INTERNSHIP REPORT

Neuroscience-Inspired Encoding and Learning: A Path to Robust Representation Learning

Mohammed-Yassine Habibi

Supervisor :
Prof. Makoto Yamada

Machine Learning and Data Science Unit,
Okinawa Institute of Science and Technology OIST, Okinawa, Japan

Research Internship **April-August 2025**

Acknowledgement

In a first place, I want to thank Pr. Makoto Yamada who was my supervisor during this internship at the Okinawa Institute of Science and Technology (OIST). Indeed, Makoto was a very sensitive and sensible person and researcher that created a safe and stimulating environment in his lab for research. I am very glad for all I could learn both in term of scientific knowledge and human relationships at the Machine Learning and Data Science Unit and I am looking forward to continue to collaborate with him.

The second person who was very helpful, and whom I want to thank in particular, is Ms. Chikako Sugiyama. She was the first to welcome me to this new place I didn't know, and she guided me throughout the internship, helping to facilitate all kinds of procedures. Thank you for your support, your kindness, and your patience—it truly made a difference in my experience.

Finally, in this unit, I had the chance to share with a lot of other smart and interesting students from different backgrounds on a daily basis. I learnt from them a lot about new research fields I didn't know, and I had the chance to form friendships and explore Okinawa island with some of them. I am thinking about Cléa, Kaoru, Eiji, Rémi, Klea, Vedant, Keyu, Antoine and Félix, all of whom made the experience enjoyable and memorable. I want also to specifically acknowledge Klea for trusting me and helping me in my project during the last weeks.

Of course, I would also like to express my gratitude to the various staff members of the university campus who make this place enjoyable and easy to live in every day, and who provide invaluable support to the students. This includes the graduate school staff, the cafeteria workers, and the gardeners.

Abstract

In this work, we focus on neuroscience-inspired strategies to improve the robustness of encoder models. Specifically, we consider the Artificial Kuramoto Oscillatory Neurons (AKOrN, Miyato et al. 2025), an encoder that models neurons as interacting oscillators, and PhiNet (Ishikawa et al. 2025), a non-contrastive self-supervised learning model inspired by the hippocampus and temporal prediction theory (Y. Chen et al. 2024).

We combine the advantages of the AKOrN model with several self-supervised pre-training strategies and evaluate them on the CIFAR-10 test set under the adversarial robustness benchmark AutoAttack.

Our experiments highlight two key factors for reaching state-of-the-art robust accuracy with this approach: (i) increasing the number of pre-training epochs, and (ii) enlarging the convolutional connectivity size of the AKOrN model. Building on these insights, we further show that pre-training AKOrN with PhiNet or SimSiam yields significant gains, achieving up to 75.03% robust accuracy, which is competitive with the state of the art. Moreover, our method displays a significant advantage in not relying on adversarial training or synthetic data augmentation, while still achieving similar performance to methods that do.

These findings underscore the potential of neuroscience-motivated models for making representation learning both more robust and more general, and offers a promising direction for future work.

Contents

Introduction

With the rapidly evolving landscape of deep learning, advancements have frequently been driven by inspirations taken from the complex behavior of the human brain. Groundbreaking developments, like Convolutional and Recurrent Neural Networks (LeCun et al. 1998; Elman 1990; Hopfield 1982) have conceptual precursors in neuroscience, emulating how biological neurons receive and interpret outside stimuli. This long-standing influence continues to drive the research field, leading to investigations of new architectures that depart from classical models of the neuron. At the same time, the approach to self-supervised learning (SSL) has become widespread, providing a powerful alternate to supervised learning by training models on vast, unlabeled datasets without the need for human-supplied labels (T. Chen et al. 2020; X. Chen et al. 2021; Grill et al. 2020). SSL seeks to learn data representations that are robust and generalizable by using pretext tasks, acting as a proxy supervisory signal for learning of features.

This research explores the intersection of neuroscience-inspired encoders and self-supervised pre-training paradigms to enhance adversarial robustness. Our key objective is to leverage the unique properties of these models to achieve state-of-the-art performance in learning representations resilient to perturbations.

In order to accomplish this, we will, in the first instance, provide an overview of the landscape of self-supervised learning, both generative as well as discriminative, with particular attention to contrastive as well as non-contrastive models such as SimCLR, BYOL, and SimSiam. Then, we introduce two leading models that are inspired by neuroscience: the Artificial Kuramoto Oscillatory Neurons (AKOrN) (Miyato et al. 2025) as an encoder, viewing neurons as oscillations that are interacting, as well as PhiNet (Ishikawa et al. 2025), a pre-training model that is founded on neuroscientific theories of vision and memory. Then, we detail the problem of adversarial robustness and the standard metrics that are used for evaluating it (Croce, Andriushchenko, et al. 2020; Croce and Hein 2020a). Finally, we present a series of experiments evaluating the AKOrN encoder combined with various SSL models, achieving state-of-the-art robust accuracy on the CIFAR-10 test benchmark.

1 Self-supervised learning

1.1 Context

In opposition to supervised learning, the self-supervised learning (SSL) paradigm **doesn't rely on labels** provided by humans but only on the data itself to train a model. It is very commonly used for pre-training as it enables one to leverage large unlabeled datasets for running annotation-free pretext tasks. Those tasks usually aim at extracting useful features and learning a general representation from the data.

If those annotation-free pretext tasks can be very diverse, they share the fact that they provide a surrogate supervision signal for feature learning. We will present the most common approaches in self-supervised learning on this first part. There are basically two categories of methods : the generative approaches and the instance discrimination approaches.

1.1.1 Generative Models

The generative models learn to **reconstruct original images** that are modified, corrupted, or only partially observed. It consists of **pixel-level generation** (generation in the input space) and is therefore computationally very expensive. Those models usually rely on autoencoder (AE) architectures (Hinton et al. 2006) that are composed of an encoder f and a decoder g and are trained to minimize a reconstruction error of the following form $\|x - g(f(x))\|_2$. This is equivalent to maximizing a lower bound on the mutual information between the image x and its representation $f(x)$ (Vincent, Larochelle, Lajoie, et al. 2010).

The dimension of $f(x)$ is often chosen smaller than x, in order to avoid learning a trivial identity encoder and decoder. The representation corresponds to the **bottleneck of the information**. As shown below, different methods can be used to corrupt the inputted image that we want to reconstruct (generate).

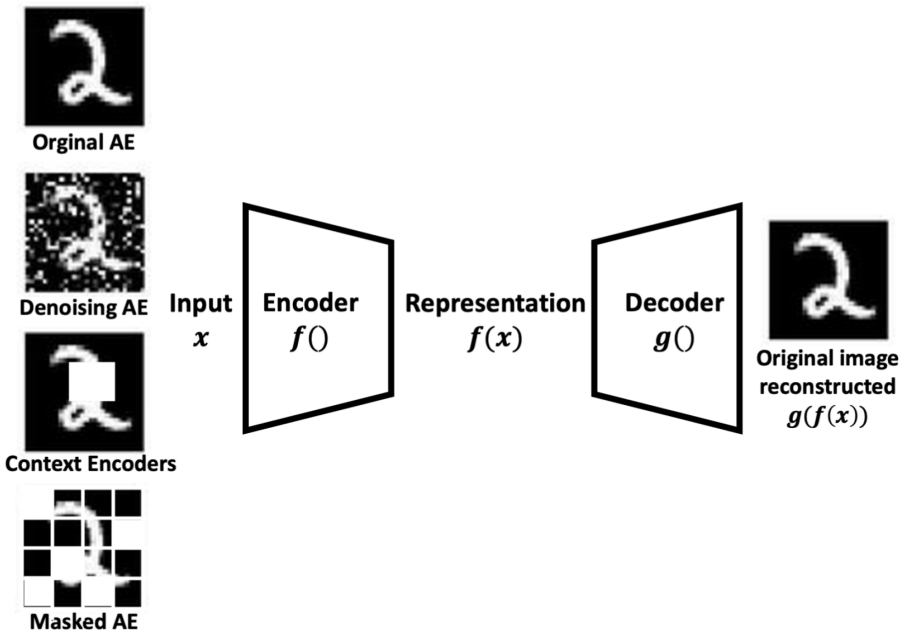

Figure 1: Denoising-based autoencoders (Gori 2025)

Among the most famous methods we can mention the following ones.

- **Denoising AE** (Vincent, Larochelle, Bengio, et al. 2008): The model learns to reconstruct an image from a version with noise (gaussian noise, for example). But this task can be done without learning a good representation of images as no semantic information is needed for this task. Therefore it is not the most efficient for representation learning.
- **Context Encoder** (Pathak et al. 2016): In this approach, a large-pixel region of the input image is masked and tried to be reconstructed. Being able to solve this second task implies this time to have a representation that encapsulates semantic information about the image. However, this task is difficult to evaluate and it is difficult to generalize to out-of-distribution data.
- **Masked AE** (He et al. 2022): This last approach, especially fitted to ViT models, consists in masking some patches of the image and train a decoder to predict the missing patches from the representation of the non-missing ones.

In general, the generative approach is very computationally expensive and not necessarily the most efficient for representation learning. As a consequence, we will more focus on the second approach that we are going to indroduce which is the discriminative approach.

1.1.2 Discriminative Models

The second kind of methods for learning representations is the discriminative approach. This approach is comparable to a supervised classification approach in the sense that it uses similar objective functions to separate instances. However, in the self-supervised framework, both the input and labels are derived from an unlabeled dataset. In addition, the **SSL model** doesn't separate classes but directly **separate single instances (images)**.

Discriminative pretext tasks have been designed using different heuristics. One of the recent ideas was, for example, teaching a model to count the number of visual primitives of an image. (Noroozi et al. 2017). Taking advantage of the fact that this count should be **independent of any geometric transformation** applied to an image and should decrease when we take a crop of the image, we can hope to learn a useful representation f. Those kinds of invariance properties are often used when defining a pretext task in discriminative models.

In practice, a **Siamese architecture** is used to leverage the invariance property and train such an encoder. Those architectures were first proposed by Becker et al. 1992 and Bromley et al. 1993. It consists of having **two parallel streams with the same encoder** f and two different inputs x_1 and x_2 for each of these streams. The idea is then to apply a loss on the outputs and maximize or minimize the agreement between the output (depending on the initial inputs).

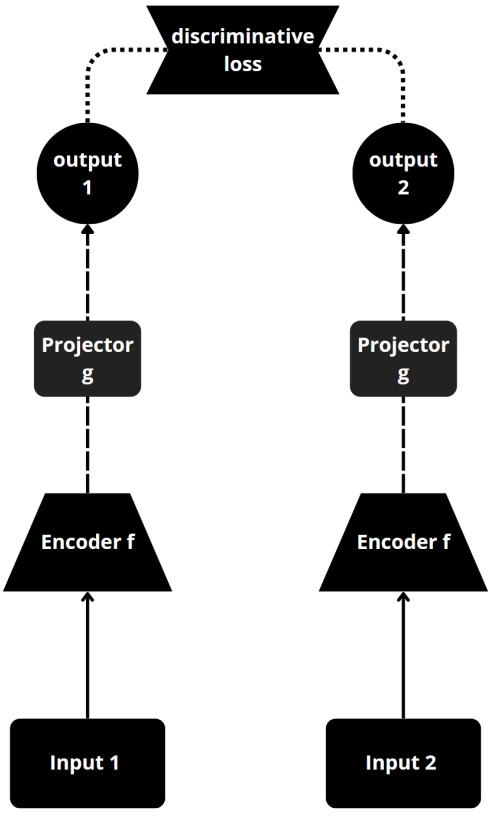

Figure 2: Siamese architecture

A very popular kind of approach that has been used with Siamese-like architectures is based on what people call **contrastive learning** and requires the use of a **pairwise contrastive loss**. Let's see in detail what contrastive learning stands for and what are the advantages and limitations of this approach for self-supervised learning.

1.2 Contrastive Learning

Contrastive learning frameworks try to achieve two different objectives at the same time for the learned encoder f :

- If $\mathbf{x_1}$ and $\mathbf{x_2}$ are two semantically similar inputs, then we want that $f(\mathbf{x_1}) \approx f(\mathbf{x_2})$


Figure 3: Positive pair example (semantically similar inputs)

- If $\mathbf{x_1}$ and $\mathbf{x_2}$ are semantically different inputs, then we want that $f(\mathbf{x_1}) \neq f(\mathbf{x_2})$


Figure 4: Negative pair example (semantically different inputs)

The models that follow this kind of discriminative approach have achieved strong performances and have been very well studied, as they have proven promising. We can distinguish two different contrastive learning approaches that consist of two different mathematical reformulations of the objectives for the encoder :

- Geometric approach: based on a distance between $f(\mathbf{x_1})$ and $f(\mathbf{x_2})$.
- Information theory approach: based on a statistical dependence measure such as Mutual information between $f(\mathbf{x_1})$ and $f(\mathbf{x_2})$.

We will focus on presenting the geometric approach as it is more meaningful for understanding our work and the methods we will focus on. The **geometric approach** is appreciated for its **easy formalism** and because it doesn't require **strong hypothesis** like the negative samples independence contrary to the information theory approach.

When working with this approach, we want to have positive (or similar) pairs as well as negative (or different) pairs of samples for training. In practice, we will usually define a positive pair (x, x_i^+) by

taking for x_i^+ a transformation (augmentation) of the anchor x or a nearest neighbor of the sample x. If we were in a supervised setting, we would simply take pairs of samples in the same class (with the same labels).

To define a negative pair (x, x_j^-), we typically select randomly two different images from the dataset. However, these images may still belong to the same latent class. While this approach is simple, it often **requires to use large batch sizes** to increase the likelihood of sampling informative and diverse negative pairs (T. Chen et al. 2020).

1.2.1 SimCLR

SimCLR (T. Chen et al. 2020) is a simple framework for contrastive learning of visual representations based on a Siamese architecture that achieved in 2020 a new state-of-the-art in self-supervised learning on ImageNet ILSVRC-2012 (Russakovsky et al. 2015). Among other things, they have highlighted the utility of a non-linear projector in the Siamese architecture, the importance of strong data augmentation for unsupervised contrastive learning, as well as the benefits of longer training and larger batch sizes, which are specific to contrastive learning.

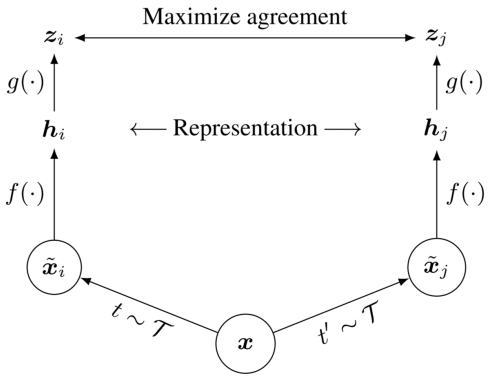

Figure 5: SimCLR (T. Chen et al. 2020)

The positive pairs of samples here are defined as augmentations $(\tilde{x}_i, \tilde{x}_j)$ of the same initial image x, and the negative pairs are pairs of augmented images $(\tilde{x}_i, \tilde{y}_j)$ that originate from two different images x and y.

During the training, minibatches of N examples are randomly sampled and each sample is augmented in two different ways so that we have 2N data points. Then the loss function used is the normalized temperature-scaled cross entropy loss, which is defined in the following way for one positive pair (i, j) :

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

with $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T\mathbf{v}/\|\mathbf{u}\|\|\mathbf{v}\|$ the cosine similarity and τ being a temperature hyperparameter. The final loss is an average over all the positive pairs of the minibatch. The use of a **temperature parameter** in this loss plays a role in controlling how much the loss **emphasizes on the hard negative samples** (ie. negative pairs of very similar images).

1.3 Non-Contrastive Learning

In **non-contrastive learning**, we want to achieve the same goal as in contrastive learning but **without having to use any negative pair of samples** but only positive. This enables the use of smaller batch sizes and eliminates concerns related to false or hard negatives. However, one cannot simply use the same model architectures with the same optimization procedures as for contrastive learning. Indeed, if one does so, there is a very high probability that the model will **collapse** and learn a poor encoder that outputs a constant value independently of the input, as it is enough to respect the constraint on positive pairs.

Let's see what are the different tricks that are used in some famous non-contrastive learning algorithms to avoid collapse.

1.3.1 BYOL

Bootstrap your own latent (BYOL, Grill et al. 2020) is the name of a method that uses for the first time **self-distillation as a way to avoid collapse** in non contrastive learning with Siamese networks. BYOL uses two encoders with different weights : one is called the online network and the other the target network. The general idea of BYOL is to use the target encoder to train an enhanced online encoder. The novelty of this method lies on the way the weights of the online and the target networks are updated, which enables to avoid collapsing and to have state-of-the-art results. During training, the target network provides a target representation that the online network tries to predict. The weights of the online network are updated by backpropagation to predict the output of the target network while the **stop-gradient** operator $sg(\cdot)$ prevent the backpropagation of the gradient to the target encoder. The target network is randomly initialized and is (only) being updated after each backpropagation on the online network as a slowly moving exponential average of the online network:

$$\xi \leftarrow \beta\xi + (1-\beta)\theta \ , \quad (\text{with } \beta \in [0,1])$$

Therefore, the target and the online networks interact and learn from each other. The use of a **slow-moving average target network** in BYOL to produce stable targets for the online network has been inspired by deep RL (Mnih et al. 2015) and allows to have smoother changes in the target representation.

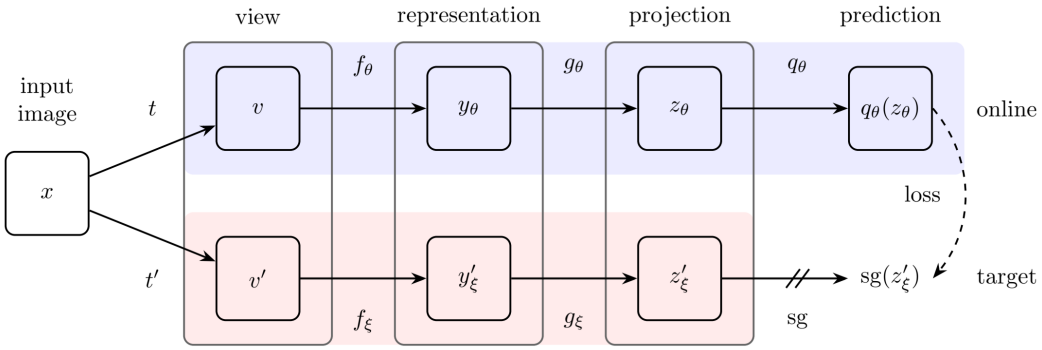

Figure 6: BYOL (Grill et al. 2020)

The loss used is a symmetrized mean squared between the normalized predictions and target projections which can be simplified into a symmetrized negative cosine similarty

$$\mathcal{L}_{BYOL} = -\frac{q_\theta(z_\theta)\cdot sg(z'_\xi)}{\|q_\theta(z_\theta)\|_2\|sg(z'_\xi)\|_2} - \frac{q_\theta(z'_\theta)\cdot sg(z_\xi)}{\|q_\theta(z'_\theta)\|_2\|sg(z_\xi)\|_2}$$

1.3.2 SimSiam

Later one, a simpler and yet competitive non-contrastive method was proposed by X. Chen et al. 2021. This new model doesn't use a slow-moving average trick for updating the weight of a target encoder. Unlike BYOL, SimSiam uses a Siamese architecture with the two **encoders sharing the same weights on the two streams**. However, only one of the two streams is subject to gradient-based optimization (the other side having a stop-gradient preventing the gradient for back-propagating). Moreover, the architecture itself is also simpler than the one of BYOL as there are only the encoder f for one stream and the same encoder f followed by a predictor h for the other stream.

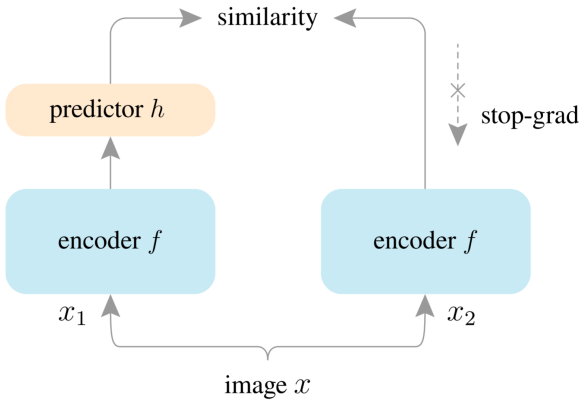

Figure 7: SimSiam (X. Chen et al. 2021)

The loss used is a symmetrized negative cosine similarity (very close to BYOL loss) defined as follow :

$$\mathcal{L}_{SimSiam} = -\frac{1}{2}\left(\frac{p_1 \cdot sg(z_2)}{\|p_1\|_2\|sg(z_2)\|_2} + \frac{p_2 \cdot sg(z_1)}{\|p_2\|_2\|sg(z_1)\|_2}\right)$$

with $p_i \coloneqq h(f(x_i))$, $z_i = f(x_i)$ and $sg(\cdot)$ being the stop-gradient.

The architectures presented here are among the most well-known and widely used for benchmarking self-supervised learning tasks or for pretraining encoders. In our experiments, we will consistently use these architectures as baselines to facilitate comparison and interpretation of our results.

In the following section, we will present our main focus during our research which was to **combine novel and promising brain-inspired methods** to improve the state-of-the-art performances of representation learning algorithms. Our focus in the next section can be divided in two parts :

- The encoder : First, we will present a brain-inspired encoder that implements dynamic (spatiotemporal) representations and that already provided improvements across a wide variety of tasks - the Artificial Kuramoto Oscillatory Neurons (**akorn**).
- The pretraining architecture : Then, we will present a brain-inspired non-contrastive learning architecture that has been developed based on a recent promising neuroscientific theory to explain long-term memory handling : PhiNets (Ishikawa et al. 2025).

2 Neuroscience-inspired models

Many advances in deep learning have been inspired by our understanding of brain functioning, particularly how neurons receive, process, and interpret external stimuli. Among the most famous, we can cite Convolutional Neural Networks (LeCun et al. 1998) and Recurrent Neural Networks (Elman 1990; Hopfield 1982). This partly explains why recent discoveries in neuroscience have garnered significant interest within the AI research community. A growing number of new studies explore novel neural network architectures that incorporate **alternative forms of information flow between neurons**. In this section, we present two such architectures that stood out for their originality and promising results. Those works will be the main foundation of our research and results that we will present in Section 4.

2.1 AKOrN

In their recent article in which they introduce the Artificial Kuramoto Oscillatory Neurons (AKOrN), Miyato et al. 2025 ask whether there are, today, any lessons from recent neuroscience discoveries that could be applied as design principles for artificial neural networks. They focus their interest on a modern **dynamical view of neurons**, where neurons are **oscillatory units** that interact with each other. Being able to reproduce this in an artificial neural network would make it possible to model the lateral connections that exist between neurons and that have been recognized early by neuroscientists (Hubel et al. 1962; Somers et al. 1995). In fact, interesting phenomena such as clustering of neuronal activity in the brain and competition between different clusters to explain the input signal have been described. The last phenomenon naturally creates an information bottleneck.

To encode and represent this lateral binding between neurons seen as oscillators, Miyato et al. 2025 build on an N-dimensional generalization of the Kuramoto model. This is a model commonly used to describe oscillators synchronization, and is closely linked to **oscillatory neural network models** that will be presented in the following. This implementation enabled performance improvements across several tasks, and in particular in **adversarial robustness**, which is our main focus.

2.1.1 The Kuramoto Model

The original Kuramoto model describes the dynamics of a system of coupled oscillators, each represented by a phase parameter $\theta_i \in [0, 2\pi)$. The time evolution of each oscillator is governed by the following system of differential equations:

$$\dot{\theta}_i = \dot{\omega}_i + \sum_j J_{ij} \sin(\theta_j - \theta_i)$$

where $\omega_i \in \mathbb{R}$ is the natural frequency of the oscillator i, and $J_{ij} \in \mathbb{R}$ is the coupling strength between oscillators i and j. A positive J_{ij} encourages phase alignment between the two oscillators, while a negative value tends to drive them out of phase.

In the multi-dimensional vector version of the Kuramoto model, each oscillator is now represented by an N-dimensional vector of $\mathbf{S}^{N-1} \coloneqq \{\mathbf{x} \in \mathbb{R}^N,\ \|\mathbf{x}\|_2 = 1\}$. The new system of differential equations that governs the time evolution of this system of N-dimensional oscillators is the following :

$$\dot{\mathbf{x}}_i = \Omega_i \mathbf{x}_i + \mathrm{Proj}_{\mathbf{x}_i}\left(\mathbf{c}_i + \sum_j \mathbf{J}_{ij}\mathbf{x}_j\right) \quad \text{with } \mathrm{Proj}_{\mathbf{x}_i}(\mathbf{y}_i) = \mathbf{y}_i - \langle \mathbf{y}_i, x_i \rangle \mathbf{x}_i$$

The non-linear projection operator $\text{Proj}_{\mathbf{x}_i}(\cdot)$ makes sure the velocity vector of an oscillator $\mathbf{x}_i$ effectively lives in the tangent plane of $\mathbf{S}^{N-1}$ at $\mathbf{x}_i$. Its behavior is highlighted in the following figure.

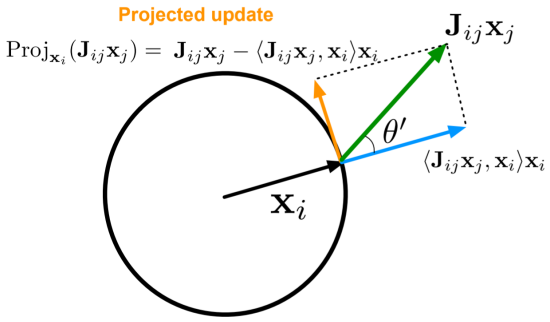

Figure 8: Kuramoto Model Updates (Miyato et al. 2025)

In the previous ODE, Ω_i is an $N \times N$ anti-symmetric matrix and $\Omega_i \mathbf{x}_i$ is the natural frequency term of the i^{th} oscillator. The second part is the result of the interaction between the oscillator i and other oscillators. It is projected on the tangent space of $\mathbf{S}^{N-1}$ at $\mathbf{x}_i$. The vector $\mathbf{c}_i$ is a conditional variable that depends on the input signal and that strongly binds $\mathbf{x}_i$ to the same direction as $\mathbf{c}_i$.

It has been shown (Aoyagi 1995) that under some simple hypothesis on $\mathbf{J}_{ij}$ and Ω_i, the Kuramoto ODE system has a Lyapunov function ensuring stability as well as the minimization of the following energy at each update : $E = -\frac{1}{2}\sum_{i,j} \mathbf{x}_i \mathbf{J}_{ij} \mathbf{x}_j - \sum_i \mathbf{c}_i^T \mathbf{x}_i$. However, in practice, it has been observed that this energy was also decreasing without binding to the symmetry hypothesis on $\mathbf{J}_{ij}$ and Ω_i.

Let's see now how Miyato et al. 2025 proposes to build a network with oscillatory neurons based on this Kuramoto model of oscillators.

2.1.2 Artificial Kuramoto Oscillatory Neurons (AKOrN)

Kuramoto layer The authors of AKOrN first define the **Kuramoto layer** that we will denote as $\mathcal{KL}$. It consists of a single layer of neurons that receives an initial image $\mathbf{X}$ and an initial conditional stimuli $\mathbf{C}$ as input. Each super-pixel $\mathbf{x}_{c,h,w}$ of $\mathbf{X}$ is an N-dimensional oscillator represented by a vector in $\mathbb{R}^N$, with c, h, and w representing the channel, height, and width positions, respectively. Each pixel $\mathbf{c}_{c,h,w}$ of $\mathbf{C}$ is the conditional stimuli used to update the state of the oscillator $\mathbf{x}_{c,h,w}$ with the Kuramoto update.

A single Kuramoto layer will apply $T \in \mathbb{N}^*$ discrete updates of the oscillator system $\mathbf{X}^{(0)}$ by applying the following Kuramoto scheme. For $t = 0, \dots, T-1$:

$$\Delta \mathbf{x}_i^{(t)} = \Omega_i \mathbf{x}_i^{(t)} + \text{Proj}_{\mathbf{x}_i^{(t)}} \left(\mathbf{c}_i + \sum_j \mathbf{J}_{ij} \mathbf{x}_j^{(t)} \right)$$
$$\mathbf{x}_i^{(t+1)} = \Pi \left(\mathbf{x}_i^{(t)} + \gamma \Delta \mathbf{x}_i^{(t)} \right)$$

where i and j are each of the form $(h, w, c) \in [H] \times [W] \times [C]$, and $\Pi : \mathbf{x} \mapsto \frac{\mathbf{x}}{\|\mathbf{x}\|_2}$, $\gamma > 0$.

The trainable parameters of the Kuramoto layer are

$$\Omega = (\Omega_i)_{i \in [H] \times [W] \times [C]} \quad \text{and} \quad \mathbf{J} = (\mathbf{J}_{ij})_{(i,j) \in ([H] \times [W] \times [C], [H'] \times [W'] \times [C'])}$$

The output of the $\mathcal{KL}$ layer is the final state of the system of oscillators after the updates : $\mathbf{X}^{(T)}$.

To create a neural network from those Kuramoto layers, one needs to stack multiple Kuramoto layers and to add a readout module between two succcessive kuramoto layers that aims at extracting features from the final oscillatory states to create a new conditional stimuli $\mathbf{C}$.

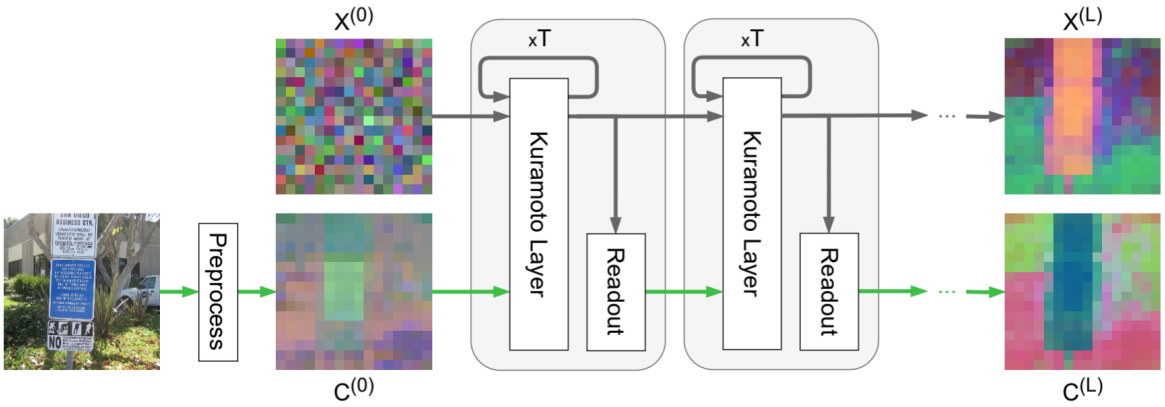

Figure 9: Artificial Kuramoto Oscillatory Network (Miyato et al. 2025)

Readout module The readout module $\mathcal{RM} : \mathbb{R}^{H\times W\times(C\times N)} \longrightarrow \mathbb{R}^{H\times W\times(C'\times N)}$ was introduced by Miyato et al. 2025 and is specifically designed to leverage the symmetries and properties of the oscillators to extract meaningful features. The oscillators are constrained to live on the unit hyper-sphere of $\mathbb{R}^N$ due to the projector operator Π. As a result, all information and patterns are encoded in the **relative directions** of the oscillators rather than their absolute positions.

Consequently, the readout module $\mathcal{RM}$ is **phase-invariant**: adding a constant phase to all oscillators does not change the output of the module. Formally, for $\mathbf{X} \in \mathbb{R}^{H\times W\times(C\times N)}$ and a rotation $R : \mathbb{R}^N \longrightarrow \mathbb{R}^N$, if we denote by

$$R(\mathbf{X}) \coloneqq (R(\mathbf{x}_i))_{i\in[H]\times[W]\times[C]}\,, \text{ and } R(\mathcal{RM}(\mathbf{X})) \coloneqq (R(\mathcal{RM}(\mathbf{X})_i))_{i\in[H]\times[W]\times[C']}$$

then the readout satisfies

$$\mathcal{RM}\big(R(\mathbf{X})\big) = R\big(\mathcal{RM}(\mathbf{X})\big).$$

This formulation ensures that $\mathcal{RM}$ depends only on the **relative orientations** of the oscillators, preserving the intrinsic structure of the data encoded on the hypersphere.

In practice, we choose the simple following readout module

$$\mathcal{RM}(\mathbf{X}) = g\left(\left(\left\|\sum_i \mathbf{U}_{ji}\mathbf{x}_i\right\|_2\right)_{j\in[H]\times[W]\times[C']\times[N]}\right)$$

with $g : \mathbb{R}^{H\times W\times(C'\times N)} \longrightarrow \mathbb{R}^{H\times W\times(C'\times N)}$ a linear layer (that can be also set to identity) and $\mathbf{U}_{ji} \in \mathbb{R}^{N\times N}$ learned weight matrices.

2.1.3 Learning connectivity parameters

Once the Artificial Kuramoto Oscillatory Network architecture is established, the next step is to learn the **connectivity parameters** $J_{ij} \in \mathbb{R}^{N \times N}$, which encode both the coupling strength and the directional influence between oscillators i and j. Each Kuramoto layer possesses its own distinct connectivity tensor. Although in principle this tensor could take any form, Miyato et al. 2025 focus on two approaches that have proven particularly effective in computer vision: **convolutional connectivity**, which leverages local convolution operators, and **attentive connectivity**, which leverages self-attention dynamic input-dependent connectivity. We show below an illustration of a convolutional connectivity which works like a convolutional kernel.

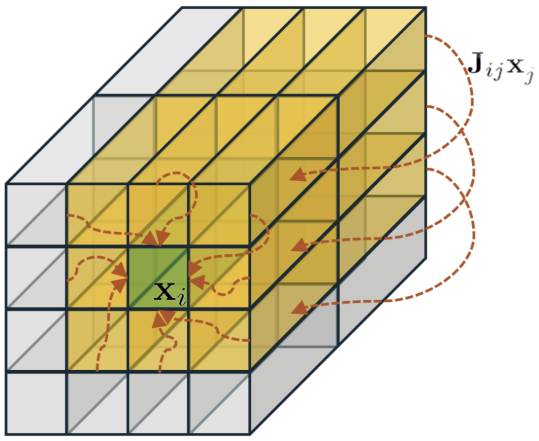

Figure 10: Visualization of a convolution connectivity

In the original article, the Artificial Kuramoto Oscillatory Network was trained in two stages for image classification: first, it was pretrained using a SimCLR framework, and then it was fine-tuned with a standard cross-entropy loss using the Adam optimizer.

In the following subsection, we present a recent brain-inspired **self-supervised learning framework, PhiNet**, by Ishikawa et al. 2025, which has demonstrated strong robustness against representational collapse and high adaptability to hyperparameter choice. Our ultimate goal is to study this brain-inspired self-supervised learning framework in conjunction with the previously introduced Artificial Kuramoto Oscillatory Neurons, and to demonstrate the mutual benefits arising from the combination of these two approaches.

2.2 PhiNet

PhiNet is a novel non-contrastive learning model proposed by Ishikawa et al. 2025, which is grounded in the temporal prediction hypothesis (Y. Chen et al. 2024) and the Complementary Learning Systems (CLS) theory (McClelland et al. 1995). These theories suggest that the hippocampus predicts sensory inputs, which are then gradually integrated into the neocortex in a structured manner. Building on these principles, several recent studies have applied similar ideas to self-supervised learning (Oord et al. 2018).

2.2.1 Implementation of the hippocampal model

Building on the SimSiam architecture, PhiNet models three distinct regions of the hippocampus along with the neocortex, capturing how they interact:

- **The entorhinal cortex (EC)** which is the main input and output cortex of the hippocampus
- **The CA3 region** which is responsible for predicting future signal
- **The CA1 region** which measures the difference between the predicted signal and its future signal.
- **The neocortex (NC)** encodes the long-term memory.

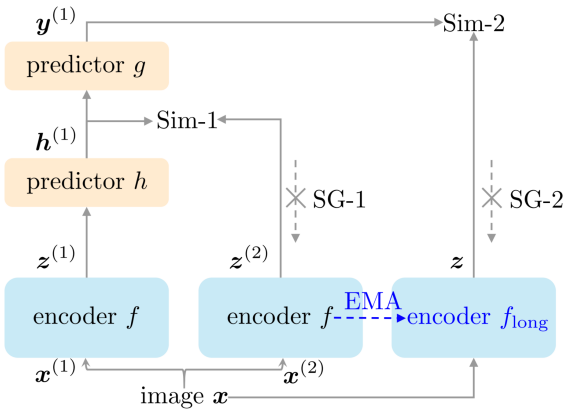

Figure 11: PhiNet (Ishikawa et al. 2025)

The PhiNet model operates as follows. A single input image x is augmented in two different ways, resulting in three inputs $x, x^{(1)}, x^{(2)}$ that are processed by three parallel streams. The outputs of these streams are then compared at different depths by computing their similarities.

In the following, we interpret the different components of this architecture in light of the hippocampal model.

EC layer The EC acts as the gateway between the cortex and the hippocampus, providing input and output pathways. The different layers of EC (**II**, **III** and **V**) have representations of the inputs of the hippocampus $x, x^{(1)}, x^{(2)}$ that correspond to $z^{(1)} = f(x^{(1)})$, $z^{(2)} = f(x^{(2)})$ and $z = f_{long}(x)$, with f being the encoder representing the cortex.

CA3 region This region is modeled by the predictor network h, implemented as a two-layer neural network with ReLU activations and batch normalization. Its objective is to predict the signal $z^{(2)}$ based on the preceding input $z^{(1)}$. Within the framework of temporal prediction theory, $z^{(2)}$ corresponds to the future state of $z^{(1)}$.

CA1 region The initial simple idea proposed by Y. Chen et al. 2024 to model the CA1 region that measures the error of prediction made by CA3 is to use an MSE loss. On the more recent PhiNet implementation, the authors propose instead to use a mixture of a symmetric negative cosine loss function denoted by Sim-1 and a predictor g. The loss measures the concordance between the predicted representation outputted by CA3, $h^{(1)}$, and the temporally distant signal $z^{(2)}$. Ultimately the predictor g enables to propagate a feedback from CA1 to the EC and eventually to the neocortex (NC) to store it in a long-term memory.

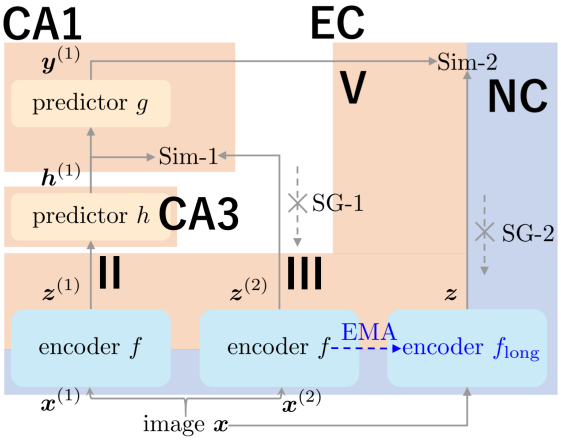

Figure 12: PhiNet interpretation with the Hippocampal Model (Ishikawa et al. 2025)

2.2.2 Model Training

The model is trained by jointly minimizing two losses : the first one, Sim-1 for the hippocampus model, and the second one, Sim-2 for the neocortex model. Those two losses are combined to update the encoder f by backpropagation.

Sim-1 This hippocampus loss is a symmetric negative cosine loss function, exactly as in the SimSiam architecture :

$$L_{\text{Cos}}(\theta) = -\frac{1}{2n}\sum_{i=1}^{n}\frac{(h_i^{(1)})^\top sg(z_i^{(2)})}{\|h_i^{(1)}\|_2\,\|sg(z_i^{(2)})\|_2} - \frac{1}{2n}\sum_{i=1}^{n}\frac{sg(z_i^{(1)})^\top h_i^{(2)}}{\|sg(z_i^{(1)})\|_2\,\|h_i^{(2)}\|_2}$$

This loss is appropriate because it becomes minimal when the predicted representation from CA3, $h^{(1)}$, is perfectly aligned with the temporally distant signal $z^{(2)}$.

Next, we examine how information transfer from the hippocampus to the neocortex is captured during the model's training.

Sim-2 In the neocortex, this loss is defined as an MSE loss between the CA1 output and the representation $z = f(x)$ from the entorhinal cortex (EC):

$$L_{\mathrm{NC}}(\theta) = \frac{1}{2n}\sum_{i=1}^{n}\left\|y_i^{(1)} - sg(z_i)\right\|_2^2 + \frac{1}{2n}\sum_{i=1}^{n}\left\|y_i^{(2)} - sg(z_i)\right\|_2^2$$

The final loss for PhiNet is the sum of Sim-1 and Sim-2. Sim-2 is considered a slow-learning loss.

Stable encoder To incorporate long-term memory, a different encoder f_{long} has been added to PhiNet's third stream. This stream is designed to model the neocortex and f_{long} corresponds to a stable encoder that further improve slow learning with an ability to maintain long-term signals. The weights of f_{long} are updated by using an exponential moving average of the model parameters of f and f_{long}. It is similar to techniques used by other self-supervised methods as BYOL architecture. If we denote by ξ and ξ_{long} the respective parameters of f and f_{long} we have the following update scheme :

$$\xi_{long} \leftarrow \beta\xi_{long} + (1-\beta)\xi$$

with $\beta \in [0, 1]$ a hyperparameter usually chosen close to 1.

The benefit of maintaining a separate encoder whose weights are updated via an exponential moving average (EMA) scheme is that it yields a representation network with more stable parameters during training, while simultaneously mitigating the risk of representation collapse.

PhiNet has demonstrated several advantages for encoder pre-training compared to other state-of-the-art architectures such as SimSiam and BYOL. Among these benefits are its **robustness to the choice of the weight decay hyperparameter** and its superior performance in online learning settings.

Having introduced self-supervised learning and the neuroscience-inspired models that guide our approach, we now turn to the task at hand. Our goal is to leverage the strengths of AKOrN and PhiNet to push beyond current state-of-the-art performance in term of robustness to adversarial attacks.

3 Adversarial robustness

A pretrained encoder can later be fine-tuned for a wide range of downstream applications. One of the most common, and a standard choice in benchmarks for evaluating the effectiveness of self-supervised frameworks, is **image classification**. Popular datasets used for this purpose include *CIFAR-10*, *CIFAR-100*, and *ImageNet*.

3.1 Motivation

While some works report only **test-set accuracies** on these benchmark datasets, this evaluation alone may **not** be **sufficient** to faithfully assess the performance of an encoder (or of a pre-training framework). Indeed, it is common for a model to achieve high test accuracy while failing to learn robust and generalizable features, effectively overfitting to the data distribution. This issue becomes apparent when the model is attacked: adversarial examples—inputs that are very close to correctly classified evaluation samples—can highlight those weakness in a model, revealing its vulnerability.

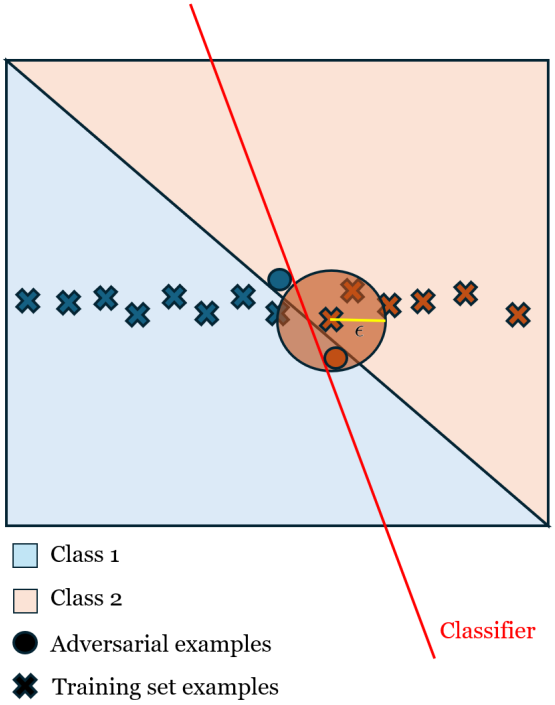

Figure 13: Adversarial perturbation illustration

Many different methods have been proposed to generate adversarial examples and reliably evaluate a model's robustness. Recently, one evaluation method has gained widespread popularity for its reliability. Known as **AutoAttack** (Croce and Hein 2020a), it is described by its authors as a "parameter-free, computationally affordable, and user-independent ensemble of attacks to test adversarial robustness." AutoAttack has been tested on more than 50 different models and is currently the standard tool used by the reference benchmark for adversarial robustness, **RobustBench** (Croce, Andriushchenko, et al. 2020).

Let's see what are the main ideas used by the AutoAttack to realize those adversarial attacks.

3.2 Adversarial attack

First, we formally define what constitutes an adversarial example for a K-class classifier $g : \mathcal{D} \subset \mathbb{R}^d \to \mathbb{R}^K$, which assigns labels according to $\text{argmax}_k\, g_k(\cdot)$. Let $x_{\text{org}} \in \mathbb{R}^d$ be a point correctly classified by g

as class c. Given a distance metric $d(\cdot, \cdot)$ and a perturbation budget $\epsilon > 0$, the feasible set of the attack at x_{org} is defined as

$$\mathcal{B}_\epsilon(x_{\text{org}}) = \{z \in \mathcal{D} \mid d(x_{\text{org}}, z) < \epsilon\}.$$

The most popular attacks rely on l_p distances $d : (x, y) \in \mathbb{R}^{2d} \longmapsto \|x - y\|_p$ with $p \in \{2, \infty\}$. This choice is made for practical reasons as it will be explained below.
An adversarial example for g at x_{org} is then any input such that

$$\arg \max_{k=1,\dots,K} g_k(z) \neq c \quad \text{and} \quad z \in \mathcal{B}_\epsilon(x_{\text{org}}).$$

Intuitively, it is an input that is almost indistinguishable from x_{org} to a human observer, but is misclassified by the model.
To find an adversarial example z, it is common to solve a constrained optimization problem

$$\max_{z \in \mathcal{D}} L(g(z), c) \quad \text{such that} \quad d(x_{\text{orig}}, z) \leq \epsilon, \; z \in \mathcal{D}. \tag{1}$$

where L is a function that enforces z into not being assigned to class c.
Let's explore a popular algorithm used in adversarial attack to solve this kind of problem.

3.2.1 Auto-PGD

In **PGD-attack** (Kurakin et al. 2017), one of the most popular white-box attack, used by AutoAttack, the function L used is the cross-entropy $-\log(q_c(g(z))$ where $q_c(g(z))$ is the predicted probability assigned to the correct class c.

To solve this optimization problem (1), Croce and Hein 2020a introduced **Auto-PGD**, a variant of the Projected Gradient Descet (PGD) algorithm that solves issues one may face while using PGD for solving this optimization problem.

Let's recall first the regular PGD algorithm. For a given $f : \mathbb{R}^d \longrightarrow \mathbb{R}$, $\mathcal{S} \subset \mathbb{R}^d$ and the problem $\max_{x \in \mathcal{S}} f(x)$, the PGD is an iterative algorithm. For $k = 1, \dots, N_{iter}$,

$$x^{(k+1)} = P_{\mathcal{S}} \left(x^{(k)} + \eta^{(k)} \nabla f(x^{(k)}) \right),$$

where $\eta^{(k)}$ is the step size at iteration k and $P_{\mathcal{S}}$ is the projection onto $\mathcal{S}$ and the initial point $x^{(0)}$ is either x_{org} or $x_{\text{org}} + \zeta$ with ζ a random vector such that $(x_{\text{org}} + \zeta) \in \mathcal{S}$. We can notice that Problem 1 can be effectively solved for $d(\cdot, \cdot)$ being an l_2 or an l_∞ distance.

In the original PGD attack, the step size is fixed, i.e. $\eta^{(k)} = \eta$ for every iteration k. This choice is suboptimal and does not guarantee convergence. In contrast, Auto-PGD divides the total number of iterations N_{iter} into an **exploration phase** and an **exploitation phase**, during which the step size η is adaptively updated according to different criteria.

Let's present the ideas of Auto-PGD in four steps.

Gradient step Auto-PGD updates use a step size $\eta^{(k)}$ at iteration k and incorporates a momentum term, which is absent in the original PGD algorithm. The initial steps of APGD are typically large, so the momentum helps to regularize the updates by incorporating information from previous iterations, smoothing the optimization trajectory. The update step corresponds to the following :

$$z^{(k+1)} = P_S\left(x^{(k)} + \eta^{(k)}\nabla f(x^{(k)})\right)$$
$$x^{(k+1)} = P_S\left(x^{(k)} + \alpha \cdot (z^{(k+1)} - x^{(k)}) + (1-\alpha)\cdot(x^{(k)} - x^{(k-1)})\right)$$

where $\alpha \in [0,1]$ regulates the influence of the momentum term. $= 0.75$ is used in practice.

Step size selection The initial step size at iteration 0 is chosen such that $\eta^{(0)} = 2\epsilon$. Then the algorithm will decide weather or not the step size will be halved at fixed chosen checkpoints $w_0 = 0 < w_1, \cdots < w_n < N_{\text{iter}}$. To update the step size, one of the two following conditions has to be true:

1. $\sum_{i=w_{j-1}}^{w_j - 1} \mathbf{1}_{f(x^{(i+1)}) > f(x^{(i)})} < \rho \cdot (w_j - w_{j-1})$,
2. $\eta^{(w_{j-1})} \equiv \eta^{(w_j)}$ and $f_{\max}^{(w_{j-1})} \equiv f_{\max}^{(w_j)}$,

- *1. Insufficient Improvement:* The step size is halved if the proportion of iterations that increased the objective function f since the prior checkpoint, w_{j-1}, is below a threshold ρ (typically $\rho = 0.75$).
- *2. Stagnation and Cycling Prevention:* The step size is also halved if it remained unchanged at the previous checkpoint (w_{j-1}) **and** the maximum value of the objective function f has not improved since that checkpoint. This rule helps the algorithm escape cycles.

Checkpoints restart At each checkpoint w_j where the step size has been halved, the algorithm restarts from the best point found so far. It means in that case that we fix $x^{w_j+1} := x_{\max}$ with $x_{\max} := \text{argmax}_{k \leq w_j} f(x^{(k)})$. The idea is to refine the research of an adversarial example around the neighborhood of the current best candidate solution.

Exploration vs Exploitation The choice of the checkpoints w_j at which the step size can be reduced is important to define a transition between an initial exploration phase exploring the whole feasible set $\mathcal{S}$ and an exploitation phase where the step size is reduced more often leading to more frequent improvements of the objective but with smaller magnitude.

The checkpoints are defined as follow:

$$w_j := \lceil p_j N_{\text{iter}} \rceil \leq N_{\text{iter}}$$

with $p_j \in [0,1]$ defined as

$$\begin{cases} p_0 = 0 \\ p_1 = 0.22 \\ p_{j+1} = p_j + \max\{p_j - p_{j-1} - 0.03, 0.06\} \end{cases}$$

The only remaining free parameter in the implementation of AutoAttack is the budget N_{iter}. By construction, the resulting checkpoints are increasingly dense as the iterations progress.

In conclusion, the ensemble AutoAttack consists in combination of two parameter-free versions of PGD, APGD with cross-entropy loss, APGD with Difference of Logits Ratio Loss (cf Supplementary Materials), and two existing complementary attacks, FAB (Croce and Hein 2020b) and Square Attack (Andriushchenko et al. 2020).

The **adversarial accuracy** of a model is obtained by evaluating it under an adversarial attack. It corresponds to the number of correctly classified samples that cannot be successfully perturbed into adversarial examples by the attacker (here, AutoAttack) divided by the number of samples in total. It is this adversarial accuracy that serves as the principal metric highlighted in robustness benchmarks for comparing model robustness.

In the next part, we will focus on the current state of the research in adversarial accuracy. We will rely on RobustBench (Croce, Andriushchenko, et al. 2020), a standardized online benchmark for adversarial robustness.

3.3 Robust Accuracy Benchmark

RobustBench evaluates classifiers robustness using AutoAttack and gathers the results into leaderbords. The classifiers that appear on this leaderboard need to verify the following constraints:

- Not having zero gradients with respect to the input.
- Not being a randomized classifier.
- Not using an optimization loop at inference time.

Indeed, it has been shown that models that do not verify those constraints may make gradient-based attack harder while not being really robust. The evaluation can be conducted on three datasets: *CIFAR-10*, *CIFAR-100*, and *ImageNet*. For each dataset, robustness is measured under AutoAttack with the l_∞ norm and a perturbation budget of $\epsilon = 8/255$. In addition, for *CIFAR-10*, a separate leaderboard is available, which is based on AutoAttack with the l_2 norm and a perturbation budget of $\epsilon = 0.5$.

In the following, we restrict our attention to the leaderboard reporting robust accuracy under l_∞ attacks on the *CIFAR-10* dataset.

The current **best adversarial accuracy** reported by RobustBench is **75.28**%, achieved by Amini et al. 2024. Moreover, this high robustness accuracy is achieved thanks to a post-processing of an adversarially trained model (ie. trained using adversarial examples in the training dataset). This SOTA method integrates sparsity around the feature mean value (μ_a) of a feature vector $a^{(in)}$ before the activation function:

$$a^{(\text{out})} = \begin{cases} \mu_a & \text{if } |a^{(\text{out})} - \mu_a| \leq Th \\ a^{(\text{in})} & \text{if } |a^{(\text{out})} - \mu_a| > Th \end{cases}$$

where Th is a threshold chosen during the experiment. This sparsification around the mean is done in order to block non-robust features.

Similarly to other methods that reach a high robust accuracy on RobustBench, MeanSparse method relies on post-trained models that use additional synthetic data.

We aim to explore methods that **rely solely on pre-training** of brain-inspired models to achieve state-of-the-art robust accuracy. Specifically, we show that combining Artificial Kuramoto Oscillatory Neurons with an appropriate pre-training architecture can achieve competitive robust performance on this benchmark.

4 Experiments

In Section 1, we introduced different pre-training frameworks. We then explored in Section 2 two neuroscience-inspired approaches: the Artificial Kuramoto Oscillatory Networks (AKOrNs), an encoder that leverages oscillatory neuronal representations to achieve state-of-the-art results on multiple downstream tasks, and PhiNet, a pre-training framework based on a recent hippocampus model, which we highlight for its robustness to hyperparameter variations. Finally, in Section 3, we discussed the concept of adversarial robustness and the methodology used to evaluate the robustness of a model.

In the following section, we will present the experiments conducted, where we combine insights from self-supervised learning with brain-inspired encoders in order to achieve state-of-the-art robust accuracy on *CIFAR-10*.

4.1 Baseline Results from Prior Work

The best AutoAttack robust accuracy on CIFAR-10 listed on RobustBench is **75.28%**, achieved by Amini et al. 2024. Matching such performance using only AKOrN combined with a pre-training framework would represent a significant result. Indeed, the state-of-the-art performance reported in Amini et al. 2024 relies on **additional synthetic data, adversarial training, and post-training procedures**, whereas our methods do not make use of these tricks.

The table below reports robust accuracy results for the AKOrN encoder with SimCLR pre-training. We show both the results reported by the AKOrN authors and the results we reproduced using the official GitHub code (which differs slightly from the paper's implementation). While the two sets of results are not identical, they still provide a reliable indication of AKOrN's performance, even without optimizing the pre-training stage.

Table 1: Initial results of Adversarial Attack (AutoAttack) Robustness on CIFAR10 with AKOrN encoder

Type	pre-training Method	Clean Acc	Robust Acc
AKOrN paper claim[1]	SimCLR (50 epochs)	88.91	58.91
Github experiment [2]	No pre-training	83.24	62.66

Miyato et al. 2025 has been able to reach this robust accuracy (around 60%) by carefully selecting the hyperparameters for AKOrN.

4.2 Improving AKOrN robust accuracy

In our first designed experiment, we used the same AKOrN encoder but replaced SimCLR **pre-training with PhiNet**. This already yielded promising results, achieving a **robust accuracy** of **68.30**%. However, it came at the cost of a significant **drop in clean accuracy**, which fell to **78.05**%. To address this, we conducted further experiments with different hyperparameters to continue improving robust accuracy while mitigating the loss in clean accuracy.

For our next experiments, we used the AKOrN encoder with hyperparameters close to those reported to achieve the best robust accuracy in the literature, but with a **different number of channels** for the

[1] AKOrN paper
[2] AKOrNGitHub repo

convolution layers. Specifically, our encoder now consists of three Kuramoto layers with convolutional kernels of sizes 9, 7, and 5 from shallow to deep, $T = 3$, $N = 2$ (oscillator dimension), and 128 channels. The model was trained using the Adam optimizer with zero weight decay for both pre-training and fine-tuning. The training was performed solely on the *CIFAR-10* dataset, without any additional data.

We compared the results obtained using four different pre-training architectures (SimCLR, BYOL, SimSiam and PhiNet), each run for an initial 200 epochs and finetuned for 400 epochs. This corresponds to the same number of finetuning steps as in the original AKOrN article, but more pre-training steps.

Table 2: Adversarial Attack (AutoAttack) Robustness on CIFAR10 with AKoRN encoder (**128 channels**, **pre-training : 200 epochs**, finetuning : 400 epochs)

Type	Method	Clean Acc	Robust Acc
Contrastive	SimCLR (bs=256)	81.55	53.54
Non-Contrastive	BYOL	82.71	58.57
	SimSiam	84.22	**68.89**
	Phinet	**84.36**	68.79

While SimCLR seems to perform worse with this bigger AKOrN model, **SimSiam and PhiNet** pre-training allow to reach **robust accuracies approximately 6% higher** than those previously achieved with AKOrN while maintaining AKOrN's initial clean accuracy around 84%.

Witnessing those encouraging results, we tried to improve even more the number of pre-training steps, from 200 to 400 epochs, to see if that would make the encoder even more robust.

Table 3: Adversarial Attack (AutoAttack) Robustness on CIFAR10 with AKOrN encoder (**128 channels**, **pre-training : 400 epochs**, finetuning : 400 epochs)

Type	Method	Clean Acc	Robust Acc
Contrastive	SimCLR (bs=256)	81.46	70.76
Non-Contrastive	BYOL	76.05	46.15
	SimSiam	**83.50**	74.80
	Phinet	83.00	**75.03**

By doubling the number of pre-training steps, we observed that clean accuracies remained largely unchanged, while robust accuracies varied significantly. This emphasizes the importance of measuring robust accuracy and the fact that regular (clean) accuracy is not a robust evaluation of a model. The best **training method with PhiNet** achieved a **robust accuracy** of **75.03**%, which is very close to the state-of-the-art robust accuracy reported by Amini et al. 2024. Once again, SimSiam and PhiNet pre-trainings yielded the best results; however, we also observed a substantial improvement in robust accuracy, and reached 70.76% when using SimCLR pre-training for this increased number of epochs.

To summarize, we identified **two key factors that significantly improved AKOrN's robust accuracy**, allowing it to reach state-of-the-art performance:

- Increasing the representational capacity of the AKOrN encoder by **raising the number of channels in the convolutional connectivities of the Kuramoto layers**.
- More effectively leverage existing pretraining architectures, combined with **longer pretraining periods**.

4.3 Stabilize and improve training

Studying the training behavior of our AKOrN encoder can provide useful insights into how the model learns and how its performance might be improved. To this end, we monitored both the gradient norm and the loss during training, and made an interesting observation.

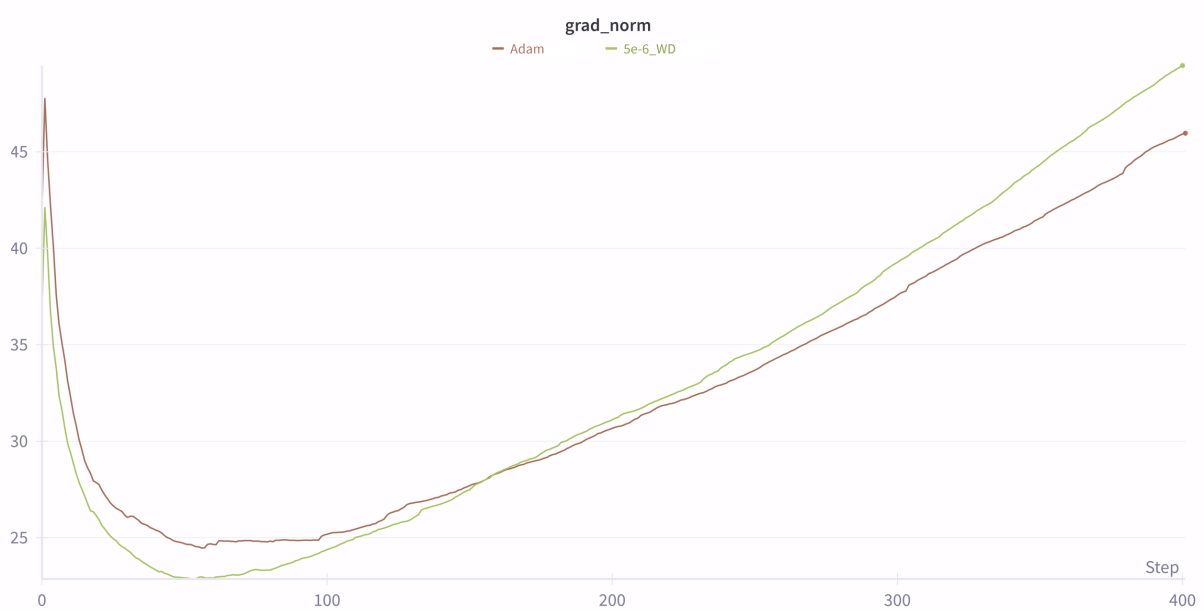

Figure 14: Gradients norm during AKOrN pretraining with PhiNet (with and without weight decay)

We observe that the gradient norm does not decrease during pretraining; instead, it grows steadily. This behavior is unusual, as it typically indicates difficulties in convergence. In our case, this is somewhat expected since we did not apply weight decay in order to maximize model performance. Nevertheless, it remains important to balance performance with training stability. To address this, we experimented with **gradient clipping**, which constrained the gradient norm and improved the optimization dynamics, leading to a lower pretraining loss.

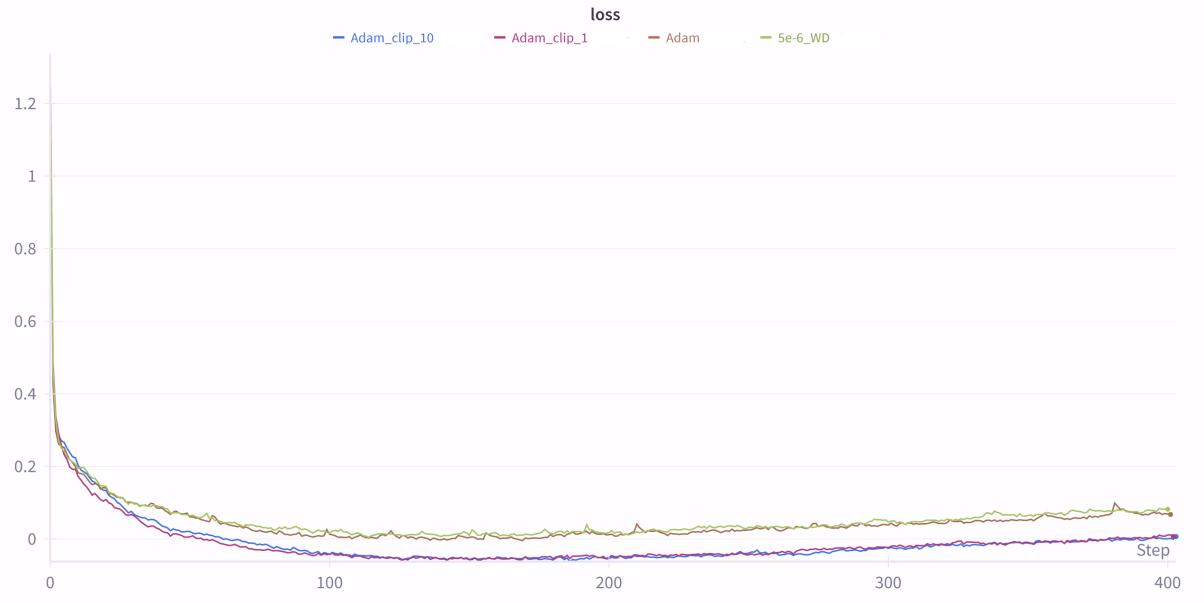

Figure 15: Loss value during AKOrN pretraining with PhiNet

We tested two different thresholds for gradient clipping, namely 1 and 10. These values were chosen based on the magnitude of the unclipped gradients so as not to completely suppress the learning signal, but rather to improve training stability while still allowing meaningful updates.

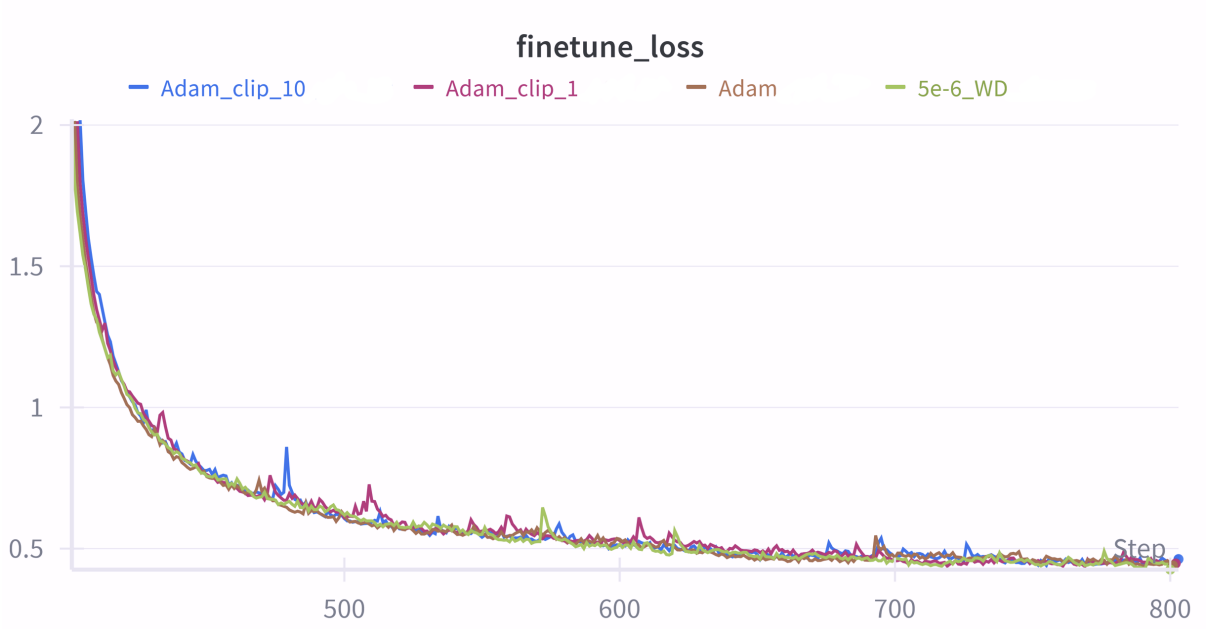

Figure 16: Models Loss during training

We can see here that, during fine-tuning, the AKOrN model pre-training with gradient clipping reaches a loss similar to that of the regular one. This suggests that while gradient clipping helps stabilize training and reduce pre-training loss, its impact on final accuracy remains limited. However, the approach may still be valuable when scaling to larger datasets or longer training schedules, where exploding gradients could otherwise have a stronger negative effect.

Discussion and Conclusion

In this report, we explored the integration of neuroscience-inspired models with self-supervised learning frameworks to enhance adversarial robustness. Recent discoveries in neuroscience have inspired novel approaches to encoding information flow in AI architectures, offering notable advantages and promising perspectives. Building on these insights, we investigated the combination of such biologically inspired mechanisms with established self-supervised learning algorithms as a strategy to develop models that are more resilient to small perturbations.

Our experiments demonstrated that pairing the Artificial Kuramoto Oscillatory Neurons (AKOrN) encoder (Miyato et al. 2025) with suitable pre-training architectures as PhiNet (Ishikawa et al. 2025) or even SimSiam (X. Chen et al. 2021) can yield significant gains in robust accuracy on the CIFAR-10 dataset.

In addition to combining the Kuramoto model inspired neural network to non-contrastive learning methods, we discovered 2 parameters that were essential to be able to reproduce the SOTA robust accuracy with AKOrN encoder and without using any post-training. Firstly, we increased the representational capacity of the AKOrN encoder by expanding the number of channels in its convolutional connectivities. Secondly, we discovered that increased pre-training periods and used efficient non-contrastive architectures like SimSiam and PhiNet to improve significantly adversarial robustness. Finally, the combination of a carefully selected AKOrN encoder with PhiNet pre-training for 400 epochs achieved a robust accuracy of 75.03%, closely approaching the current state-of-the-art of 75.28% without resorting to adversarial training or additional synthetic data.

More experiments still need to be run to evaluate completely the performances of this method and confirm the results we obtained. First, the new framework needs to be evaluated on more complicated and/or bigger datasets as *CIFAR-100* or *ImageNet*. Then, the representational power of Kuramoto layers could be even more leverage by implementing them in new architectures like ResNet. On the one hand this could enable the improvement of the clean accuracy and therefore of the overall robustness accuracy of our AKOrN models, because ResNet gives very good clean accuracy results on CIFAR-10. On the other hand, this could help limitate the non-stability of the training that we have highlighted.

Supplemantary Material

AutoAttack Complements

Difference of Logits Ratio Loss

The Difference of Logits Ratio (DLR) is a loss that has been defined and used for AutoAttack (cf. Section 3) in order to cancel some counter-attacks based on extreme rescaling that can induce gradient masking.

The authors of AutoAttack introduced the **Difference of Logits Ratio (DLR)** loss, which is invariant under both shifts and rescaling of the logits. This ensures that it has the same degrees of freedom as the classifier's decision:

$$\mathrm{DLR}(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}}, \tag{2}$$

where $z \in \mathbb{R}^C$ denotes the vector of logits produced by the classifier for C classes, and π is the permutation that orders the components of z in decreasing value. The use of a logit difference in the denominator guarantees shift-invariance.

Maximizing DLR with respect to the input x identifies adversarial points that are not assigned to the true label y (since $\mathrm{DLR}(x, y) > 0$ only if $\arg\max_i z_i \neq y$). Once this is achieved, the loss further reduces the relative confidence of class y compared to the competing classes.

When x is correctly classified, we have $\pi_1 = y$, and the loss simplifies to

$$\mathrm{DLR}(x, y) = -\frac{z_y - z_{\pi_2}}{z_y - z_{\pi_3}}, \qquad \text{with} \quad \mathrm{DLR}(x, y) \in [-1, 0].$$

Here, the normalization factor $z_{\pi_1} - z_{\pi_3}$ encourages solutions where $z_y \approx z_{\pi_2} > z_{\pi_3}$, effectively biasing the optimization towards changing the model's prediction.

References

Miyato, Takeru et al. (2025). *Artificial Kuramoto Oscillatory Neurons*. arXiv: 2410.13821 [cs.LG]. URL: https://arxiv.org/abs/2410.13821.

Ishikawa, Satoki et al. (2025). *PhiNets: Brain-inspired Non-contrastive Learning Based on Temporal Prediction Hypothesis*. arXiv: 2405.14650 [cs.LG]. URL: https://arxiv.org/abs/2405.14650.

Chen, Y. et al. (2024). "Predictive sequence learning in the hippocampal formation". In: *Neuron* 112, pp. 2645–2658.

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Elman, Jeffrey L. (1990). "Finding structure in time". In: *Cognitive Science* 14.2, pp. 179–211.

Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558.

Chen, Ting et al. (2020). "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. Vol. 119. PMLR, pp. 1597–1607.

Chen, Xinlei and Kaiming He (2021). "Exploring simple siamese representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15750–15758.

Grill, Jean-Bastien et al. (2020). "Bootstrap your own latent: A new approach to self-supervised learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33, pp. 21271–21284.

Croce, Francesco, Maksym Andriushchenko, et al. (2020). "RobustBench: a standardized adversarial robustness benchmark". In: *arXiv preprint arXiv:2010.09670*.

Croce, Francesco and Matthias Hein (2020a). *Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks*. arXiv: 2003.01690 [cs.LG]. URL: https://arxiv.org/abs/2003.01690.

Hinton, Geoffrey E. and Ruslan R. Salakhutdinov (2006). "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786, pp. 504–507.

Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, et al. (2010). "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *Journal of Machine Learning Research* 11.106, pp. 3371–3408.

Gori, Pietro (2025). *Representation Learning for the MVA Master Program*. https://pietrogori.github.io/teaching/RepLearnMVA. Accessed: 2025-07-22.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, et al. (2008). "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning (ICML)*. ACM, pp. 1096–1103.

Pathak, Deepak et al. (2016). "Context Encoders: Feature Learning by Inpainting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544.

He, Kaiming et al. (2022). "Masked Autoencoders Are Scalable Vision Learners". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009.

Noroozi, Mehdi, Hamed Pirsiavash, and Paolo Favaro (2017). "Representation Learning by Learning to Count". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 5898–5906.

Becker, Stephen and Geoffrey E. Hinton (1992). "A Self-organizing Neural Network That Discovers Surfaces in Random-dot Stereograms". In: *Nature* 355, pp. 161–163.

Bromley, Jane et al. (1993). "Signature Verification using a "Siamese" Time Delay Neural Network". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 6, pp. 737–744.

Russakovsky, Olga et al. (2015). "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3, pp. 211–252.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533.

Hubel, David H and Torsten N Wiesel (1962). "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of Physiology* 160.1, pp. 106–154.

Somers, David C, Sacha B Nelson, and Mriganka Sur (1995). "An emergent model of orientation selectivity in cat visual cortical simple cells". In: *Journal of Neuroscience* 15.8, pp. 5448–5465.

Aoyagi, Toshio (1995). "Network of neural oscillators for retrieving phase information". In: *Physical Review Letters* 74.20, pp. 4075–4078.

McClelland, J. L., B. L. McNaughton, and R. C. O'Reilly (1995). "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory". In: *Psychological Review* 102.3, p. 419.

Oord, A. van den, Y. Li, and O. Vinyals (2018). "Representation Learning with Contrastive Predictive Coding". In: *arXiv preprint arXiv:1807.03748*.

Kurakin, Alexey, Ian J. Goodfellow, and Samy Bengio (2017). "Adversarial examples in the physical world". In: *Proceedings of the International Conference on Learning Representations (ICLR) Workshop*.

Croce, Francesco and Matthias Hein (2020b). "Minimally distorted adversarial examples with a fast adaptive boundary attack". In: *International Conference on Machine Learning (ICML)*.

Andriushchenko, Maksym et al. (2020). "Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search". In: *European Conference on Computer Vision (ECCV)*.

Amini, Sajjad et al. (2024). *MeanSparse: Post-Training Robustness Enhancement Through Mean-Centered Feature Sparsification*. arXiv: 2406.05927 [cs.CV]. URL: https://arxiv.org/abs/2406.05927.

Wang, Feng et al. (2021). "Understanding the behaviour of contrastive loss". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2495–2504.