q through the graph defined by K. When $\eta/\sigma^2 \rightarrow 0, K \rightarrow \sigma^2 I$. In this setting, Equation (5.9) becomes equivalent to Equation (5.8). Thus, we obtain the similarity-based aggregation strategy of works such as SimMatch [167] and CoMatch [84]. Furthermore, clipping the kernel below a given threshold results in a matrix K equivalent to an epsilon graph.



(a) Linear classifier confidence

(b) Similarity-based classifier confidence

(c) Gaussian Processes-based classifier confidence

Figure 5.3: **Comparison of confidence maps**: (a) a linear model, (b) a similarity-based classifier [84] and (c) a GP classifier are represented using the contour lines. The number of samples per class grows clockwise by a factor of 2 starting from the top right cluster. We gray out regions that are below 80% confidence. The outlier at (-3,3) is indicated with an $\times$. (c) Only the GP classifier can define confidence levels that are not biased toward the majority classes and ignore the outlier.

SemiGPC robustness to class imbalance. We illustrate with a toy example how SemiGPC is more robust to class imbalance than previous methods. In particular, we compare SemiGPC with a linear classifier and the similarity-based classifier used in CoMatch [84] in Figure 5.3. We build a dataset with four normally distributed classes centered at $(1, 1)$, $(1, -1)$, $(-1, -1)$, and $(-1, 1)$, respectively, and plot the model confidence as defined in Equation (5.5). To simulate class imbalance, the number of samples per class grows by a factor of 2 starting from the top right cluster and going clockwise. First, note how samples far from the data distribution, *e.g.* $(-3, 3)$, are assigned very high confidence by the linear model, although such points are isolated from the others and therefore should not be considered well supported by evidence. Second, note that the minority class (in blue) is a low-confidence region for both the linear and similarity-based classifiers, despite locally containing many samples supporting that class. On the other hand, the GP-based classifier defines an appropriate high-confidence region for each supported class, and its sensitivity to the confidence threshold is much smaller than the similarity-based classifier used in CoMatch [84] as highlighted by the contour plots. In summary, thanks to the use of a GP-based label refinement strategy, SemiGPC is confident if: (1) the considered sample is close to a subset of h_Q regardless of whether it belongs to the majority or minority classes, since K^{-1} reweighs the kernel similarity to counteract disparities in class populations while all samples far from the data are considered outliers, and (2) the sample is located in a high purity region w.r.t. y_Q since the averaged conflicting result in a low model confidence that is spread between classes.

5.2.3 Efficient GP update

As we mentioned in Section 5.2.2, applying GPs in a classification setting requires approximation schemes that are, in general, computationally expensive. To reduce the forward time of SemiGPC we choose to model the refined pseudo-labels using a Gaussian likelihood. In this way, computing the posterior mean for each input image only requires solving a quadratic optimization problem available in closed form. However, computing μ^{GP} is still computationally expensive, since we need to update the set h_Q after each update of the model and invert the covariance matrix K that scales with the cube of the memory bank size (N_Q) at each mini-batch forward pass. To speed up computations, we start from the key observation that at each optimization iteration, most of the samples in the queue do not change. Therefore, at the t-th iteration after observing the new batch of data of size B, we update the previously computed covariance at step $t-1$ with an incremental update rule. In the following, we implement SemiGPC using the well-known matrix inversion lemma [12] (Woodbury identity), which provides a simple batched iterative rank-B correction to the inverse of a given invertible matrix.

In particular, for each labeled training mini-batch of size B, we replace the B oldest samples in the buffer with the features computed using the current mini-batch. Let K_{t-1} and K_t be the covariance matrices in iterations $t-1$ and t, respectively. We now show how to compute K_t by updating K_{t-1} after having updated the memory bank with the new samples from the current mini-batch. We write

$$K_t = \begin{bmatrix} k(h_o, h_o) & k(h_o, h_n) \\ k(h_n, h_o) & k(h_n, h_n) \end{bmatrix} = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} \tag{5.13}$$

where h_o and h_n denote the old samples that were kept in the buffer and the new samples added to the buffer. is the identity matrix. The inverse of K_t is given by

$$K_t^{-1} = \begin{bmatrix} K_{11} & K_{12} \\ K_{12}^\top & K_{22} \end{bmatrix} \tag{5.14}$$

$$\text{with } K_{22} = (D^{-1}C^\top A^{-1}C)^{-1} \tag{5.15}$$

$$K_{11} = A^{-1} + A^{-1}CK_{22}C^\top A^{-1} \tag{5.16}$$

$$K_{12} = -A^{-1}CK_{22}. \tag{5.17}$$

Note that computing K_t^{-1} only requires inverting the two matrices $(D^{-1}C^\top A^{-1}C)$ and A. The former is of size $B \times B$, while the latter is still a relatively large matrix of size $(N_Q - B) \times (N_Q - B)$. However, A does not depend on the newly added samples, and its inverse A^{-1} can be computed efficiently only by requiring the inverse of a $B \times B$ matrix as follows:

$$A^{-1} = M_{11} - M_{12}M_{22}^{-1}M_{21} \tag{5.18}$$

$$\text{where } K_{t-1}^{-1} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \tag{5.19}$$

Proof. In the previous time step $t-1$, the kernel matrix is given by

$$\begin{aligned} K_{t-1} &= k([h_o, h_d], [h_o, h_d]) + \sigma^2 I \\ &= \begin{bmatrix} A & C_d \\ C_d^T & D_d \end{bmatrix} \end{aligned} \tag{5.20}$$

where h_d represents the samples that were deleted when h_n was added to the buffer. The inverse of the kernel matrix can be expressed as

$$K_{t-1}^{-1} = \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{bmatrix} \tag{5.21}$$

The inverse of matrix A can be derived as follows:

$$I = K_{t-1}K_{t-1}^{-1} \tag{5.22}$$

$$\Rightarrow AM_{11} + C_d M_{12} = I \tag{5.23}$$

$$\text{and} \quad AM_{21} + C_d M_{22} = 0 \tag{5.24}$$

$$\Rightarrow C_d = -AM_{21}M_{22}^{-1} \tag{5.25}$$

$$\Rightarrow A(M_{11} - M_{21}M_{22}^{-1}M_{12}) = I \tag{5.26}$$

$$\Rightarrow A^{-1} = M_{11} - M_{21}M_{22}^{-1}M_{12} \tag{5.27}$$

□

For simplicity, we assume that the new samples are located at the end of the buffer; however, the derivation remains true for an arbitrarily ordered buffer up to a permutation matrix.

Table 5.1: **Complexity Comparison of GP updates**. We observe a $\times 7.5$ speedup in practice.

Classic GP update	Efficient GP update
$O(N^3Q + BN^2Q)$	$O(B^3 + BN^2Q + B^2NQ)$

In summary, using block matrix linear algebra, the cost of inverting K_t can be reduced to computing the inverse of a couple of $B \times B$ matrices, which is in turn much more efficient when $B << N_Q$ as is the case in our setting. For example, for a buffer size $N_Q \sim 16k$ and a batch size $B = 8$, using our efficient update rule results in $\times 7.5$ speedup. The exact comparison of complexity is provided in Table 5.1.

Class-balanced SemiGPC. SemiGPC has the additional benefit of allowing us to explicitly address class imbalance without altering the training scheme. We split h_Q into C class buffers and insert the new samples based on their labels, thus ensuring a balanced h_Q. We compare this approach to the classic class rebalancing in the supplementary material.

5.3 Experimental Settings

5.3.1 Implementation details

We use the semi-supervised training recipe of USB [150][1]. It uses an ImageNet [26] pre-trained ViT to initialize the student model. This training scheme allows for faster training time and better overall performance. We use a ViT Small/Tiny with a patch size of 2 and a resolution of 32 for CIFAR100/10, respectively. For our other experiments, we use a ViT Small with a patch size of 16 and a resolution of 224. All our experiments can be run on a single V100 GPU. All our models are trained using AdamW [91] for 200 epochs using a batch size of 8. The detailed set of hyperparameters is provided in Table 5.2. For most of our experiments, we use SimMatch as our baseline. We include a comparison of SemiGPC across different algorithms in Section 5.5.1. For all SemiGPC experiments, we use a buffer size N_Q=16300. Following most works in the literature, we adopt the Top 1 Accuracy as our main evaluation metric and report the mean and standard deviation across 3 random seeds.

5.3.2 Datasets

CIFAR10-LT, CIFAR100-LT. We evaluate SemiGPC on imbalanced versions of CIFAR10 and CIFAR100. The class distribution of these datasets can be fully described using the imbalance ratio γ and the number of samples in the majority class N_1 . For each class $1 < i \leq K$ its number of samples N_i is defined as

$$N_i = N_1 \gamma^{\frac{i-1}{K-1}}, \gamma = \frac{N_1}{N_K} \tag{5.28}$$

where γ, N_K, and K are the imbalance ratio, the cardinality of the minority class and the number of classes respectively.

FGVC Benchmarks. We also evaluate SemiGPC on the fine-grained semi-supervised benchmarks introduced in [139, 140]. These challenging benchmarks contain naturally long-tailed distributions with highly similar class pair. Note that both works [139, 140] argue that semi-supervised methods struggle on such benchmarks. These datasets include a labeled set L_{in} and two unlabeled $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ with seen and unseen classes.

[1] https://github.com/microsoft/Semi-supervised-learning

Table 5.2: **SemiGPC training hyper-parameters**. SemiFGVC refers to the Semi-Aves, SemiCUB, SemiFungi and Semi-iNat benchmarks.

	CIFAR	**SemiFGVC**
Backbone (ViT)	Small (C100) / Tiny (C10)	Small
Patch Size	2	16
Resolution	32	224
N_Q	16300	16300
length scale l	10	20
noise variance σ	10	1
scale factor η	1	1
labeled batch size	8	8
unlabeled batch size	8	8
learning rate	$5e-4$	$1e-3$
layer decay	0.5	0.65
epochs	200	200
steps per epoch	1024	1024
warmup epochs	5	5
temperature T	0.1	0.2
logits scale λ	15	15
confidence threshold τ	0.95	0.95

SemiAves. This dataset [137] is built using the Aves kingdom in iNaturalist 2018 dataset [2]. L_{in}, $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ include 200/200/800 species and 5959/26640/122208 images, respectively. The test set is balanced and contains 40 samples per class. Its reported imbalance ratio is $\gamma = 7.9$.

SemiFungi. This dataset is based on the CVPR 2018 FGVCx Fungi challenge dataset [1]. L_{in}, $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ include 200/200/1194 species and 4141/13166/64871 images, respectively. The test set is balanced and contains 20 samples per class. Its reported imbalance ratio is $\gamma = 10.1$.

Semi-iNat. This dataset was introduced at the CVPR 2021 FGVC8 workshop [138]. L_{in}, $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ include 810/810/1629 species and 13771/91336/221912 images, respectively. The test set is balanced and contains 100 samples per seen class. Its imbalance ratio is $\gamma = 8.5$.

SemiCUB. This dataset is based on the Caltech-UCSD Birds-200-2011 (CUB) dataset [145]. L_{in}, $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ include 100/100/100 species and 1000/3853/5903 images, respectively. Unlike the other three, only the unlabeled sets are imbalanced with $\gamma \sim 2$ for $\mathcal{U}_{in}$. The test set is balanced and contains 1000 samples. We use $U = \mathcal{U}_{in}$ as our unlabeled dataset.

5.4 Experimental Results

In this section, we present the robustness of SemiGPC under various degrees of class imbalance in different data regimes on CIFAR10-LT and CIFAR100-LT. We then report the performance on the more challenging long-tailed semi-supervised benchmarks SemiAves, SemiCUB, SemiFungi, and Semi-iNat. Lastly, we benchmark SemiGPC on the classic balanced semi-supervised splits of CIFAR10 and CIFAR100. For all our imbalanced experiments, we forgo using techniques such as CReST [153], as they do not necessarily improve performance when combined with the USB [150] training recipe. These results can be found in the in Section 5.5.3.

5.4.1 Imbalanced Semi-Supervised Learning

In this section, we evaluate the robustness of SemiGPC under different degrees of class imbalance on CIFAR10-LT and CIFAR100-LT. More specifically, we explore two imbalanced settings based on whether one has access to a balanced labeled dataset or not:

Setting A $\gamma_l = \gamma_u > 1$ Both the labeled and unlabeled sets are imbalanced using the same factor. Following prior works, we use $(N_1^l, N_1^u) = (150, 500)$ for the imbalanced version CIFAR100, *i.e.* CIFAR100-LT. N_1^l and N_1^u are the number of samples for the majority class in the labeled and unlabeled datasets, respectively.

Setting B $\gamma_l = 1; \gamma_u > 1$ Only the unlabeled set is imbalanced. For the labeled setting, we use 4 samples per class resulting in $\times 100 / \times 10$ fewer labeled samples compared to A for CIFAR10-LT and CIFAR100-LT respectively. We argue that this setting is more challenging and better represents real-world scenarios. Indeed, realistically, a small set of balanced labeled samples can be curated, but one cannot make any assumptions on the distribution of the unlabeled dataset based on its labeled counterpart.

CIFAR100-LT (Table 5.3). For setting A, we use the class-balanced version of SemiGPC. We also include the numbers reported by [36] for CoSSL+ReMixMatch as a reference since they represent the current state of the art. For setting A, we observe that SemiGPC outperforms the baseline for all values of γ_u. This highlights the

Table 5.3: **Top1 Accuracy obtained on CIFAR100-LT**. We compared models for different values of γ_l and γ_u. †: A class balanced buffer is used for SemiGPC. *: as reported by [36]. The difference to the baseline is highlighted in green/red.

Model	γ_l	γ_u	n_{lb}	Top1 Acc
CoSSL [36]*	20	20	4741	55.80±0.62
SimMatch	20	20	4741	83.38±0.48
w/ SemiGPC †	20	20	4741	**83.76**±0.26(**+0.37**)
CoSSL [36]*	50	50	3751	48.90±0.61
SimMatch	50	50	3751	78.82±0.60
w/ SemiGPC †	50	50	3751	**79.79**±0.08(**+0.97**)
CoSSL [36]*	100	100	3218	44.10±0.59
SimMatch	100	100	3218	73.90±0.75
w/ SemiGPC †	100	100	3218	**74.48**±0.98(**+0.58**)
SimMatch	1	20	400	76.28
w/ SemiGPC	1	20	400	**77.799**±0.51(**+1.53**)
SimMatch	1	50	400	72.78±0.29
w/ SemiGPC	1	50	400	**75.21**±0.53(**+2.43**)
SimMatch	1	100	400	70.19±0.43
w/ SemiGPC	1	100	400	**73.47**±0.63(**+3.28**)

robustness of SemiGPC w.r.t. class imbalance and its inherent ability to address it explicitly using a balanced buffer. The results obtained for setting B further support the robustness of SemiGPC w.r.t. class imbalance. Indeed, when provided with balanced samples that are 10× fewer than setting A, SemiGPC is able to outperform our baseline for all values of γ_u by a margin greater than +1.5%. Additionally, for each model and value γ_u we measure the gap $\delta(\gamma_u) = Acc(A) - Acc(B)$ between the accuracies $Acc(A)$ and $Acc(B)$ in settings A and B, respectively. When comparing Δ averaged over all γ_u values, we observe a gap of 5.62% and 3.85% for SimMatch and SemiGPC, respectively. In addition to improving performance across both settings, SemiGPC is better at bridging the gap between the two data regimes by approximately 32%.

CIFAR10-LT (Table 5.4). For setting A, we observe that SemiGPC outperforms the baseline for different values of γ_u especially for the more challenging setting $\gamma_u = 150$ where we observe a gap of +1.34%. This highlights the robustness of SemiGPC w.r.t. class imbalance. SemiGPC also largely outperforms our baseline in the setting B. We observe an accuracy increase of at least 7.63% across all values of γ_u with the gap growing bigger for higher values of γ_u up to +10.17%. Additionally, we report an average gap across settings of 17.39% and 9.54% for SimMatch and SemiGPC

Table 5.4: **Top1 Accuracy obtained on CIFAR10-LT**. We compared models for different values of γ_l and γ_u. †: A class balanced buffer is used for SemiGPC. *: as reported by [36]. The difference to the baseline is highlighted in green.

Model	γ_l	γ_u	n_{lb}	Top1 Acc
CoSSL [36]*	50	50	4196	87.70±0.21
SimMatch	50	50	4196	96.48±0.26
w/ SemiGPC †	50	50	4196	**96.80**±0.12(**+0.32**)
CoSSL [36]*	100	100	3720	84.10±0.56
SimMatch	100	100	3720	9±.50.59
w/ SemiGPC †	100	100	3720	**95.74**±0.37(**+1.15**)
CoSSL [36]*	150	150	3496	81.30±0.83
SimMatch	150	150	3496	94.07±1.46
w/ SemiGPC †	150	150	3496	**95.41**±0.56(**+1.34**)
SimMatch	1	50	40	80.59±2.24
w/ SemiGPC	1	50	40	**88.22**±2.38(**+7.63**)
SimMatch	1	100	40	76.69±2.13
w/ SemiGPC	1	100	40	**86.86**±4.48(**+10.17**)
SimMatch	1	150	40	75.68±4.31
w/ SemiGPC	1	150	40	**84.25**±8.92(**+8.57**)

respectively, *i.e.* a relative improvement of 45%. Thanks to its normalization scheme, SemiGPC reduces the risk of confirmation bias which is more prominent when the labeled data is scarce.

5.4.2 Semi-Supervised FGVC Benchmarks

In the section, we evaluate the performance of SemiGPC on the naturally long-tailed semi-supervised benchmarks such as SemiAves, SemiFungi, and SemiCUB and for SemiiNat. In addition to class imbalance, these datasets include highly similar classes.

Seen Classes. We report the results obtained for the seen classes *i.e.* $\mathcal{U} = \mathcal{U}_{in}$ in Table 5.5. For reference, we report the numbers obtained by [140] for SemiAves, Semi-Fungi, and SemiCUB and by [139] for Semi-iNat. We show that using the USB [150] training recipe produces a strong semi-supervised baseline compared to the numbers reported by [139, 140]. Furthermore, SemiGPC outperforms the baseline on all fine-grained benchmarks. This is especially true for the SemiFungi dataset with the highest imbalance ($\gamma = 10.1$) where SemiGPC improves upon the baseline accuracy by +3.49%.

Table 5.5: **Top1 Accuracy obtained on the SemiFGVC benchmarks**. We report the results on the considered fine-grained semi-supervised benchmarks when training with or without unseen classes in the unsupervised set. *: as reported by [139, 140]. The difference to the baseline is highlighted in green.

Dataset	Model	$\mathcal{U}$	Top1 Acc
SemiCUB	FixMatch [140]*	$\mathcal{U}_{in}$	53.20
	SimMatch	$\mathcal{U}_{in}$	84.53 ± 0.45
	w/ SemiGPC	$\mathcal{U}_{in}$	$\mathbf{85.43 \pm 0.67}$**(+0.90)**
SemiAves	FixMatch [140]*	$\mathcal{U}_{in}$	57.40 ± 0.80
	SimMatch	$\mathcal{U}_{in}$	68.47 ± 0.43
	w/ SemiGPC	$\mathcal{U}_{in}$	$\mathbf{69.59 \pm 0.09}$**(+1.12)**
SemiFungi	FixMatch [140]*	$\mathcal{U}_{in}$	56.30 ± 0.50
	SimMatch	$\mathcal{U}_{in}$	68.01 ± 0.19
	w/ SemiGPC	$\mathcal{U}_{in}$	$\mathbf{71.50 \pm 0.49}$**(+3.49)**
Semi-iNat	FixMatch [139]*	$\mathcal{U}_{in}$	44.10
	SimMatch	$\mathcal{U}_{in}$	64.95 ± 0.11
	w/ SemiGPC	$\mathcal{U}_{in}$	$\mathbf{66.54 \pm 0.85}$**(+1.59)**
SemiCUB	FixMatch [140]*	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	52.8
	SimMatch	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	82.60
	w/ SemiGPC	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	**84.20(+1.60)**
SemiAves	FixMatch [140]*	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	49.7
	SimMatch	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	63.32
	w/ SemiGPC	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	**65.03(+1.71)**
SemiFungi	FixMatch [140]*	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	51.20
	SimMatch	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	62.98
	w/ SemiGPC	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	**65.57(+2.59)**
Semi-iNat	FixMatch [139]*	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	38.5
	SimMatch	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	60.90
	w/ SemiGPC	$\mathcal{U}_{in} \cup \mathcal{U}_{out}$	**61.28(+0.38)**

Unseen Classes. The considered fine-grained semi-supervised benchmarks contain two separate unlabeled sets $\mathcal{U}_in$ and $\mathcal{U}_out$ with seen and unseen classes respectively. We reported the semi-supervised performance of models training using $\mathcal{U} = \mathcal{U}_{out} \cup \mathcal{U}_{in}$ as the unlabeled set in Table 5.5. Note that the risk of confirmation bias is higher in this setting compared to the setting where $\mathcal{U} = \mathcal{U}_in$. In particular, a model can learn to assign confident predictions to unlabeled samples from unseen classes, since the labeled set does not contain any samples to disprove such predictions. This effect is reflected by the lower accuracy values for all models. However, SemiGPC consistently outperforms both the baseline and the numbers reported in the literature [139, 140]. Indeed, since the buffer used in SemiGPC only contains samples from the seen classes, SemiGPC limits the impact of the wrong predictions for the unseen classes on the rest of the training.

These results show that SemiGPC is not only more robust w.r.t. class imbalance on artificially skewed benchmarks such as CIFAR10/100-LT but is also better suited for naturally imbalanced datasets containing fine-grained classes where it establishes a new state of the art.

5.4.3 Standard CIFAR10/CIFAR100

Table 5.6: **Top1 Accuracy obtained on CIFAR10 and CIFAR100**. We evaluate the semi-supervised performance for different numbers of labeled samples. $*$: reported by [150, 151].

Dataset	Model	n_{lb}	Top1 Acc
CIFAR100	USB [150]*	200	79.15
CIFAR100	SimMatch	200	79.18
CIFAR100	w/ SemiGPC	200	**80.01(+0.83)**
CIFAR100	USB [150]*	400	83.20
CIFAR100	SimMatch	400	83.25
CIFAR100	w/ SemiGPC	400	**83.87(+0.62)**
CIFAR10	FreeMatch [151]*	40	95.10
CIFAR10	SimMatch	40	**97.32**
CIFAR10	w/ SemiGPC	40	97.14(**−0.18**)
CIFAR10	FreeMatch [151]*	250	95.12
CIFAR10	SimMatch	250	97.21
CIFAR10	w/ SemiGPC	250	**97.39(−0.18)**

Lastly, we evaluate our SemiGPC method on different splits of CIFAR100 and CIFAR10. We report the performance obtained in Table 5.6 when using 200/400 and 40/250 labeled samples for CIFAR100 and CIFAR10, respectively. For reference, we include the numbers reported by USB [150] and FreeMatch [151] as the current state of the art. We observe that SemiGPC improves performance for different amounts of available labeled samples on CIFAR100, with the biggest improvement +0.83% in the low data regime. However, SemiGPC is simply on par with the baseline on CIFAR10. We argue that the semi-supervised performance is already saturated on this benchmark when using the USB training recipe.

Table 5.7: **Impact the chosen semi-supervised method**. We compare the Top1 Accuracy on SemiAves when using different Semi-supervised algorithms.

Model	Dataset	Top1 Acc
FreeMatch	SemiAves	66.97
w/ SemiGPC	SemiAves	**67.93**(**+0.96**)
FixMatch	SemiAves	67.36
w/ SemiGPC	SemiAves	**68.31**(**+0.95**)
ReMixMatch	SemiAves	67.9
w/ SemiGPC	SemiAves	**68.31**(**+0.41**)
SimMatch	SemiAves	68.45
w/ SemiGPC	SemiAves	**69.30**(**+0.85**)

5.5 Ablations

In order to establish the general purpose nature of SemiGPC, throughout this section, we highlight the impact of SemiGPC on top of different underlying algorithms and/or pre-training strategies. We also provide empirical results regarding the importance of class rebalancing and why we forgo the use of CReST in our main experiments.

5.5.1 Semi-Supervised Learning Algorithms

The design of SemiGPC is agnostic to the underlying choice of the semi-supervised algorithms. We evaluate the impact of our proposed GP-based classifier on different Semi-Supervised methods including FixMatch [133], ReMixMatch [14], SimMatch [167] and FreeMatch [151] on the SemiAves benchmark. The obtained results are reported in Table 5.7. Despite FreeMatch [151] being designed to better deal with class imbalance, we observe that SimMatch [167] outperforms it when using the USB [150] training recipe. This justifies why we use SimMatch as our baseline throughout this work. SemiGPC not only improves performance across all considered methods, but its performance also improves monotonically with respect to the performance of the base method. This allows SemiGPC to remain relevant w.r.t. future better semi-supervised algorithms.

5.5.2 Pre-training Strategy

As stated in Section 5.3.1, we used a pre-trained ViT [32] to initialize our semi-supervised models. We evaluate the impact of SemiGPC across different pre-training strategies by training SimMatch on the SemiAves benchmark using supervised pre-training, Dino [20] and MSN [6] pre-training on ImageNet [26]. Both Dino [20] and

Table 5.8: **Pretraining strategy**. Comparison of the Top1 Accuracy on SemiAves when using different pre-training strategies.

Model	Pretraining	Top1 Acc
SimMatch	DINO	64
w/ SemiGPC	DINO	**65.32(+1.32)**
SimMatch	MSN	64.7
w/ SemiGPC	MSN	**67.73(+3.03)**
SimMatch	Supervised	68.45
w/ SemiGPC	Supervised	**69.30(+0.85)**

MSN [6] are self-supervised methods that produce competitive performance on ImageNet, with MSN being the top performer out of the two. We report the obtained results in Table 5.8. Not only does the SemiGPC performance scale based on the performance of the pre-training methods, it also improves performance across all considered pre-training strategies.

5.5.3 CReST experiments

Table 5.9: **Top1 Accuracy when using CReST** [153]. We report the performance of SimMatch and SemiGPC after the second phase of the CReST training.

Dataset	γ	n_{lb}	**SimMatch**	**SemiGPC**
CIFAR10	50	4196	96.77	**96.96**
CIFAR10	100	3720	95.27	**96.23**
CIFAR10	150	3496	94.82	**96.05**
CIFAR100	20	4741	83.61	**83.63**
CIFAR100	50	3751	77.25	**78.54**
CIFAR100	100	3218	72.22	**73.64**
SemiCUB	~ 2	1000	83.4	**85.7**
SemiFungi	10.1	4141	68.02	**70.75**
Semi-iNat	8.5	13771	66.53	**68.77**

We explore the effect of combining SemiGPC with CReST [153], a semi-supervised technique designed for imbalanced classification. In CReST, the semi-supervised training consists of two phases. In the first phase, the model is trained normally, *e.g.* using SimMatch. Then, the labeled set is updated to include pseudo-labeled unlabeled samples such that each class $1 \leq i \leq C$ is sampled at a rate of

$$\beta_i = \left(\frac{N(C+1-i)}{N_1}\right)^{\alpha} \tag{5.29}$$

where α, N_1 and N_j are the tuning scaler and the number of labeled samples for the majority and j-th classes respectively. The model is then re-trained during the second phase using the updated labeled set, which is designed to be more balanced. We did not observe any improvement when using CReST with the USB [150] training scheme for almost all settings. In Table 5.9, we observe that SemiGPC+CReST consistently outperforms SimMatch+CReST. We argue that SemiGPC is better at handling the label noise introduced in the second phase of the training thanks to its normalized buffer aggregation. Indeed, noisy labels can only contribute to the SemiGPC predictions if they are corroborated by the other samples in the buffer.

5.5.4 Class Imbalance

We compare our SemiGPC using a class-balanced memory buffer to the SimMatch baseline trained using a class-balanced labeled set obtained by weighted re-sampling on CIFAR100-LT with $\gamma_l = \gamma_u = 100$. We report the obtained results in Table 5.10.

Table 5.10: **Comparison of class balancing strategies**. We compare different balancing strategies on CIFAR100-LT with $\gamma_l = \gamma_u = 100$. † : using a balanced memory buffer.

	Labeled Set	Top1 Acc
SimMatch	imbalanced	74.00
SimMatch	balanced using resampling	74.51
w/ SemiGPC †	imbalanced	**75.36**

The model using SemiGPC performs best. When using weight resampling, some majority class samples are randomly discarded in favor of over-sampled minority class samples, which in turn affects the training dynamics. Instead, balancing the memory buffer used by SemiGPC can be done independently of the sampler used during training. Furthermore, the noise term $+\sigma^2 I$ in Equation (5.9) allows SemiGPC to handle the presence of near duplicates in the memory buffer, which are otherwise a source of potential overfitting.

5.6 Discussions

Our method SemiGPC is able to achieve state-of-the-art results across different benchmarks and settings thanks to its ability to counteract imbalances in the data distribution. However, SemiGPC still has a few limitations. We observe in Tables 3 and 5 that SemiGPC shows mixed results when used on top of an already strong baseline (¿ 94% accuracy) such as on CIFAR10. Also, although our update rule greatly speeds up the matrix inversion, it does not fully eliminate the additional computational overhead.

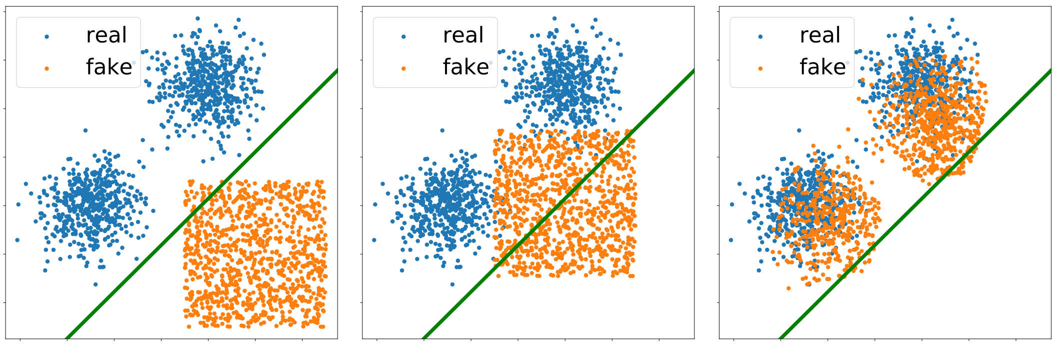
Furthermore, the quadratic scaling of the memory cost of this matrix limits the maximum buffer size in SemiGPC to around $N_Q = 16K$. However, this limitation can be addressed using an ensemble of GPs, each using a separate buffer. This would allow us to scale SemiGPC to $N_Q \sim 80K$. Furthermore, our update rule is not compatible with using trainable kernel hyperparameters since it relies on reusing previous values of the kernel matrix. Leveraging matrix-vector-matrix solvers [147] fixes both these limitations. Indeed, by enabling efficient GP inference with trainable hyperparameters, SemiGPC would forgo sharing the kernel hyperparameters across classes and adapt the geometry induced by the kernel function on a per-class basis. We leave deriving an online update rule using matrix-vector-matrix solvers for future work. Lastly, combining SemiGPC with alternative definitions of confidence to Equation (5.5) by either using the sample-wise posterior covariance provided by GP or by leveraging recent advances in efficient Neural Tangent Kernel (NTK) computation [106, 168] remains an open area of research.

Chapter 6

Generative Adversarial Learning via Kernel Density Discrimination

We introduce Kernel Density Discrimination GAN (KDD GAN), a novel method for generative adversarial learning. KDD GAN formulates the training as a likelihood ratio optimization problem where the data distributions are written explicitly via (local) Kernel Density Estimates (KDE). This is inspired by recent progress in contrastive learning and its relation to KDE. In our approach, features are no longer optimized for linear separability, as in the original GAN formulation, but for the more general discrimination of distributions in the feature space. Moreover, we formally prove that KDD GAN is guaranteed to converge to the real data distribution. We also analyze the gradient of our loss with respect to the feature representation and show that it is better behaved than that of the original hinge loss. We perform experiments with the proposed KDE-based loss, used either as a training loss or as a regularization term, on both CIFAR10 and scaled versions of ImageNet. We use BigGAN/SA-GAN as a backbone and baseline, since our focus is not to design the architecture of the networks. We show a boost in the quality of the generated samples with respect to FID from 10% to 40% compared to the baseline.

Generative learning finds applications in many computer vision applications such as image translation [56, 169, 34, 112], image processing [78, 74], image restoration [144, 110, 163], text-to-image mapping [122, 160, 81, 119] and, more generally, defining image priors in image-based optimization problems [144, 95]. Generative models based



(a) Initial state (b) Hinge loss (c) KDD loss

Figure 6.1: **Illustration of the difference between the hinge loss and KDD loss during the generator update.** The blue and orange point clouds represent the discriminator features of the real and fake samples. The initial positions of the samples are shown in Fig. 6.1a. The green line in all three sub-figures represents the decision boundary associated with the optimal linear classifier separating the two distributions at the initial state. Fig. 6.1b and Fig. 6.1c show the updated positions of the fake samples using the Hinge loss and KDD loss respectively. The generator update via the KDD loss leads to a more detailed overlap.

on adversarial learning have been widely successful thanks to several breakthroughs in the design of the generator and discriminator architectures [17, 161, 63], of the loss functions [4, 62, 159] and regularization methods [94, 99, 162, 62]. However, the training of generative models is not straightforward and can still be prone to mode collapse [134, 158, 87] or the inability to capture long-range statistics in the data, leading to visible artifacts [161, 86].

One key assumption in the basic formulation of adversarial learning of [44] is that the generator network should compete with an optimal discriminator, that is, a classifier that can separate real from generated data if any of their statistics does not match. Thus, the general wisdom is that the more powerful the discriminator is, the better the generator trains. Given that training models with contrastive losses yields better performance than training with cross-entropy losses [67], and that contrastive learning can be seen as introducing Kernel Density Estimate (KDE) approximations of the data distribution [148], we propose to train the discriminator and generator models through a KDE approximation of the likelihood ratio loss. Moreover, this approach ensures that the loss defines a valid statistical divergence between the real and generated data distributions at all times. In contrast, the loss used to train state-of-the-art generative adversarial networks corresponds to a known statistical divergence between distributions of real and fake data only when at the saddle point of the Min-Max game.

Our analysis shows that the gradients of the proposed loss are better behaved than those of the hinge loss (defined, for example, by [98]). We propose a KDE defined directly in feature space, so that non-invertible features are allowed. Our method includes a much broader set of discriminator solutions than in the binary classification task of the original GAN formulation. In fact, in the KDE approach, the features are no longer optimized for linear separability, but for the more general discrimination of distributions in the feature space. This can be seen clearly in Fig. 6.1 for 2D point clouds. We call our method *Kernel Density Discrimination GAN* (KDD GAN).

Contributions :

- A theoretical proof that KDD GAN converges to the unique equilibrium point, where the distribution of generated samples matches that of real data;
- KDD GAN outperforms BigGAN [17] (which we use as a backbone) on CIFAR10 [73] and Tiny ImageNet [76] by more than 10% in the FID and IS metrics;
- The proposed KDD loss is flexible and when combined with other methods as a regularizer improves the training in terms of FID and IS on CIFAR10, Tiny ImageNet, and ImageNet 64×64, which has images scaled to 64×64 pixels (derived from [26]);
- The implementation of KDD GAN is on par with conventional hinge loss training [98] in terms of the computational load and the memory footprint.

6.1 Related works

In Section 2.5 we introduce Generative Adversarial Networks as well as other relevant works in generative modeling literature. Of particular interest for our work is Instance Selection [29] that allows us to train GANs more efficiently.

Similar to our KDD GAN, other kernel-based GANs have been previously proposed. [131] explore the idea of using a nonparametric estimate of the Jensen-Shanon Divergence and use KDEs for the purpose of training GANs. This idea is very similar to the one explored in this work. The main differences are that Kernel GAN [131] computes its KDEs in the image space and for a simpler selection of datasets; it also requires an additional autoencoding constraint and computing the KDEs in the feature space for more complex datasets. Alternatively, MMD-GAN [82] and its variants such as Wang et al. [149] explore the idea of matching the two distributions at hand by optimizing the Maximum Mean Discrepancy defined by the chosen kernel. Although the improved MMD introduced in [149] bares a few similarities to our work in terms of having both attractive and repulsive loss terms, the two frameworks are fundamentally different. Our KDD-GAN aims at matching the two distributions in the feature space defined by the discriminator, while MMD-GAN and its variants aim at minimizing the maximum mean discrepancy in the RKHS defined by the kernel choice.

6.2 Kernel Density Discrimination

Let $S_r = \{x_r^{(1)}, \ldots, x_r^{(m)}\}$ be a dataset of m image samples $x_r^{(i)} \in \mathbb{R}^d$, which we call *real data*. They are the instances of a probability density function (pdf) p_r, which we call the *real data* pdf. We aim to build a generative model that maps zero-mean Gaussian samples to images and such that they also follow the real data distribution. To distinguish real from generated samples, we denote the dataset of generated data by S_g, a generated image sample by x_g, and the *generated data* pdf by p_g.

We build our generative model through adversarial learning as in the pioneering work of [44], and thus work with a *discriminator* network D and a *generator* network G. Then, generative adversarial learning can be cast as the following bilevel optimization problem

$$\min_G \; \mathcal{L}_G(D^*, G) \tag{6.1}$$

$$\text{s.t.} \quad D^* = \arg\min_D \mathcal{L}_D(D, G), \tag{6.2}$$

where the optimization in G and D is implemented as the optimization with respect to the parameters of the neural networks implementing them. In the case of hinge loss optimization (see *e.g.* , [98]), the losses in Equation (6.1) are defined as

$$\begin{aligned}\mathcal{L}_D^{\text{Hinge}}(D, G) = & \frac{1}{|S_g|} \sum_{x_g \in S_g} \max\{0, 1 + D(x_g)\} \\ & + \frac{1}{|S_r|} \sum_{x_r \in S_r} \max\{0, 1 - D(x_r)\} \end{aligned} \tag{6.3}$$

$$\mathcal{L}_G^{\text{Hinge}}(D^*, G) = \frac{1}{|S_g|} \sum_{x_g \in S_g} -D^*(x_g), \tag{6.4}$$

which rely on the assumption that the discriminator takes the form of

$$D^*(x) = \log p_r(x) - \log p_g(x). \tag{6.5}$$

In our approach, we would rather explicitly approximate the form $\log \frac{p_r(x)}{p_g(x)}$. The main advantage of having this form is that it is a well-defined divergence between distributions. Thus, it defines a valid gradient for the generator at all times, up to the errors resulting from the chosen approximation.

We propose approximating $p_r(x)$ and $p_g(x)$ in the definition of $D^*(x)$ with Kernel Density Estimates (KDE). The kernels are defined in feature space, and the feature mappings are estimated during training. A simple way to ensure that at the convergence of the bi-level optimization (*i.e.* , when the minima have been reached) the real and fake pdfs match, is to require the invertibility of the feature mappings. Invertibility is the same requirement of Normalizing Flows (*e.g.* [71]) and thus one would have to follow similar restrictions in the neural architectures used to compute the features. However, training invertible neural networks is not easy and, as we argue here below, also not necessary. To simplify the training of the generative model, we instead propose to use KDEs in the feature space $\phi : \mathbb{R}^d \to \mathbb{R}^K$ defined by the last layer of the discriminator , and to allow the feature mapping to be non-invertible. Thus, we aim to match the push-forward measures $\phi_* p_r$ and $\phi_* p_g$, which we denote by $\hat{p}_r^\phi$ and $\hat{p}_g^\phi$, respectively.

We write the losses for KDD GAN explicitly as

$$\mathcal{L}_D^{KDD}(\phi, G) = \frac{1}{|S_g|} \sum_{x_g \in S_g} \max\left\{0, 1 + \log \frac{\hat{p}_r^\phi(x_g)}{\hat{p}_g^\phi(x_g)}\right\} + \frac{1}{|S_r|} \sum_{x_r \in S_r} \max\left\{0, 1 - \log \frac{\hat{p}_r^\phi(x_r)}{\hat{p}_g^\phi(x_r)}\right\} \tag{6.6}$$

$$\mathcal{L}_G^{KDD}(\phi^*, G) = \frac{1}{|S_g|} \sum_{x_g \in S_g} -\log \frac{\hat{p}_r^{\phi^*}(x_g)}{\hat{p}_g^{\phi^*}(x_g)}, \tag{6.7}$$

by approximating the push-forward measures of the pdfs p_r and p_g via the following KDEs in feature space

$$\hat{p}_r^\phi(\xi) = \frac{1}{|S_r|} \sum_{x_r \in S_r} \mathcal{K}_\tau(\phi(x_r), \xi), \tag{6.8}$$

$$\hat{p}_g^\phi(\xi) = \frac{1}{|S_g|} \sum_{x_g \in S_g} \mathcal{K}_\tau(\phi(x_g), \xi) \tag{6.9}$$

where

$$\mathcal{K}_\tau(\phi(x), \xi) = \frac{1}{Z} e^{\frac{\langle \phi(x), \xi \rangle}{\tau}} \tag{6.10}$$

is a positive kernel that integrates to 1 in ξ, $\tau > 0$ is a temperature parameter that relates to the spread of each kernel, $|S|$ is the cardinality of S, and Z is the normalization constant (this becomes irrelevant as it cancels out in the ratios in $\mathcal{L}_D(\phi, G)$ and $\mathcal{L}_G(\phi^*, G)$). The features $\phi(x)$ are L^2-normalized through the projection on the unit hypersphere, *i.e.* $|\phi(x)|_2 = 1$. Essentially, we assume that the features are samples of a mixture of von Mises-Fisher (vMF) distributions, where all concentration parameters are equal to $1/\tau$.

As mentioned above, the convergence of KDD GAN does not need the invertibility of the feature mapping ϕ. We show this result formally in Theorem 1 and address the invertibility in Lemma 1.

Lemma 1. *Let p_r and p_g be two distributions over $\mathbb{R}^d$. Given a positive integer K, we have $p_r = p_g \Leftrightarrow \forall\, \phi : \mathbb{R}^d \rightarrow \mathbb{R}^K, \hat{p}_r^\phi = \hat{p}_g^\phi$.*

Proof of Lemma 1. $p_r = p_g \Rightarrow \forall \phi : \mathbb{R}^d \rightarrow \mathbb{R}^K, \hat{p}_r^\phi = \hat{p}_g^\phi$ is trivial since $\{\phi(x), x \sim p_r\}$ and $\{\phi(x), x \sim p_g\}$ are the same set when $p_r = p_g$.

Assume $p_r \neq p_g$. Then, there exists an optimal binary classifier c, whose accuracy is above chance level, *i.e.* , $P(\{c(x) = 1, x \sim p_r\}) > \frac{1}{2}$. We can define a mapping $\phi(x) := c(x)\mathbb{1}_K$ where $\mathbb{1}_K$ is the vector of ones in $\mathbb{R}^K$. In this case, we obtain $E_{x\sim p_r}[\phi(x)^T \mathbb{1}_K] = E_{x\sim p_r}[c(x)]K > \frac{K}{2}$ and $E_{x\sim p_g}[\phi(x)^T \mathbb{1}_K] = E_{x\sim p_g}[c(x)]K < \frac{K}{2}$. This implies that the first moments of $\hat{p}_r^\phi$ and $\hat{p}_g^\phi$ are different, thus $\hat{p}_r^\phi \neq \hat{p}_g^\phi$. Therefore, by contradiction, $\forall\, \phi : \mathbb{R}^d \rightarrow \mathbb{R}^K,\ \hat{p}_r^\phi = \hat{p}_g^\phi \Rightarrow p_r = p_g$ □.

Theorem 1. *$p_g = p_r$ is the unique equilibrium point for KDD GAN.*

Proof of Theorem 1. Let us assume that there exists an equilibrium point (ϕ, G) such that $p_r \neq p_g$. Then we have two cases: $\hat{p}_r^\phi = \hat{p}_g^\phi$ or $\hat{p}_r^\phi \neq \hat{p}_g^\phi$. Assume $\hat{p}_r^\phi = \hat{p}_g^\phi$. Then, according to Lemma 1, there exists a φ such that $\hat{p}_r^\varphi \neq \hat{p}_g^\varphi$; *i.e.* , ϕ is not an equilibrium point of $\mathcal{L}_D^{KDD}$. Now, let us instead assume that $\hat{p}_r^\phi \neq \hat{p}_g^\phi$, then G is not an equilibrium point of $\mathcal{L}_G^{KDD}$ □.

6.2.1 Improving KDE through Data Augmentation

The KDEs in Equation (6.9) are mixtures of von Mises-Fisher distributions centered around a set of *anchor points*. In the KDE approximation, we cannot use the entire dataset S_r as anchor points, because it would be too computationally demanding. Instead, at each iteration of the training procedure, we sample a subset (a minibatch) and use this as anchor points. A fundamental requirement of the KDE approximation is that these sets should be representative of the true distributions p_r or p_g. However, KDE approximations are in general very poor with high-dimensional data, as they require a very large number of anchor points. This is because only the kernels that correspond to anchor points that are “similar” to the input sample dominate in the KDE. However, the likelihood of finding these anchor points through uniform sampling becomes extremely small as we grow in the dimensionality of the data.

One way around this problem is to enrich the set of anchors using data augmentations. Provided that the chosen data augmentation does not produce samples outside the manifold of natural images, this allows us to obtain anchor points that are close enough to give a meaningful KDE.

For similar reasons, we use a *leave one out* KDE, where we remove the anchor point from the set S_r or S_g on which the KDE is evaluated. This avoids a bias towards the unlikely case where we sample exactly a point in the anchor point set. We experimentally show that these KDE implementation details are indeed quite important in boosting the effectiveness of the proposed approach.

6.2.2 Loss Analysis

We analyze the impact of the proposed loss on the generator training and compare it to the case of the standard hinge loss discriminator of [98]. For simplicity, let us consider a discriminator for the standard loss that can be written as the inner product $D_{\text{STN}}(x) = \phi(x)^\top\theta$, for some θ vector (this is updated only when we optimize with respect to the discriminator). In the case of our KDD loss we instead use simply $D_{\text{KDE}}(x) = \phi(x)$. Suppose that the discriminator is now given and we minimize the loss $\mathcal{L}_G$ with respect to the generator G. In the case of a first-order optimization method, we obtain the updates for the generator parameters through the gradients of $\mathcal{L}_G$,

$$\frac{\partial \mathcal{L}_G}{\partial G} = \frac{\partial \mathcal{L}_G}{\partial \phi}\frac{\partial \phi}{\partial G}. \tag{6.11}$$

Since in both the standard hinge loss and our loss the term $\frac{\partial \phi}{\partial G}$ is the same, we can reduce the analysis to the study of $\frac{\partial \mathcal{L}_G}{\partial \phi}$. We obtain:

$$\frac{\partial \mathcal{L}_G^{\text{STN}}}{\partial \phi} = \theta \tag{6.12}$$

and

$$\frac{\partial \mathcal{L}_G^{KDD}}{\partial \phi} = \frac{1}{|S_g|}\sum_{x_g \in S_g} \frac{\partial \log \hat{p}_g^{\phi}(x_g)}{\partial \phi} - \frac{\log \hat{p}_r^{\phi}(x_g)}{\partial \phi}. \tag{6.13}$$

The formulas above show that in the case of the hinge loss the gradient update results in a constant shift, *i.e.* an identical shift for all samples, whereas our KDD loss increases (resp. decreases) the likelihood of x_g under $\hat{p}_g^{\phi}$ (resp. $\hat{p}_r^{\phi}$). An illustration of this effect in 2D is shown in Fig. 6.1.

We also compare our KDD loss to the MMD loss proposed by [149]. Without loss of generality, for a given sample $x \sim p_1$ we compare each term $E_{y\sim p_2}[k(x,y)]$ in their work with its counterpart in ours $\log(E_{y\sim p_2}[k(x,y)])$, where $p_1, p_2 \in \{p_r, p_g\}$ and k is a kernel function. For the vMF kernel, we obtain

$$\text{KDD:} \sum_y \frac{k(x,y)}{\sum_v k(x,v)}\phi(y), \tag{6.14}$$

$$\text{MMD:} \sum_y k(x,y)\phi(y). \tag{6.15}$$

In both cases, the gradient is a weighted average of the samples $\phi(y)$. The key difference is that the Improved MMD loss has a local weighting, *i.e.* it only depends on the current y, and the KDD loss has a global weighting.

Empirical Analysis of the KDD Loss In Figure 6.1, we illustrate the difference between the Hinge and KDD losses already described in Section 3. We consider two point clouds in 2D representing the real and fake push-forward distributions. In this example, the real point cloud is designed to have two Gaussian modes, while the fake one starts off with one uniformly sampled square mode. We first find the optimal linear classifier that separates the two point clouds through gradient descent. The corresponding decision boundary is represented by the green line in Figure 6.1. We then optimize the features of the fake samples with respect to the Hinge loss and the KDD loss. In this example, we forgo feature normalization as its main purpose is to prevent the Discriminator from converging to degenerate solutions where the space collapses. Thus, for visualization purposes, we work with 2D features. In the setting, the vMF kernel is equivalent to a Gaussian kernel with $\sigma = 1$ for the KDE, *i.e.* , $\mathcal{K}(\phi, \xi) \propto \exp -\frac{|\phi-\xi|^2}{2}$.

The minimization of the Hinge loss simply results in translating the fake point cloud without changing its internal structure as shown in Figure 6.1b. In contrast, the KDD loss encourages the fake samples to head towards the closest real mode as shown in Figure 6.1c. For both losses, the optimization was performed using SGD [16] for 200 iterations with a learning rate of 10. 1000 samples were used for both real and fake point clouds. Note that for a frozen Discriminator, updating the Generator using the Hinge loss can result in overshooting the real point cloud, since the translation vector is constant for all subsequent Generator updates. In fact, the optimum is to translate the fake point cloud to infinity. This makes the Generator update with respect to the Hinge loss less well behaved than its KDD counterpart since the latter does not introduce such instability.

6.2.3 Class-Conditioning Extension

We also consider training generative models subject to class conditioning. Let us denote by $y^{(i)}$ the label corresponding to the real image $x_r^{(i)}$. Now, we are interested in the approximation of the quantity $\log \frac{p_r(x,y)}{p_g(x,y)}$, which we can rewrite as

$$\log \frac{p_r(y|x)p_r(x)}{p_g(y|x)p_g(x)} = \log \frac{p_r(y|x)}{p_g(y|x)} + \log \frac{p_r(x)}{p_g(x)}. \tag{6.16}$$

The second term is exactly what we used in $\mathcal{L}_D(\phi, G)$ and $\mathcal{L}_G(\phi^*, G)$. Thus, we can focus on the conditional term $\log \frac{p_r(y|x)}{p_g(y|x)}$. By following [98], we assume the linear form

$$\log \frac{p_r(y|x)}{p_g(y|x)} = y^\top V D(x), \tag{6.17}$$

where V is a (learned) matrix that defines the embedding for the label y.

6.2.4 Regularization of the Feature Mapping

If ϕ maps many samples to the same feature, the discrimination task would become less effective. To avoid this scenario, we encourage φ, the feature mapping before the normalization layer, to be as "responsive" as possible to variations around samples of p_r and p_g by introducing the following additional *Jacobian regularization* term

$$\mathcal{L}_{\text{Jac}} = \frac{1}{|S_r|} \sum_{\substack{x \in S_r \bigcup S_g \\ \Delta x \sim \mathcal{U}(\mathbb{S}^{d-1})}} \left| \frac{|\varphi(x + \delta \Delta x) - \varphi(x)|_2}{\delta} - 1 \right|_1 \tag{6.18}$$

where $\delta > 0$ is a small scalar and Δx is a random unitary direction in image space. φ is defined so that $\phi = \varphi / |\varphi|_2$. This regularization term computes a finite-difference approximation of the gradient of φ with respect to its input and projects it along the random direction Δx. It preserves as much as possible the volume in the feature space, but only for the data in the image distribution. In addition, this regularization term prevents the magnification of the output gradient, which is typically associated with high confidence, and would make the discriminator more susceptible to adversarial inputs. This is a stronger constraint compared to the classic gradient penalty [46], since we implicitly require orthonormality for all rows of the Jacobian, *i.e.* , $\nabla\varphi(x)\nabla\varphi(x)^\top = I_d$.

6.2.5 KDD GAN Formulation

Finally, we can put all the terms together and define the generator and discriminator losses via

$$\mathcal{L}_{G/D} = \gamma \mathcal{L}_{G/D}^{\text{KDD}} + \alpha \mathcal{L}_{G/D}^{\text{Hinge}} + \lambda_\nabla \mathcal{L}_{\text{Jac}}, \tag{6.19}$$

where γ, α and λ_∇ live in $\mathbb{R}^+ \times \{0, 1\} \times \{0, 1e\text{-}5\}$, and where KDD and Hinge refer to our KDD loss and the classic hinge loss used in BigGAN for both the generator and discriminator. Training with the lone hinge loss uses $\alpha = 1, \gamma = 0$; the training with the lone KDD loss uses $\alpha = 0, \gamma = 1$; the setting where $\alpha = 1, \gamma > 0$ is called `Joint` training.

6.3 Implementation

Training Details. We evaluate our models on three different datasets: CIFAR10 [73], Tiny ImageNet and ImageNet 64 × 64. The Tiny ImageNet [76] dataset is a subset of the ILSVRC-2012 ImageNet classification dataset [26] consisting of 200 object classes and 500 training images, 50 validation images and 50 test images per class. Unless otherwise specified, we use $\tau = 1$, $\delta = 1e\text{-}3$, and $\lambda_\nabla = 1e\text{-}5$. Experiments using data augmentations and the Jacobian regularization are denoted with **+DA**

and **+JacD** respectively. All training was run on at most two 2080Ti or one 3090Ti GPUs. Using KDD-GAN results in around 10% longer training times.

Architectures. The architecture used for our CIFAR10 experiments is the same[1] used in the original BigGAN work by [17]. For both Tiny ImageNet and ImageNet 64×64, we use the modified SA-GAN [161] architecture adopted by [29] [2]. We do not use instance selection on CIFAR10 and Tiny ImageNet, as we noticed it hurts performance on smaller datasets. For Instance Selection on ImageNet 64×64, we use a retention ratio of 50%. We choose to train BigGAN/SA-GAN rather than StyleGAN2-ADA for their simpler training scheme and their lesser reliance on regularization terms and implementation tricks. This allows us to isolate the contribution of our KDD loss without requiring a hyperparameter search for the rest of the moving pieces of the training.

Evaluation Metrics. Throughout this paper, we evaluate our generative models using Fréchet Inception Distance (FID) [50], Inception Score (IS) [124], Density and Coverage [101]. These metrics are computed using the original *tensorflow* implementation. As in [29], the real moments used for the FID are computed using the entire dataset and not the filtered one. For FID and IS we use 50k generated samples, for Density and Coverage, we use 10k samples for both distributions and 5 nearest neighbors. Unless specified otherwise, the reported numbers are calculated after 100 k iterations for both CIFAR10 and Tiny ImageNet and after 500k iterations for ImageNet 64×64. The batch size used is 64 for Tiny ImageNet and CIFAR10 and 128 for ImageNet 64×64. The FID moments are computed on the training set for all datasets. We report the performance of the best model obtained during training.

Differentiable Augmentations. We use differentiable random brightness, saturation, contrast, translation, and cut-out data augmentations proposed by [164]. For all our experiments, the loss is computed only on the nonaugmented images. The augmented samples are only used for the Kernel Density Estimation. This is an important distinction from the work by [164].

6.4 Experiments

In this section, we show the quantitative results obtained in CIFAR10, Tiny ImageNet, and ImageNet 64×64. The best and second best values per metric are highlighted and

[1]https://github.com/ajbrock/BigGAN-PyTorch/
[2]https://github.com/uoguelph-mlrg/instance_selection_for_gans/

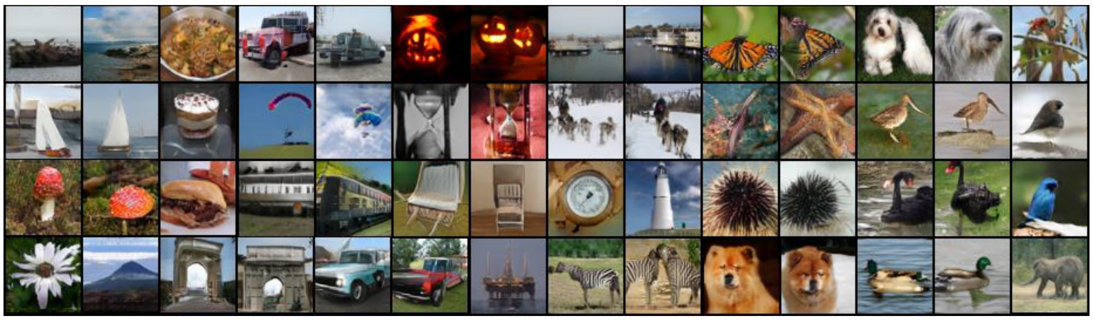


Figure 6.2: **Sample images generated using KDD GAN on ImageNet** 64×64. These sample images were generated using the Joint†model trained on ImageNet 64×64.

underlined, respectively. Generated samples from one of our best models are shown in Figure 6.2. Further qualitative results can be found in the Supplementary Material.

6.4.1 Ablation Results

In Table 6.1, we perform various ablations by training BigGAN [17] on CIFAR10 for 200k iterations each. The three main loss functions used are: the hinge loss [98], the KDD loss, and the joint loss. We study the effects of the parameters associated with the new losses. The first set of experiments studies the effect of the temperature τ used in the KDD loss. We observe that both high and low values of τ are problematic. When comparing $\tau = 0.05$ to $\tau = 5.00$, we observe a trade-off between Image Fidelity (FID) and Diversity (IS). The value of τ determines the level of blurriness of the KDE. Additionally, we explore the effect of the Jacobian regularization. We use a coefficient of $\lambda_\nabla = 1e\text{-}5$. Our KDD GAN using $\tau = 1$ with and without Jacobian regularization outperforms its BigGAN counterpart in both FID and IS. The performance gap is bigger when adding Jacobian regularization.

The second set of experiments examines the effect of γ during the joint training. We observe that all joint models improve on the baseline in terms of IS. This improvement correlates positively with γ, except for $\gamma = 10$ where the IS stagnates. The best joint model ($\gamma = 1$) also outperforms the baseline in terms of FID. This highlights the benefit of using the KDD loss as a regularization term. Lastly, we train our models without the class projection head proposed by [98] and/or without a conditional input for the generator. All models obtained with $\gamma > 0$ in the third block in Table 6.1 outperform the BigGAN baseline in the unconditional setting. This proves that training is not driven solely by the class projection term in the conditional setting. The difference in performance between unconditional KDD model and one that only is missing the projection head can be attributed to the slightly higher number of parameters that

Table 6.1: **KDD GAN Ablations on CIFAR10**. Comparison of the various BigGANs trained on CIFAR10 for different values of τ and γ. **UnCond** refers to the unconditional setting, while **NoProj** refers to removing the class-projection loss term in ProjGAN [98].

Experiments	τ	γ	**FID** ↓	**IS** ↑	**D** ↑	**C** ↑
Hinge	-	-	8.751	8.835	0.966	0.851
KDD	0.05	-	8.753	**9.233**	0.876	0.832
KDD	1.00	-	8.422	<u>9.155</u>	0.868	0.849
KDD	5.00	-	8.604	8.852	**0.970**	0.862
KDD + JacD	1.00	-	**7.237**	9.029	0.932	<u>0.867</u>
Joint	1.00	0.1	9.144	8.767	<u>0.969</u>	0.857
Joint	1.00	0.5	8.795	8.920	0.922	0.855
Joint	1.00	1.0	<u>7.932</u>	9.046	0.968	**0.868**
Joint	1.00	10.0	8.352	9.102	0.930	0.857
KDD + NoProj	0.05	-	13.668	8.274	0.722	0.711
Hinge (Uncond)	-	-	17.782	8.120	0.692	0.686
KDD (Uncond)	0.05	-	15.828	8.326	0.620	0.650
Joint (Uncond)	0.05	1.0	14.394	8.532	0.662	0.712

the latter has since it is still using the class label as input to the generator.

We additionally examine the impact of the kernel choice and the dimension of the features on the KDD-GAN. The results are shown in Table 6.2. We compare the vMF kernel, which is equivalent to the RBF kernel due to the normalization used, with the IQ kernel [149]. We observe a similar performance level in CIFAR-10 for both kernel choices. Regarding K, we compare our default setting on CIFAR-10 ($K = 128$) to $K = 64$ and $K = 256$. Although increasing K slightly improves the IS, the best model overall remains the default one. We can conclude from both experiments that our KDD loss is not too sensitive to the choice of the kernel and dimension of the features

Table 6.2: **KDD GAN kernel and dimensionality choice**. We evaluate the impact of Kernel Choice and Feature Dimension on KDD GAN

Kernel	**Feature Dimension**	**FID**	**IS**
vMF	$K = 128$	8.384	8.887
IQ	$K = 128$	8.375	8.901
vMF	$K = 64$	8.842	8.885
vMF	$K = 128$	8.375	8.901
vMF	$K = 256$	9.050	9.058

Table 6.3: **Experimental results on CIFAR10**. The values shown below are obtained after 100k iterations. We show the benefit of adding various augmentation factors for the KDD setting. We also explore the effect of the Jacobian regularization. * are numbers reported by [62].

Experiments	**FID** $\downarrow$	**IS** $\uparrow$	**D** $\uparrow$	**C** $\uparrow$
ContraGAN*	8.065	9.729	-	-
ContraGAN + DiffAug*	7.193	<u>9.996</u>	-	-
BigGAN + DiffAug*	**7.157**	9.775	-	-
BigGAN + CR*	<u>7.178</u>	**10.380**	-	-
Hinge loss	8.859	8.814	0.917	0.841
KDD	8.375	8.901	0.875	0.845
KDD + DA	7.089	9.250	0.893	0.860
KDD + DA $\times 3$	<u>6.063</u>	9.280	<u>0.951</u>	<u>0.892</u>
KDD + DA $\times 7$	**5.713**	**9.389**	**0.968**	**0.899**
KDD + JacD	7.944	8.959	0.895	0.847
KDD + JacD + DA$\times 7$	6.713	<u>9.333</u>	0.9000	0.875

as opposed to reported observations for models such as MMD-GAN [131].

6.4.2 Generative Learning on CIFAR10

In Table 6.3, we compare the performance of different variations of our KDD GAN with a BigGAN baseline and the numbers reported by [62] for a selection of their best models. The KDD GAN outperforms the BigGAN baseline for IS and FID. Also, it drastically improves its FID when using augmentations as described in Sec. 6.2.1. Augmentation $\times N$ means that an additional $N \times \textit{batchsize}$ augmented images are used for the KDE anchor points. We observe that on CIFAR10, the number of augmentations is positively correlated with a significant improvement of the FID. In the case of the Jacobian regularization, the results are mixed. It seems to improve the performance of the KDD model, but it also negatively impacts performance when used in combination with data augmentation. The Jacobian regularization may be too strict a requirement, as the dimension K of the gradient of ϕ is smaller than the dimension d of the images, and perhaps a more flexible loss term could work better.

6.4.3 Generative Learning on ImageNet

Tiny ImageNet. Table 6.4 shows the performance of our models on Tiny ImageNet compared to the SA-GAN baseline and the best models reported by [62]. The KDD GAN outperforms the baseline for all settings. On the one hand, similarly to CI-

Table 6.4: **Experimental results on Tiny ImageNet**. We compare the baseline to both the KDD and joint trainings. We also explore the effect of adding the Jacobian regularization on D and show the effect of using more augmentations for the density estimation. * are numbers reported by [62].

Experiments	γ	**FID** $\downarrow$	**IS** $\uparrow$	**D** $\uparrow$	**C** $\uparrow$
ContraGAN*	-	27.027	13.494	-	-
+ DiffAugment*	-	**15.755**	**17.303**	-	-
Hinge loss	-	29.525	11.048	0.520	0.516
KDD	-	24.022	13.204	0.658	0.613
KDD+DA	-	<u>20.204</u>	<u>14.100</u>	<u>0.673</u>	<u>0.663</u>
KDD+DA $\times 3$	-	**18.261**	**14.943**	**0.716**	**0.683**
KDD+JacD	-	25.504	13.215	0.597	0.595
KDD+JacD+DA	-	20.717	13.787	0.630	0.645
Joint	1	25.709	13.124	0.595	0.582
Joint+DA	1	22.854	13.421	0.591	0.613
Joint+JacD	1	26.369	13.169	0.582	0.582
Joint+JacD+DA	1	21.512	13.728	0.639	0.627
Joint	0.5	24.341	13.337	0.626	0.614
Joint+DA	0.5	23.357	12.918	0.619	0.621
Joint+JacD	0.5	23.854	13.251	0.651	0.617
Joint+JacD+DA	0.5	23.928	13.059	0.575	0.594

FAR10, using additional augmented images for the KDE results in a significant boost in performance. Indeed the KDD GAN with DA $\times 3$ outperforms ContraGAN in terms of FID and IS. On the other hand, the additional Jacobian regularization is not helpful. The only exception being the joint training ($\gamma = 0.5$) without data augmentation and the joint training with $\gamma = 1$ and data augmentation where Jacobian regularization introduces a slight performance boost. Note that the ContraGAN+Diff.Aug. numbers reported by [62] were obtained using twice as many iterations as the rest of the models (ContraGAN and our experiments), putting it at an advantage.

ImageNet 64 $\times$ 64. Table 6.5 shows our experimental results on ImageNet 64×64. We compare our models to the SA-GAN baseline and the numbers reported by [29] and [166]. For all our trained models, we use Instance Selection [29] with a retention ratio of 50%. We observe that the baseline outperforms our KDD GAN even with additional augmentations and regularization. It is also worth noting that, in this setting, although a small amount of data augmentation seems to help, adding more is not necessarily beneficial. The high level of diversity in ImageNet both in terms of the number of

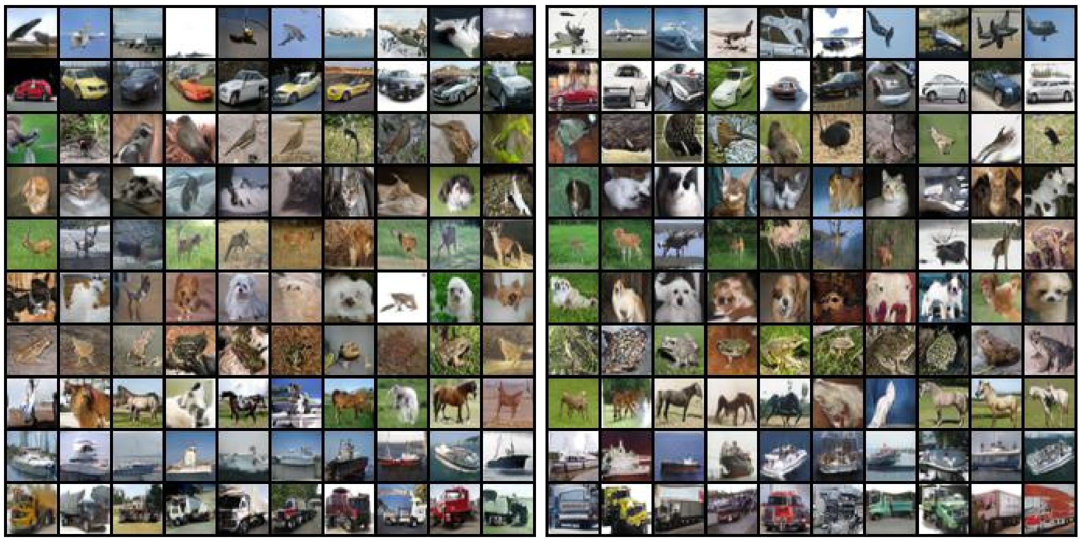
Table 6.5: **Experimental results on ImageNet** 64×64. We explore the use of augmentation, Jacobian regularization and Joint training. † refers to a setting where the feature $\phi(x_r)$ were computed using the weights from the previous discriminator update step. * are numbers reported by [29].

Experiments	γ	**FID** ↓	**IS** ↑	**D** ↑	**C** ↑
SA-GAN+IS@50%*	-	<u>9.63</u>	31.04	<u>1.07</u>	<u>0.88</u>
FQ-BigGAN*	-	<u>9.67</u>	25.96	-	-
Hinge loss	-	10.452	32.869	1.034	0.877
KDD	-	12.570	31.404	0.953	0.850
KDD+DA	-	12.367	31.069	0.954	0.861
KDD+DA $\times 3$	-	14.680	27.949	0.928	0.810
KDD+JacD	-	12.651	31.188	0.938	0.850
KDD+JacD+DA	-	79.790	10.603	0.376	0.139
Joint	1	11.387	32.471	0.991	0.872
Joint+DA	1	10.385	33.753	1.048	0.880
Joint+JacD	1	10.320	34.296	1.010	0.868
Joint+JacD+DA	1	9.702	34.619	1.062	<u>0.892</u>
Joint	0.5	10.544	33.447	1.017	0.879
Joint †	0.5	**9.450**	**35.648**	<u>1.070</u>	**0.897**
Joint+DA	0.5	10.111	33.494	1.048	<u>0.891</u>
Joint+JacD	0.5	10.242	<u>35.120</u>	**1.072**	<u>0.891</u>
Joint+JacD+DA	0.5	10.010	34.074	1.053	0.889

classes and samples might limit the effectiveness of our density estimation given the relatively small batch size used. However, all joint training models outperform the hinge-based models in terms of IS and most outperform our SA-GAN baseline in terms of FID. Interestingly, the best model is the Joint† model where $\hat{p}_r$ is estimated using features computed during the last discriminator step. This suggests that using a memory bank for the features might be a promising extension of this work.

6.5 Examples of Generated Images

We show non-cherry picked images generated by our Hinge loss baseline and our best model per dataset in Figures 6.3 to 6.9. The truncation trick was not used [17]. In all figures, each row represents a different class starting with the first class in the top row down to the last class in the bottom row.



(a) Hinge loss (b) KDD + Aug ×7

Figure 6.3: **Qualitative results on CIFAR10**. Samples generated using the Hinge loss model and the KDD + Aug ×7 model trained on CIFAR10 (one class per row).

6.6 Limitations and Future Work

One of the main challenges in the use of KDD GAN is to ensure that the anchor points for the KDE are representative of the evaluation points. In our experiments between Tiny ImageNet and ImageNet 64×64, we observe that the performance of KDD GAN is sensitive to the size of the set anchor points, the number of augmentations, and the complexity of the dataset seems to also play a role. Also, with large datasets, the impact of samples at the tails of the distribution on the KDE approximation is unclear. In general, it might be necessary to design better sampling strategies for the anchor points used for the KDE estimation: Some options are to use a memory bank or to sample using k-NN. Another direction to evaluate is the role of class projection in the training. We chose to separate the category aspect from the unlabeled problem not only because it would make KDD GAN suitable for unsupervised learning, but also because it would not require large minibatches since the current KDE completely ignores the class labels. It would be interesting to evaluate the performance in the case where the loss with class labels is entirely based on KDE. Finally, as mentioned in the introduction, KDD GAN can be combined with other techniques and regularization methods that are known to improve the performance of the GAN training, such as Consistency Regularization of [162] and Differentiable Augmentation of [164]. We leave these investigations to future work.

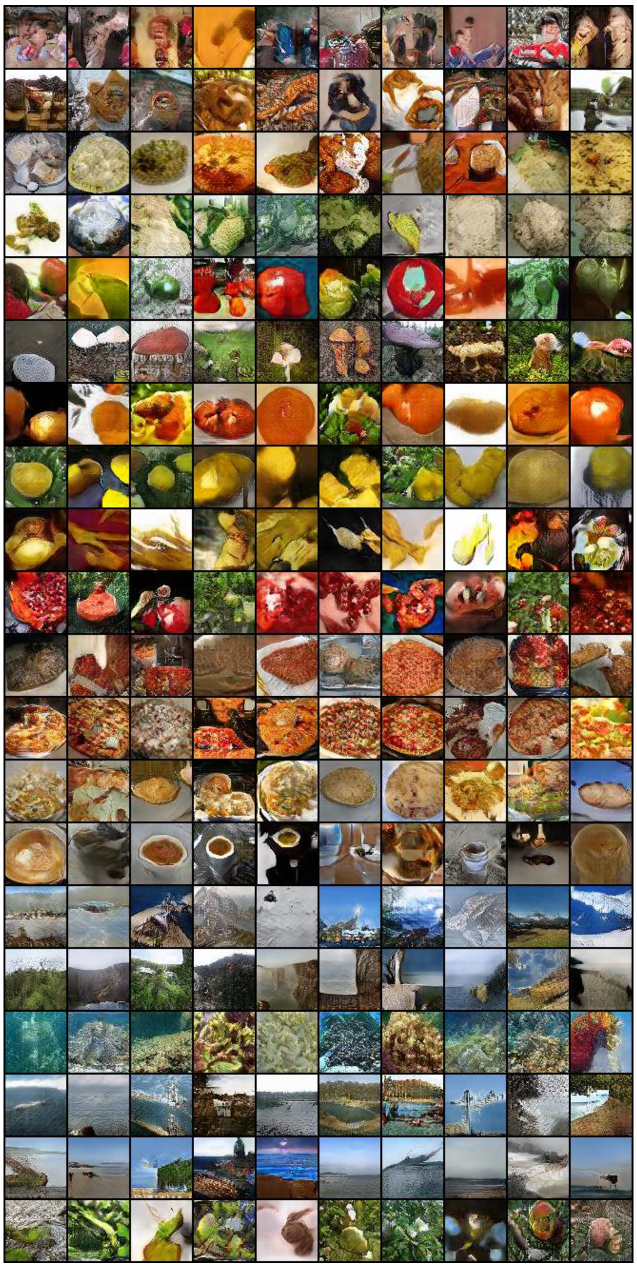


Figure 6.4: **Qualitative results on Tiny ImageNet for classes 181-200 using the Hinge loss**. Samples generated using the Hinge loss model trained on Tiny ImageNet for classes 181-200 (one class per row).

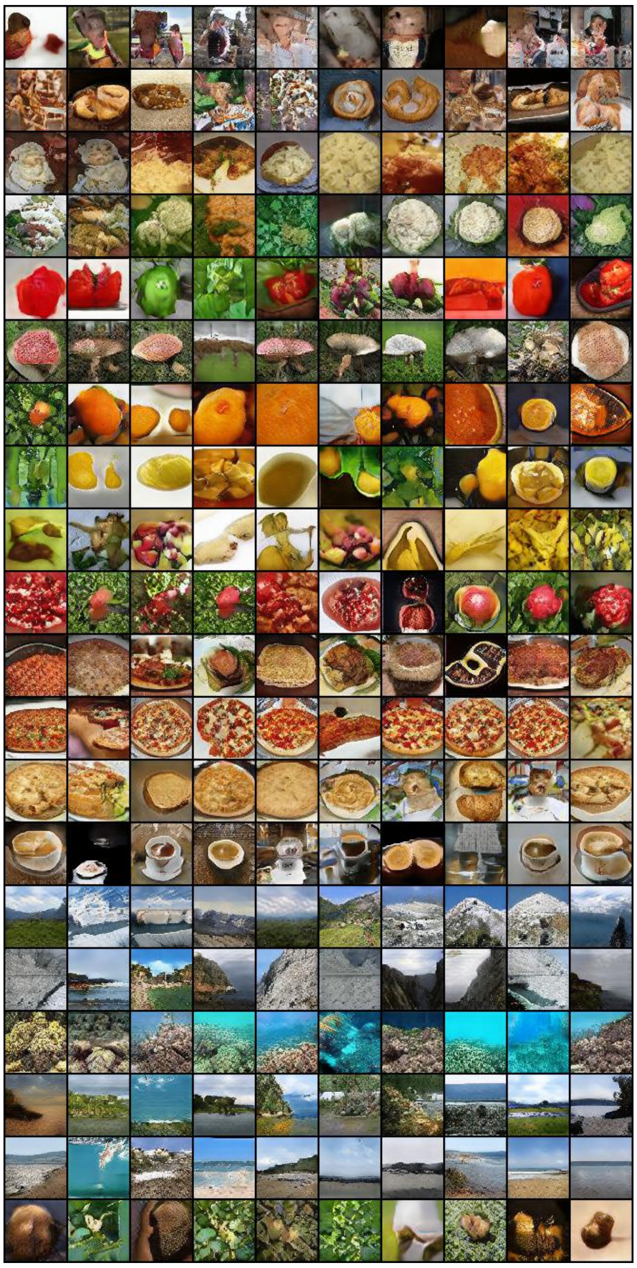


Figure 6.5: **Qualitative results on Tiny ImageNet using the KDD loss**. Samples generated using the KDD+Aug $\times 3$ model trained on Tiny ImageNet for the classes 181-200 (one class per row).

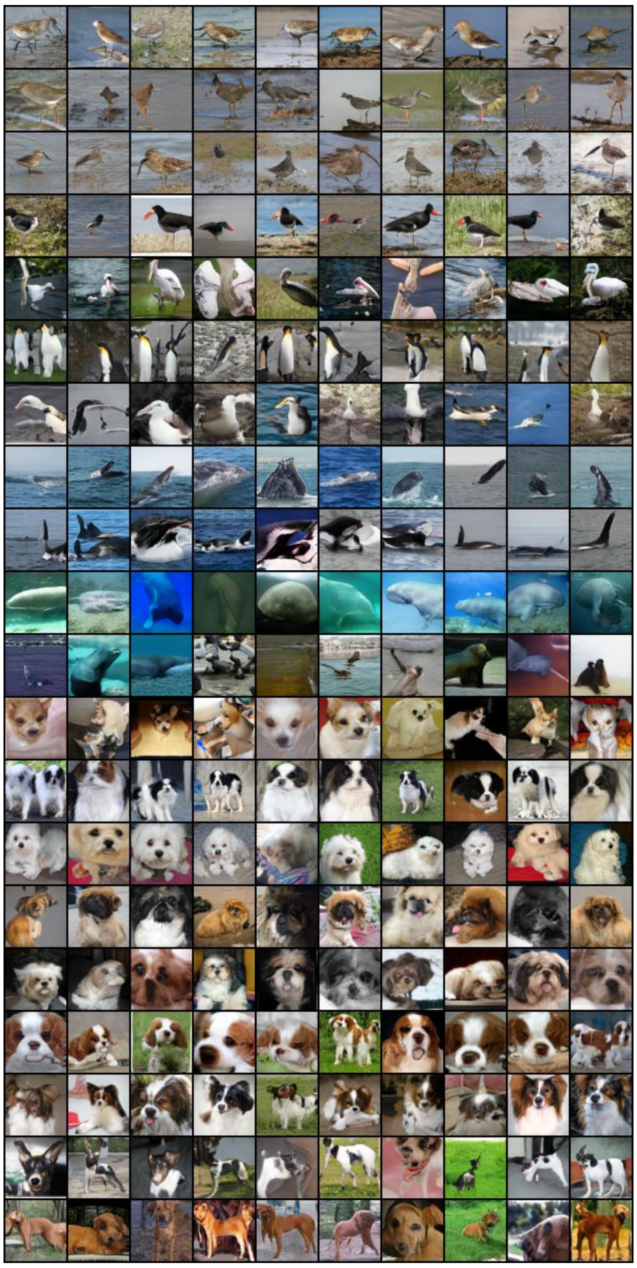


Figure 6.6: **Qualitative results on ImageNet**64×64 **using the Hinge loss**. Samples generated using Hinge loss model trained on ImageNet 64×64 for the classes 141-160 (one class per row).

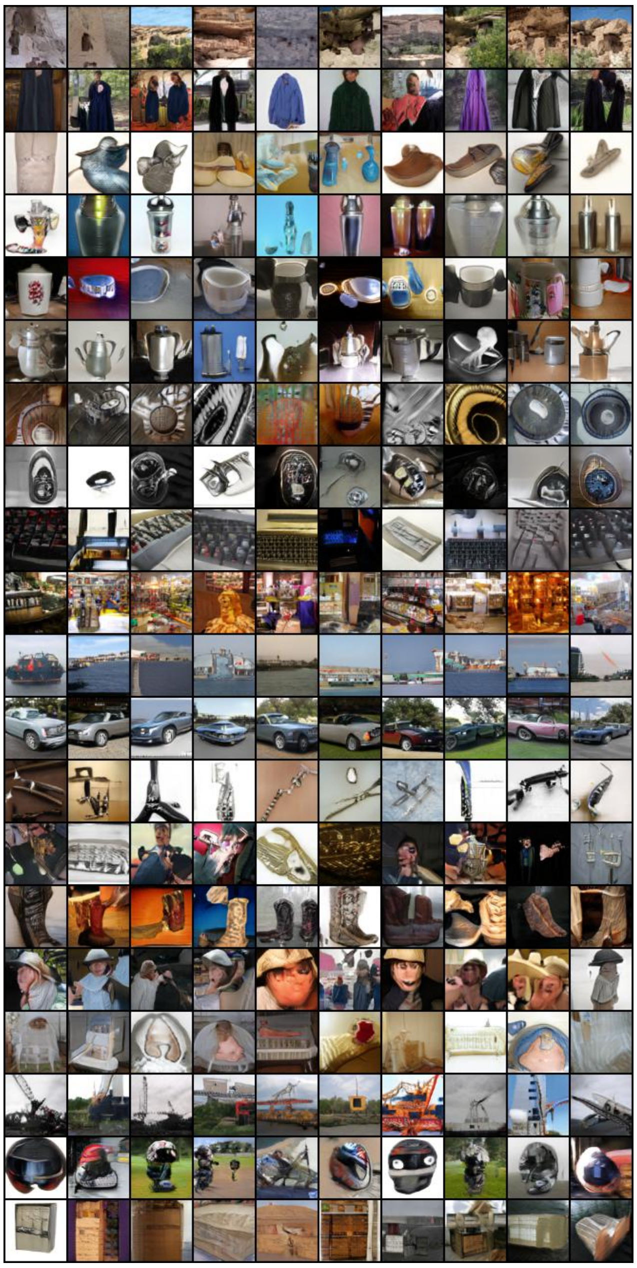


Figure 6.7: **Qualitative results on ImageNet**64×64 **using the Hinge loss**. Samples generated using Hinge loss model trained on ImageNet 64×64 for the classes 501-520 (one class per row).

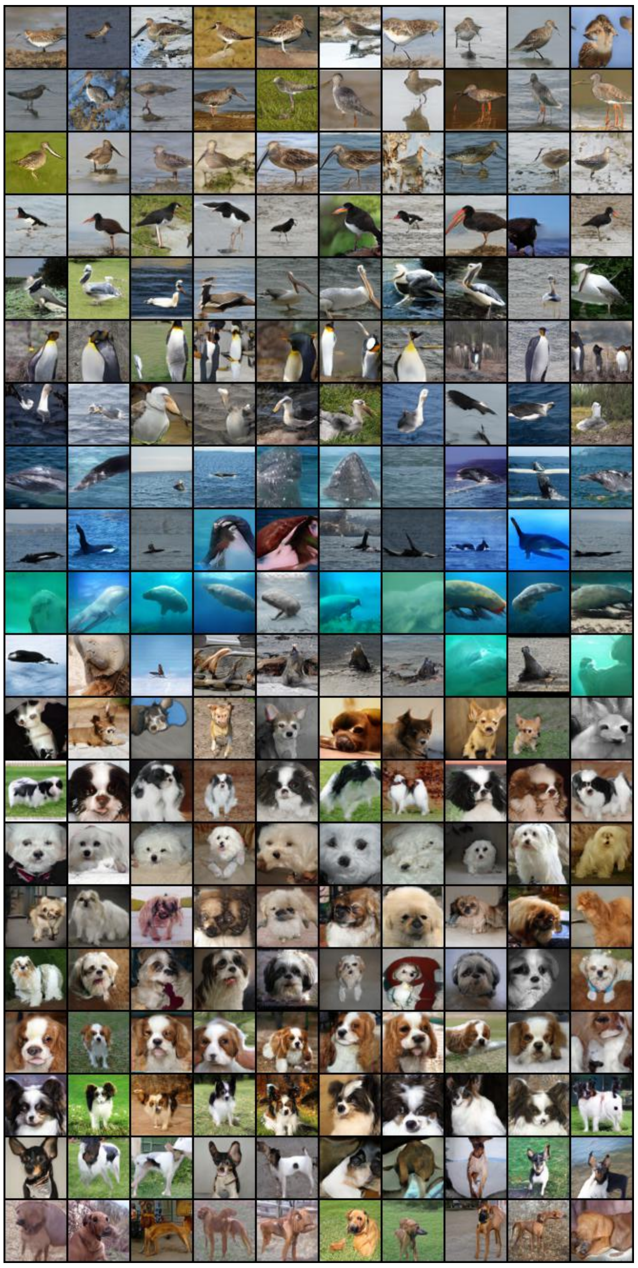


Figure 6.8: **Qualitative results on ImageNet**64×64 **using the KDD loss**. Samples generated using Joint† model trained on ImageNet 64×64 for the classes 141-160 (one class per row).

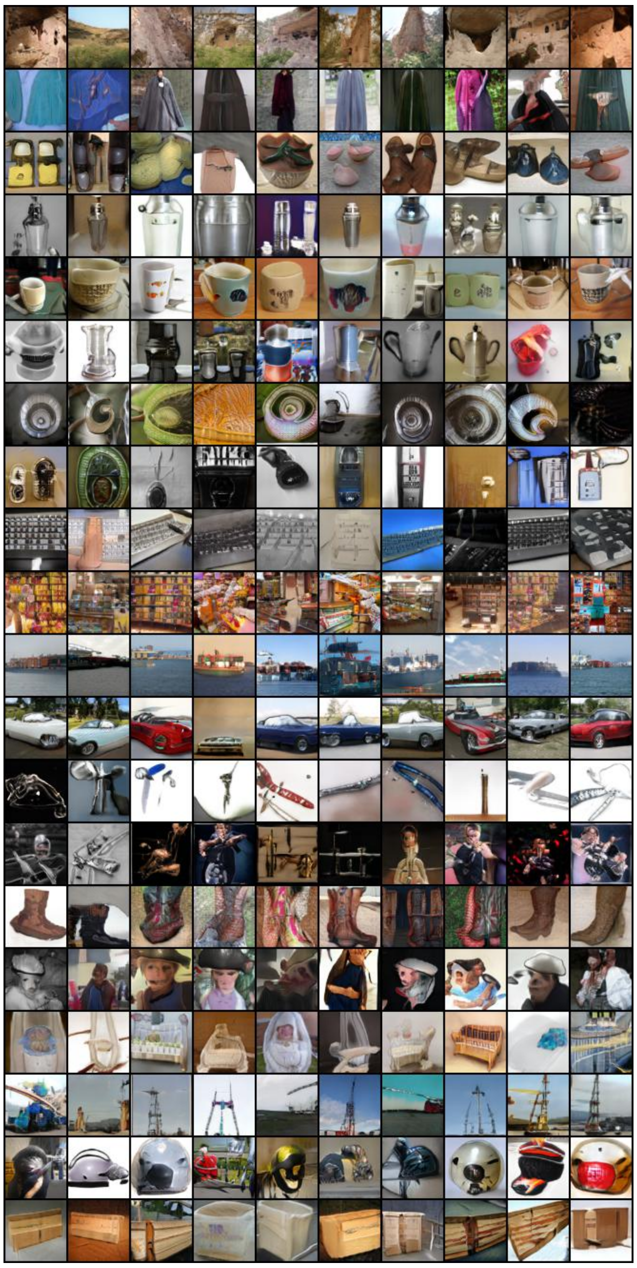


Figure 6.9: **Qualitative results on ImageNet64×64 using the KDD loss**. Samples generated using Joint† model trained on ImageNet 64×64 for the classes 501-520 (one class per row).

Chapter 7

Conclusions

This thesis studies the problem of building more robust and generalizable machine learning in multiple settings. In Chapter 3, we introduce *Phase Swap*, a biologically inspired self-supervised learning task for physiological time series. By training the model to detect correspondence between the phase and amplitude information of a given signal, we obtain features that learn to focus on the structured parts of the signal and ignore the noisier ones. Such features are shown to generalize better to new subjects across different medical time-series classification tasks even when only a limited number of training subjects is available. In Chapter 4, we combine the *Phase Swap* features and meta-learning into a training algorithm *S2MAML* that explicitly emphasizes generalization. On top of generalizing better to new subjects, *S2MAML* is able to significantly outperform the standard meta-learning baseline in terms of zero-shot generalization to completely new datasets with electrodes and demographic properties different from those used for training. We also introduce *SemiGPC* in Chapter 5, a semi-supervised label refinement method tailored for imbalanced image classification. By leveraging a more efficient Gaussian Processes inference logic, we build a classifier head whose confidence maps are locally sensitive to minority class samples while preserving the global data distribution structure. This significantly increases the semi-supervised performance of the considered models across a broad range of challenging imbalanced semi-supervised benchmarks. Lastly, in Chapter 6, we define a novel GAN loss based on kernel density discrimination (KDD GAN) as a statistical divergence between the kernel density estimates of the real and fake data distribution in the feature space of the discriminator. Unlike standard GAN losses whose statistical divergence interpretation depends on the optimality of the discriminator, our KDD loss is valid for any discriminator. This results in better training gradients that encourage the generator to seek missing distribution modes.

As shown throughout this thesis, model robustness can take different meaning, ranging from overcoming the risk of capturing spurious correlations in noisy setting by learning to identify general purpose patterns that are useful for both unseen subjects and cohorts, handling class imbalances and other forms of data biases when training with limited annotations, to addressing mode collapse in generative training. The generalization and robustness problem is not an artifact of low-data regimes and persists as the amount of available data scales up. Indeed, collecting more data is rarely a bias-free process and often introduces more distribution shifts and imbalances while increasing the complexity of the learning problem. Therefore, as highlighted in this thesis, tackling the generalization problem requires a careful design of the adopted training schemes and models. Beyond the content of this thesis, we identify in the following some extensions and future works based on our contributions.

Extending *Phase Swap* to spatial-temporal case. On top the temporal aspect that motivated the design of our *Phase Swap* task, the brain activity is also known to exhibit a strong spatial component, where different regions of the brain communicate using different waves and global patterns. One could extend the *Phase Swap* task to this setting by adopting a spatiotemporal Fourrier transform. The features obtained would be able to capture both local and global brain activity patterns.

Multi-task meta-learning. In Chapter 4, *S2MAML* is only trained on the sleep scoring task. However, the design of the various meta-learning mini-batches is completely modular and could allow for mixing different EEG-based tasks. This way, one could boost the performance on tasks with more limited data, such as seizure detection, thanks to the potential feature cross-over from other tasks with more abundant data, such as sleep scoring.

Incorporating the GP uncertainty estimate into the semi-supervised training. In addition to the mean function that we used to derive our *SemiGPC* method, Gaussian Processes also provide a covariance estimate to quantify their uncertainty. Provided we can preventing trivial solutions such as variance shrinkage, semi-supervised training could benefit from a more principled uncertainty/confidence estimate.

Sliced and Mixture of Experts versions of KDD GAN. In Chapter 6, our KDD loss defines a statistical divergence in feature space. Unlike the input space, the choice of the feature space is not unique. Indeed, there exist as many feature spaces as possible discriminator parameter configurations. Therefore, an analog version of our KDD loss could be designed to the sliced Wasserstein distance [27] where the feature space is randomly selected per minibatch or adopt a mixture of experts (MoE) [7] approach with multiple feature spaces.

Bibliography

[1] 2018 fgvcx fungi classification challenge. https://github.com/visipedia/fgvcx_fungi_comp. 79

[2] inaturalist 2018 competition dataset. https://github.com/visipedia/inat_comp/tree/master/2018, 2018. 79

[3] Yasunori Aoki, Masahiro Hata, Masao Iwase, Ryouhei Ishii, Roberto D Pascual-Marqui, Takufumi Yanagisawa, Haruhiko Kishima, and Manabu Ikeda. Cortical electrical activity changes in healthy aging using eeg-eloreta analysis. *Neuroimage: Reports*, 2(4):100143, 2022. 29

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. 39, 90

[5] Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017. 20

[6] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *Computer Vision – ECCV 2022*, pages 456–473. Springer, 2022. 69, 71, 85, 86

[7] Tara Baldacchino, Elizabeth J Cross, Keith Worden, and Jennifer Rowson. Variational bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems. *Mechanical Systems and Signal Processing*, 66:178–200, 2016. 112

[8] Nannapas Banluesombatkul, Pichayoot Ouppaphan, Pitshaporn Leelaarporn, Payongkit Lakhan, Busarakum Chaitusaney, Nattapong Jaimchariya, Ekapol Chuangsuwanich, Wei Chen, Huy Phan, Nat Dilokthanakul, et al.

Metasleeplearner: A pilot study on fast adaptation of bio-signals-based sleep stage classifier to new individual subject using meta-learning. *IEEE Journal of Biomedical and Health Informatics*, 2020. 29, 56, 57

[9] Hubert Banville, Graeme Moffat, Isabela Albuquerque, Denis-Alexander Engemann, Aapo Hyvärinen, and Alexandre Gramfort. Self-supervised representation learning from electroencephalography signals. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019. 30, 49

[10] 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. 60

[11] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(11), 2006. 36

[12] D. Bernstein. *Matrix Mathematics.* Princeton University Press, 2005. 76

[13] Richard B Berry, Rita Brooks, Charlene E Gamaldo, Susan M Harding, C Marcus, Bradley V Vaughn, et al. The aasm manual for the scoring of sleep and associated events. *Rules, Terminology and Technical Specifications, Darien, Illinois, American Academy of Sleep Medicine*, 176:2012, 2012. 26, 27, 29

[14] David Berthelot, Nicholas Carlini, Ekin Dogus Cubuk, Alexey Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations*, 2020. 37, 69, 71, 72, 73, 85

[15] Alceu Bissoto, Eduardo Valle, and Sandra Avila. Gan-based data augmentation and anonymization for skin-lesion analysis: A critical review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1847–1856, June 2021. 22

[16] Léon Bottou. Online algorithms and stochastic approximations. *Online learning and neural networks*, 1998. 96

[17] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018. 39, 90, 91, 98, 99, 103

[18] Niko A Busch, Julien Dubois, and Rufin VanRullen. The phase of ongoing eeg oscillations predicts visual perception. *Journal of Neuroscience*, 29(24):7869–7876, 2009. 42, 43

[19] Nicholas Carlini, Matthew Jagielski, Chiyuan Zhang, Nicolas Papernot, Andreas Terzis, and Florian Tramer. The privacy onion effect: Memorization is relative. *Advances in Neural Information Processing Systems*, 35:13263–13276, 2022. 20

[20] 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. 20, 31, 32, 34, 69, 71, 85

[21] 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. 20, 31, 32, 39

[22] Ting Chen, Calvin Luo, and Lala Li. Intriguing properties of contrastive losses. *arXiv preprint arXiv:2011.02803*, 2020. 31

[23] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019. 19, 37

[24] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 19

[25] Tristan Deleu, David Kanaa, Leo Feng, Giancarlo Kerg, Yoshua Bengio, Guillaume Lajoie, and Pierre-Luc Bacon. Continuous-time meta-learning with forward mode differentiation. In *International Conference on Learning Representations*, 2022. 35

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 34, 78, 85, 91, 97

[27] Ishan Deshpande, Ziyu Zhang, and Alexander G Schwing. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3483–3491, 2018. 112

[28] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. In *arXiv preprint arXiv:1708.04552*, 2017. 37

[29] Terrance DeVries, Michal Drozdzal, and Graham W Taylor. Instance selection for gans. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13285–13296. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/99f6a934a7cf277f2eaece8e3ce619b2-Paper.pdf`. 39, 91, 98, 102, 103

[30] Fadi Dornaika, Jingjun Bi, and Chongsheng Zhang. A unified deep semi-supervised graph learning scheme based on nodes re-weighting and manifold regularization. *Neural Networks*, 158:188–196, 2023. 36

[31] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 22

[32] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 85

[33] Valeria Drago, Debora Aricò, Kenneth Heilman, Paul Foster, John Williamson, Pasquale Montagna, and Raffaele Ferri. The correlation between sleep and creativity. *Nature Precedings*, pages 1–1, 2010. 29

[34] Aysegul Dundar, Karan Sapra, Guilin Liu, Andrew Tao, and Bryan Catanzaro. Panoptic-based image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8070–8079, 2020. 89

[35] Vincent Dutordoir, James Hensman, Mark van der Wilk, Carl Henrik Ek, Zoubin Ghahramani, and Nicolas Durrande. Deep neural networks as point estimates for deep gaussian processes. In *NeurIPS*, 2021. 71

[36] Yue Fan, Dengxin Dai, Anna Kukleva, and Bernt Schiele. Cossl: Co-learning of representation and classifier for imbalanced semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14574–14584, 2022. 80, 81, 82

[37] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022. 17

[38] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017. 34, 35, 57, 60

[39] Luigi Fiorillo, Paolo Favaro, and Francesca Dalia Faraci. Deepsleepnet-lite: A simplified automatic sleep stage scoring model with uncertainty estimates. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:2076–2085, 2021. 30, 56, 57, 61, 62

[40] Robert S Fisher, Helen E Scharfman, and Marco DeCurtis. How can we identify ictal and interictal abnormal activity? In *Issues in Clinical Epileptology: A View from the Bench*, page 7. Springer, 2014. 47

[41] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016. 71

[42] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bygh9j09KX. 20

[43] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000. 45

[44] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 22, 38, 90, 92

[45] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 34

[46] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5769–5779, 2017. 97

[47] 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. 32

[48] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022. 33

[49] Conor Heneghan. St. vincent's university hospital/university college dublin sleep apnea database, 2011. 58

[50] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6629–6640, 2017. 38, 98

[51] Jie Hong, Pengfei Fang, Weihao Li, Tong Zhang, Christian Simon, Mehrtash Harandi, and Lars Petersson. Reinforced attention for few-shot learning and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 913–923, 2021. 35

[52] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 17

[53] Sheng-Wei Huang, Che-Tsung Lin, Shu-Ping Chen, Yen-Yi Wu, Po-Hao Hsu, and Shang-Hong Lai. Auggan: Cross domain adaptation with gan-based data augmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 22, 38

[54] Ahmed Imtiaz Humayun, Asif Shahriyar Sushmit, Taufiq Hasan, and Mohammed Imamul Hassan Bhuiyan. End-to-end sleep staging with raw single channel eeg using deep residual convnets. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 1–5. IEEE, 2019. 30, 52

[55] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015. 44

[56] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 89

[57] Junji Ito, Andrey R Nikolaev, and Cees Van Leeuwen. Spatial and temporal structure of phase synchronization of spontaneous alpha eeg activity. *Biological cybernetics*, 92(1):54–60, 2005. 50

[58] Alice F Jackson and Donald J Bolger. The neurophysiological bases of eeg and eeg measurement: A review for the rest of us. *Psychophysiology*, 51(11):1061–1071, 2014. 25, 29

[59] Behrouz Jafari and Vahid Mohsenin. Polysomnography. *Clinics in chest medicine*, 31(2):287–297, 2010. 25

[60] Jongheon Jeong and Jinwoo Shin. Training GANs with stronger augmentations via contrastive discriminator. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=eo6U4CAwVmg. 39

[61] Joakim Johnander, Johan Edstedt, Michael Felsberg, Fahad Shahbaz Khan, and Martin Danelljan. Dense gaussian processes for few-shot segmentation. In *ECCV*, 2022. 71

[62] Minguk Kang and Jaesik Park. Contragan: Contrastive learning for conditional image generation. In *NeurIPS 2020*. Neural Information Processing Systems, 2020. 39, 90, 101, 102

[63] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 38, 90

[64] Bob Kemp, Aeilko H Zwinderman, Bert Tuk, Hilbert AC Kamphuisen, and Josefien JL Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000. 45, 46

[65] Bob Kemp, Aeilko H Zwinderman, Bert Tuk, Hilbert AC Kamphuisen, and Josefien JL Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000. 56, 57

[66] Sirvan Khalighi, Teresa Sousa, José Moutinho Santos, and Urbano Nunes. Isruc-sleep: a comprehensive public dataset for sleep researchers. *Computer methods and programs in biomedicine*, 124:180–192, 2016. 45, 46, 48, 58

[67] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In H. Larochelle, M. Ranzato, R. Had-

sell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/d89a66c7c80a29b1bdbab0f2a1a94af8-Paper.pdf. 90

[68] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 46

[69] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. 62

[70] Jens G Klinzing, Niels Niethard, and Jan Born. Mechanisms of systems memory consolidation during sleep. *Nature neuroscience*, 22(10):1598–1610, 2019. 29

[71] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 93

[72] Nikos Komodakis and Spyros Gidaris. Unsupervised representation learning by predicting image rotations. In *International conference on learning representations (ICLR)*, 2018. 33

[73] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 91, 97

[74] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8183–8192, 2018. 89

[75] Neil Lawrence and Michael Jordan. Semi-supervised learning via gaussian processes. *Advances in Neural Information Processing Systems*, 17, 2004. 71

[76] Ya Le and X. Yang. Tiny imagenet visual recognition challenge. 2015. URL https://github.com/seshuad/IMagenet. 91, 97

[77] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 2020. 31

[78] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative

adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 89

[79] Jongseok Lee, Jianxiang Feng, Matthias Humt, Marcus Gerhard Müller, and Rudolph Triebel. Trust your robots! predictive uncertainty estimation of neural networks with sparse gaussian processes. In *CoRL*, 2022. 71

[80] Abdelhak Lemkhenter and Paolo Favaro. Boosting generalization in bio-signal classification by learning the phase-amplitude coupling. In *DAGM German Conference on Pattern Recognition*, pages 72–85. Springer, 2020. 56

[81] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Manigan: Text-guided image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7880–7889, 2020. 89

[82] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: towards deeper understanding of moment matching network. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2200–2210, 2017. 91

[83] Denghao Li, Pablo Ortega, Xiaoxi Wei, and Aldo Faisal. Model-agnostic meta-learning for eeg motor imagery decoding in brain-computer-interfacing. In *2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 527–530. IEEE, 2021. 57

[84] Junnan Li, Caiming Xiong, and Steven CH Hoi. Comatch: Semi-supervised learning with contrastive graph regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9475–9484, 2021. 37, 69, 70, 72, 73, 74, 75

[85] Leheng Li, Qing Lian, Luozhou Wang, Ningning Ma, and Ying-Cong Chen. Lift3d: Synthesize 3d training data by lifting 2d gan to 3d generative radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 332–341, June 2023. 38

[86] Chieh Hubert Lin, Hsin-Ying Lee, Yen-Chi Cheng, Sergey Tulyakov, and Ming-Hsuan Yang. Infinitygan: Towards infinite-resolution image synthesis. *arXiv preprint arXiv:2104.03963*, 2021. 90

[87] Rui Liu, Yixiao Ge, Ching Lam Choi, Xiaogang Wang, and Hongsheng Li. Divco: Diverse conditional image synthesis via contrastive generative adversarial network. *arXiv preprint arXiv:2103.07893*, 2021. 90

[88] Zhao-Yang Liu, Shao-Yuan Li, Songcan Chen, Yao Hu, and Sheng-Jun Huang. Uncertainty aware graph gaussian process for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4957–4964, 2020. 71

[89] María Eugenia López, Sandra Pusil, Ernesto Pereda, Fernando Maestú, and Francisco Barceló. Dynamic low frequency eeg phase synchronization patterns during proactive control of task switching. *NeuroImage*, 186:70–82, 2019. 42, 43

[90] Guillaume Lorre, Jaonary Rabarisoa, Astrid Orcesi, Samia Ainouz, and Stephane Canu. Temporal contrastive pretraining for video action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020. 33

[91] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 78

[92] Vitaly Maiorov and Allan Pinkus. Lower bounds for approximation by mlp neural networks. *Neurocomputing*, 25(1-3):81–91, 1999. 17

[93] Jaakko Malmivuo, Robert Plonsey, et al. *Bioelectromagnetism: principles and applications of bioelectric and biomagnetic fields*. Oxford University Press, USA, 1995. 46

[94] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017. 90

[95] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2445, 2020. 89

[96] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018. 39

[97] Kana Miyamoto, Hiroki Tanaka, and Satoshi Nakamura. Meta-learning for emotion prediction from eeg while listening to music. In *Companion Publication of the 2021 International Conference on Multimodal Interaction*, pages 324–328, 2021. 57

[98] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018. 91, 92, 95, 96, 99, 100

[99] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 90

[100] MS Mourtazaev, B Kemp, AH Zwinderman, and HAC Kamphuisen. Age and gender affect different characteristics of slow waves in the sleep eeg. *Sleep*, 18 (7):557–564, 1995. 44, 45

[101] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pages 7176–7185. PMLR, 2020. 98

[102] Amine Naït-Ali. *Advanced biosignal processing.* Springer Science & Business Media, 2009. 41

[103] Benedict Shien Wei Ng, Nikos K Logothetis, and Christoph Kayser. Eeg phase patterns reflect the selectivity of neural firing. *Cerebral Cortex*, 23(2):389–398, 2013. 42, 43

[104] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*. 35

[105] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 42

[106] Roman Novak, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Fast finite width neural tangent kernel. In *International Conference on Machine Learning*, pages 17018–17044, 2022. 88

[107] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: training generative neural samplers using variational divergence minimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 271–279, 2016. 39

[108] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 32

[109] 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, 34

[110] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and manipulation. In *European Conference on Computer Vision*, pages 262–277. Springer, 2020. 89

[111] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002. 38

[112] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, pages 319–345. Springer, 2020. 89

[113] 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):1–12, 2021. 29, 30, 65

[114] 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. 29, 30, 56

[115] Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. 38

[116] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12556–12565, 2020. 35

[117] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL `http://jmlr.org/papers/v21/20-074.html`. 17

[118] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019. 35

[119] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. 89

[120] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 38

[121] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*, volume 1. Springer, 2006. 74

[122] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069. PMLR, 2016. 89

[123] Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H Falk, and Jocelyn Faubert. Deep learning-based electroencephalography analysis: a systematic review. *Journal of neural engineering*, 16(5):051001, 2019. 30

[124] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2234–2242, 2016. 98

[125] Rick Sauber-Cole and Taghi M Khoshgoftaar. The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey. *Journal of Big Data*, 9(1):98, 2022. 38

[126] Kendrick Shen, Robbie M Jones, Ananya Kumar, Sang Michael Xie, Jeff Z HaoChen, Tengyu Ma, and Percy Liang. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 19847–19878. PMLR, 2022. 32

[127] Ali Hossam Shoeb. *Application of machine learning to epileptic seizure onset detection and treatment*. PhD thesis, Massachusetts Institute of Technology, 2009. 45, 46

[128] Revati Shriram, M Sundhararajan, and Nivedita Daimiwal. Eeg based cognitive workload assessment for maximum efficiency. *Int. Organ. Sci. Res. IOSR*, 7: 34–38, 2013. 27

[129] Jerome M Siegel. Clues to the functions of mammalian sleep. *Nature*, 437(7063): 1264–1271, 2005. 56

[130] Vikas Sindhwani, Wei Chu, and S Sathiya Keerthi. Semi-supervised gaussian process classifiers. In *IJCAI*, pages 1059–1064, 2007. 71

[131] Mathieu Sinn and Ambrish Rawat. Non-parametric estimation of jensen-shannon divergence in generative adversarial network training. In *International Conference on Artificial Intelligence and Statistics*, pages 642–651. PMLR, 2018. 91, 101

[132] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 34

[133] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, volume 33, pages 596–608, 2020. 37, 69, 71, 72, 73, 74, 85

[134] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *arXiv preprint arXiv:1705.07761*, 2017. 90

[135] Carl E Stafstrom and Lionel Carmant. Seizures and epilepsy: an overview for neuroscientists. *Cold Spring Harbor perspectives in medicine*, 5(6):a022426, 2015. 29

[136] Austin Stone, Daniel Maurer, Alper Ayvaci, Anelia Angelova, and Rico Jonschkowski. Smurf: Self-teaching multi-frame unsupervised raft with full-image warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2021. 22

[137] Jong-Chyi Su and Subhransu Maji. The semi-supervised inaturalist-aves challenge at fgvc7 workshop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 79

[138] Jong-Chyi Su and Subhransu Maji. The semi-supervised inaturalist challenge at the fgvc8 workshop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 79

[139] Jong-Chyi Su and Subhransu Maji. Semi-supervised learning with taxonomic labels. In *British Machine Vision Conference (BMVC), 2021*, 2021. 69, 71, 78, 82, 83, 84

[140] Jong-Chyi Su, Zezhou Cheng, and Subhransu Maji. A realistic evaluation of semi-supervised learning for fine-grained classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12966–12975, 2021. 69, 71, 78, 82, 83, 84

[141] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017. 17

[142] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. 22

[143] Mario Giovanni Terzano, Liborio Parrino, Adriano Sherieri, Ronald Chervin, Sudhansu Chokroverty, Christian Guilleminault, Max Hirshkowitz, Mark Mahowald, Harvey Moldofsky, Agostino Rosa, et al. Atlas, rules, and recording techniques for the scoring of cyclic alternating pattern (cap) in human sleep. *Sleep medicine*, 2(6):537–553, 2001. 58

[144] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 89

[145] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011. 79

[146] Jianfeng Wang, Thomas Lukasiewicz, Daniela Massiceti, Xiaolin Hu, Vladimir Pavlovic, and Alexandros Neophytou. Np-match: When neural processes meet semi-supervised learning. In *International Conference on Machine Learning*, pages 22919–22934, 2022. 71

[147] Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 88

[148] 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. 31, 90

[149] Wei Wang, Yuan Sun, and Saman Halgamuge. Improving mmd-gan training with repulsive loss function. In *International Conference on Learning Representations 2019*, 2019. 91, 95, 100

[150] Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie, and Yue Zhang. Usb: A unified semi-supervised learning benchmark for classification. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. 78, 80, 82, 84, 85, 87

[151] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. Freematch: Self-adaptive thresholding for semi-supervised learning. 2023. 37, 69, 71, 72, 84, 85

[152] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017. 44, 52

[153] Chen Wei, Kihyuk Sohn, Clayton Mellina, Alan Yuille, and Fan Yang. Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10857–10866, 2021. 11, 80, 86

[154] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018. 42

[155] Kristine A Wilckens, Fabio Ferrarelli, Matthew P Walker, and Daniel J Buysse. Slow-wave activity enhancement to improve cognition. *Trends in neurosciences*, 41(7):470–482, 2018. 28

[156] Katharina Wulff, Silvia Gatti, Joseph G Wettstein, and Russell G Foster. Sleep and circadian rhythm disruption in psychiatric and neurodegenerative disease. *Nature Reviews Neuroscience*, 11(8):589–599, 2010. 56

[157] Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *NeurIPS*, 2019. 71

[158] Ning Yu, Ke Li, Peng Zhou, Jitendra Malik, Larry Davis, and Mario Fritz. Inclusive gan: Improving data and minority coverage in generative models. In *European Conference on Computer Vision*, pages 377–393. Springer, 2020. 90

[159] Ning Yu, Guilin Liu, Aysegul Dundar, Andrew Tao, Bryan Catanzaro, Larry Davis, and Mario Fritz. Dual contrastive loss and attention for gans. *arXiv preprint arXiv:2103.16748*, 2021. 39, 90

[160] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017. 89

[161] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. 90, 98

[162] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *International Conference on Learning Representations*, 2019. 39, 90, 104

[163] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 89

[164] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33, 2020. 39, 98, 104

[165] Sicheng Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, Bichen Wu, Ravi Krishna, Joseph E Gonzalez, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, et al. A review of single-source deep unsupervised visual domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2): 473–493, 2020. 38

[166] Yang Zhao, Chunyuan Li, Ping Yu, Jianfeng Gao, and Changyou Chen. Feature quantization improves gan training. In *International Conference on Machine Learning*, pages 11376–11386. PMLR, 2020. 102

[167] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian, and Chang Xu. Simmatch: Semi-supervised learning with similarity matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14471–14481, 2022. 37, 69, 70, 71, 72, 73, 74, 85

[168] Yufan Zhou and Zhenyi Wang. Meta-learning with neural tangent kernels. In *The International Conference on Learning Representations (ICLR)*, 2021. 88

[169] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings*

of the IEEE international conference on computer vision, pages 2223–2232, 2017. 89