Novel Losses for Contrastive Learning using Siamese Energy-Based Models

Anonymous authors
Paper under double-blind review

Abstract

Learning useful representations without labels is central to modern machine learning, especially when annotation is costly, motivating the development of self-supervised learning. To that end, contrastive learning methods, such as SimCLR, aim to discover representations that are invariant to user-defined augmentations. Recent work has shown that these methods can be reinterpreted as energy-based models (EBMs) that learn to "de-augment" data. Building on this perspective, we propose a principled EBM formulation of contrastive representation learning. Through this formulation, we are able to offer new objectives to train this model. Particularly, we propose a Fisher-Hyvärinen divergence loss, which leverages score matching to bypass the need for negative samples. Our framework bridges contrastive learning with EBM and posterior estimation, offering a new foundation for unsupervised representation learning.

1 Introduction

Learning compact representations in machine learning has become central to building flexible and efficient models. By mapping high-dimensional data to compact and meaningful embeddings, learned representations enable transfer across tasks, improve generalisation, and reduce the reliance on costly annotated datasets. In domains such as computer vision (Henaff, 2020; Chen et al., 2020), natural language processing (Izacard et al., 2022; Gao et al., 2021), and speech (Chang et al., 2021), representation learning has been a key driver of recent breakthroughs. Contrastive learning further met successful applications for instance in medical context (Hager et al., 2023; Wang et al., 2022; Yu et al., 2025), material recognition (Drehwald et al., 2023) or astrophysics (Hayat et al., 2021; Wei et al., 2022).

Unsupervised representation learning methods most often rely on contrasting positive samples with negative samples, where positives are generated through data augmentations and negatives are drawn from other instances. A seminal work in this line is SimCLR (Chen et al., 2020), which showed that high-quality representations can be learned by contrasting augmented views of the same data point against a large set of negatives. Crucially, SimCLR demonstrated that strong augmentations, a projection head, and large batch sizes are sufficient to achieve state-of-the-art self-supervised performance without the need for specialised architectures or memory banks.

To alleviate the dependence on large negative sets, subsequent work (Kim and Ye, 2022) suggested interpreting contrastive learning as an energy-based model (EBM) that learns a joint distribution between data samples and their augmentations. Similarly, Ohl et al. (2025) showed that contrastive learning can be understood as EBMs that learn to de-augment positive samples. This framing provides a principled probabilistic view of representation learning: rather than being defined only through contrastive losses, the process can be formalised as decoding augmentations with an underlying energy function.

In this work, we extend this interpretation and propose a unified view of representation learning that decouples the roles of augmentations, architectures, and objectives. We show that representation learning can be cast as a decoding problem within an EBM framework, where sampling is not the goal, but rather the structure of the model itself provides a foundation for new training objectives.

- Firstly, we show that modelling the "de-augmenting" distribution using an EBM requires tilting the true data distribution for which we have no access.
- Inspired by the energy-based model literature, we derive different loss to train the "de-augmenting" model. We both propose likelihood-based and score-based losses, the latter not requiring any negative samples for the exact same architecture.
- We show in simple linear settings the optimal solutions for these losses and sufficient conditions for collapse.
- Finally, we show with simple experiments that those losses exhibit similar behaviours to classical losses in contrastive learning on tabular datasets depending on the choice of energy, though some optimisation remains for image datasets.

2 One EBM, multiple losses

We begin by showing how most contrastive learning models can be re-interpreted as energy-based de-augmenters. From this observation, we then build novel losses, and later provide some solutions for linear parametrisations.

2.1 The task

Let $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$ denote samples from an unknown data distribution, where $\boldsymbol{x} \in \mathcal{X}$. We aim to learn a representation function ψ_θ with parameters θ:

$$\begin{aligned} \psi_\theta : \mathcal{X} &\to \mathcal{Z}, \\ \boldsymbol{x} &\mapsto \psi_\theta(\boldsymbol{x}) = \boldsymbol{z}. \end{aligned} \tag{1}$$

Our approach follows the contrastive learning paradigm. Throughout, we consider $\mathcal{X} = \mathbb{R}^d$ the data space and $\mathcal{Z} = \mathbb{R}^m$ the representation space. Given a sample $\boldsymbol{x}$, we generate an augmented version $\tilde{\boldsymbol{x}} \sim p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$. The augmentation mechanism is design-dependent: Gaussian perturbations for tabular data, random crops or colour jitter for images, etc. It may also come from a generative model (Wang et al., 2024a).

We seek to approximate the unknown "de-augmenting" conditional distribution $p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ by a parameterised model $q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$. This is formulated as the minimisation of a statistical divergence D:

$$\mathcal{L}(\theta, \tilde{\boldsymbol{x}}) = D(p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) \,\|\, q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}))\,. \tag{2}$$

Averaging over all possible augmentations yields the global objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(\tilde{\boldsymbol{x}})}\big[\mathcal{L}(\theta, \tilde{\boldsymbol{x}})\big] = \mathbb{E}_{p(\tilde{\boldsymbol{x}})}[D(p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) \,\|\, q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}))]\,. \tag{3}$$

We now propose to model this de-augmenting distribution q_θ via an EBM:

$$q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \frac{\exp\left(-E_\theta(\boldsymbol{x}, \tilde{\boldsymbol{x}})\right)}{Z_{\tilde{\boldsymbol{x}}}}. \tag{4}$$

The energy function E_θ is dependent on a notion of proximity δ between the samples. The most commonly used distance is the temperature-scaled cosine distance on a Siamese network:

$$\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{\tau}\left(1 - \frac{\langle \psi_\theta(\boldsymbol{x}), \psi_\theta(\tilde{\boldsymbol{x}}) \rangle}{\|\psi_\theta(\boldsymbol{x})\| \|\psi_\theta(\tilde{\boldsymbol{x}})\|}\right). \tag{5}$$

This is notably motivated by the influence of the embedding norms (Draganov et al., 2025), which may encode model confidence and help convergence. We will also consider in this paper the Euclidean distance $\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{2}\|\psi_\theta(\boldsymbol{x}) - \psi_\theta(\tilde{\boldsymbol{x}})\|_2^2$. As explored in related works, see Appendix A, the proximity δ does not need to be symmetric. For this paper, we will keep it symmetric, using Siamese networks for ψ_θ.

An intuitive, though naive, approach is to directly model $E_\theta = \delta$. However, such a setup does not recover the classical contrastive losses without introducing a tilt, as we show later. As shown by Ohl et al. (2025), based on the lower bounds of Poole et al. (2019), the energy function is defined via exponential tilting (see Siegmund, 1976) of the unknown data distribution:

$$E_\theta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) - \log p_{\text{data}}(\boldsymbol{x}). \tag{6}$$

Note that this tilt is *conceptual*, not implementational. Its only operational role is the theoretical equivalence to contrastive losses that we will unfold in the following sections. Out of convenience, we will refer to E_θ as either the *Euclidean energy* or the *cosine energy* depending on the choice of distance for δ. We may note that for the cosine energy, the EBM gets factored by the constant $e^{\frac{1}{\tau}}$, which is cancelled by the normalisation constant, leading to classical cosine similarity as remainder. The normalisation constant of the EBM is now an expectation:

$$Z_{\tilde{\boldsymbol{x}}} = \int_{\mathcal{X}} e^{-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})} p_{\text{data}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}[e^{-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})}]. \tag{7}$$

In other words, the tilt leverages a normalisation constant that can be estimated by sampling from the data distribution. It would be an intractable integral otherwise. The main difficulty with using an EBM comes from the normalising constant $Z_{\tilde{\boldsymbol{x}}}$ which is intractable.

This model can be interpreted as a specific form of EBM where the energy balances distance and log-likelihood. It seeks to generate samples which are simultaneously close in representation to $\tilde{\boldsymbol{x}}$ and likely in the data distribution. We may note that this model paradoxically cannot be sampled from because $p_{\text{data}}(\boldsymbol{x})$ is unknown, making the energy impossible to evaluate. In comparison, Kim and Ye (2022) did not adopt such a tilted energy and directly modelled the EBM based on the distance between representations. They interpreted this model accordingly as the joint distribution between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, rather than a de-augmenting distribution like we do. Here, the representations are obtained as a by-product of learning a model that reconstructs data from augmentations.

This re-writing of self-supervised learning models also alleviates a new perspective on degenerate representations. When the model ψ_θ returns a constant for all $\boldsymbol{x}$, the distance δ is always null. Therefore, the model $q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ collapses to $p_{\text{data}}(\boldsymbol{x})$. In other words, the easiest fallback is to sample from the data distribution rather than try de-augmenting $\tilde{\boldsymbol{x}}$. The optimal model should satisfy $\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = -\log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$. We further explore the connections with related works in Appendix A.

We now present various estimation strategies according to the selected distance D in the loss of Equation (3).

2.2 Kullback-Leibler Divergence

For instance, using the Kullback-Leibler (KL) divergence leads to the following formula:

$$\begin{aligned}
\mathcal{L}_{\text{KL}}(\theta) &= \mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[D_{\text{KL}}\big(p(\boldsymbol{x}|\tilde{\boldsymbol{x}}) \,\|\, q_\theta(\boldsymbol{x}|\tilde{\boldsymbol{x}})\big)\right], && (8)\\
&= \mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[-\mathbb{E}_{p(\boldsymbol{x}|\tilde{\boldsymbol{x}})}[\log q_\theta(\boldsymbol{x}|\tilde{\boldsymbol{x}})]\right] + \mathbb{E}_{p(\boldsymbol{x}, \tilde{\boldsymbol{x}})}\left[\log p(\boldsymbol{x}|\tilde{\boldsymbol{x}})\right], && (9)\\
&= \mathbb{E}_{p(\boldsymbol{x}, \tilde{\boldsymbol{x}})}\big[\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})\big] + \mathbb{E}_{p(\tilde{\boldsymbol{x}})}[\log Z_{\tilde{\boldsymbol{x}}}] + \mathbb{E}_{p(\boldsymbol{x}, \tilde{\boldsymbol{x}})}\left[\log \frac{p(\boldsymbol{x}|\tilde{\boldsymbol{x}})}{p_{\text{data}}(\boldsymbol{x})}\right]. && (10)
\end{aligned}$$

Interestingly, we may identify in the KL loss the gap between the mutual information $\mathcal{I}$ and its lower bound as proposed by Barber and Agakov (2003), or rather its unnormalised version $\mathcal{I}_{\text{UBA}}$ (Poole et al., 2019, Eq. 4):

$$\mathcal{L}_{\text{KL}}(\theta) = -\mathcal{I}_{\text{UBA}}(\boldsymbol{x}; \tilde{\boldsymbol{x}}) + \mathcal{I}(\boldsymbol{x}; \tilde{\boldsymbol{x}}). \tag{11}$$

Thus minimising the KL objective is equivalent to minimising the gap between an unnormalised variational lower bound on mutual information and mutual information.

In practice, the expectation $\mathbb{E}_{p(\boldsymbol{x}, \tilde{\boldsymbol{x}})}\big[\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})\big]$ is replaced by a Monte Carlo approximation obtained with pairs of elements $(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) \sim p(\boldsymbol{x}, \tilde{\boldsymbol{x}})$:

$$\mathbb{E}_{p(\boldsymbol{x}, \tilde{\boldsymbol{x}})}\big[\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})\big] \approx \frac{1}{n} \sum_{i=1}^{n} \delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i). \tag{12}$$

The estimation of the normalisation constant can be done in different manners.

2.2.1 Direct approximation of the normalisation constant

Similarly, the normalisation constant conditioned on a given $\tilde{\boldsymbol{x}}$ can be estimated via Monte-Carlo samples; this is precisely what is done in the contrastive learning framework (Poole et al., 2019). The expectation in Equation (7) is directly replaced with a Monte-Carlo approximation with samples $\boldsymbol{x}_j$ from the distribution $p_{\text{data}}(\boldsymbol{x})$ independently from $p(\tilde{\boldsymbol{x}})$:

$$\hat{Z_{\tilde{\boldsymbol{x}}}} \approx \frac{1}{n} \sum_{j=1}^{n} \exp(-\delta(\boldsymbol{x}_j, \tilde{\boldsymbol{x}})). \tag{13}$$

The complete loss approximation leveraging joint samples $(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) \sim p(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ and disjoint samples $(\boldsymbol{x}_j, \tilde{\boldsymbol{x}}_i) \sim p_{\text{data}}(\boldsymbol{x})p(\tilde{\boldsymbol{x}})$

$$\mathcal{L}_{\text{KL}}(\theta) \approx \frac{1}{n} \sum_{i=1}^{n} \left(\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) + \log \frac{1}{n} \sum_{j=1}^{n} e^{-\delta(\boldsymbol{x}_j, \tilde{\boldsymbol{x}}_i)} \right) \tag{14}$$

In practice, each augmentation $\tilde{\boldsymbol{x}}_i$ is generated from a specific sample $\boldsymbol{x}_i$, so the matched pair $(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)$ is drawn from the joint, $(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) \sim p(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, and is a *positive* pair. To obtain *non-positive* (negative) pairs, one would ideally sample from $p_{\text{data}}(\boldsymbol{x})p(\tilde{\boldsymbol{x}})$. In a batched setting, this can be done efficiently by reusing samples and forming off-diagonal pairs $(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_j)$ with $i \neq j$. Since $\tilde{\boldsymbol{x}}_j$ was generated from $\boldsymbol{x}_j$ (not $\boldsymbol{x}_i$), these mismatched pairs are distributed as $(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_j) \approx p_{\text{data}}(\boldsymbol{x})p(\tilde{\boldsymbol{x}})$. The diagonal case $i = j$ cannot serve as a negative because $(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)$ comes from the joint, not the product of marginals, hence the constraint $i \neq j$ in InfoNCE (Van den Oord et al., 2018):

$$\mathcal{L}_{\text{KL}}(\theta) \approx \mathcal{L}_{\text{InfoNCE}}(\theta) = \frac{1}{n} \sum_{i} \left(\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) + \log \frac{1}{n} \sum_{i \neq j} e^{-\delta(\boldsymbol{x}_j, \tilde{\boldsymbol{x}}_i)} \right). \tag{15}$$

We may notice here that, in contrast to the usual setting in contrastive learning, we are considering positive and negative pairs from the perspective of the augmented sample $\tilde{\boldsymbol{x}}_i$, rather than the datum itself. Indeed, the normalising constant $Z_{\tilde{\boldsymbol{x}}}$ is estimated with samples from $p_{\text{data}}(\boldsymbol{x})$. To recover the usual contrastive loss, Poole et al. (2019) implicitly used the symmetry of mutual information on their lower bound to permute indices on the normalising constant:

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = \frac{1}{n} \sum_{i} \left(\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) + \log \frac{1}{n} \sum_{i \neq j} e^{-\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_j)} \right). \tag{16}$$

Thus, even though we permuted the indices in the estimation of the normalisation constant, we maintain the exact same model $q_\theta(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$. Interestingly, we may notice that we could directly get the InfoNCE loss by considering the following model: $q_\theta(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) \propto p(\tilde{\boldsymbol{x}}) \exp(-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}))$. We leave such considerations for future work.

Interestingly, using a single negative sample $\tilde{\boldsymbol{x}}_j$ in InfoNCE for estimating the normalisation constant leads to:

$$\mathcal{L}_{\text{KL}}(\theta) \approx \delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) - \delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_j) + \mathcal{I}(\boldsymbol{x}; \tilde{\boldsymbol{x}}). \tag{17}$$

This single-sample approximation resembles the triplet loss (Schroff et al., 2015) where the datum $\boldsymbol{x}_i$ acts as anchor, $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ as the respectively positive and negative examples. The last term is traditionally replaced by a hyperparameter called margin. The relationship between contrastive loss and triplet loss has already been established when the energy is the temperature-scaled cosine distance (Khosla et al., 2020, Appendix 4).

2.2.2 Learning the normalisation constant with Self-normalising Likelihood

Rather than approximating the normalisation constant through sampling, it can be learned directly during training. Following Senetaire et al. (2025) and Sander et al. (2025), we leverage the variational formulation of $\log Z_{\tilde{\boldsymbol{x}}}$ by introducing an auxiliary parameter λ:

$$\log Z_{\tilde{\boldsymbol{x}}} = \max_{\lambda \in \mathbb{R}} \left(Z_{\tilde{\boldsymbol{x}}} e^{-\lambda} + \lambda - 1 \right). \tag{18}$$

Applying this transformation on the Kullback-Leibler divergence for a single augmentation $\tilde{\boldsymbol{x}}$, we obtain:

$$D_{\mathrm{KL}}\big(p(\boldsymbol{x}|\tilde{\boldsymbol{x}}) \,\|\, q_\theta(\boldsymbol{x}|\tilde{\boldsymbol{x}})\big) = \mathbb{E}_{p(\boldsymbol{x}|\tilde{\boldsymbol{x}})}[\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})] + \log Z_{\tilde{\boldsymbol{x}}} \tag{19}$$

$$= \max_{\lambda\in\mathbb{R}} \mathbb{E}_{p(\boldsymbol{x}|\tilde{\boldsymbol{x}})}[\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})] + Z_{\tilde{\boldsymbol{x}}} e^{-\lambda} + \lambda - 1 \tag{20}$$

$$= \max_{\lambda\in\mathbb{R}} \mathbb{E}_{p(\boldsymbol{x}|\tilde{\boldsymbol{x}})}[\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) + \lambda] + \mathbb{E}_{p_{\mathrm{data}}(\boldsymbol{x})}\left[e^{-\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})-\lambda}\right] - 1. \tag{21}$$

Taking directly the expectation over all augmentations $\tilde{\boldsymbol{x}}$ is not directly possible because each of them will get a different normalisation constant. Instead, we model the variational variable to be dependent on the $\tilde{\boldsymbol{x}}$ using a neural network λ_φ. We maximize the loss of both the representation network ψ_θ and the bias network λ_φ:

$$\mathcal{L}(\theta) = \max_{\varphi} \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}[\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) + \lambda_\varphi(\tilde{\boldsymbol{x}})] + \mathbb{E}_{p(\tilde{\boldsymbol{x}})p_{\mathrm{data}}(\boldsymbol{x})}\left[e^{-\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})-\lambda_\varphi(\tilde{\boldsymbol{x}})}\right] - 1. \tag{22}$$

Following the design proposed by Gustafsson et al. (2022), we append a lightweight neural network head to the encoder. This head maps the learned representation $\psi_\theta(\boldsymbol{x})$ to a single scalar that estimates the log-normalisation constant, enabling λ_φ to be optimised jointly with the EBM parameters θ.

2.2.3 Learning the normalisation constant with Noise Contrastive Estimation

Following an idea from Gutmann and Hyvärinen (2010), one could use Noise Contrastive Estimation (NCE). In this seminal paper, the EBM training task is framed as a classification problem where we differentiate the samples coming from the true conditional $p(.|\tilde{\boldsymbol{x}})$ and some arbitrary negative distribution p_ζ. The idea is train the density $q_\theta(.|\tilde{\boldsymbol{x}})$ by maximising the logits of samples from the true conditional $p(.|\tilde{\boldsymbol{x}})$ and minimising the logits from the negative distribution p_ζ. The construction is detailed in the appendix Appendix C.

NCE begins by defining a mixture model with mixing proportion π between the density and the noisy distribution:

$$p_\theta(\boldsymbol{x}, y \mid \tilde{\boldsymbol{x}}) = \pi q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})\mathbb{1}[y=1] + (1-\pi)p_\zeta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})\mathbb{1}[y=0]. \tag{23}$$

It immediately follows that the posterior distribution of the mixing variable y, can be expressed using the following parametric logit, where we expanded the definition of our tilted EBM:

$$\ell_\theta(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \log\frac{\pi}{1-\pi} + \log q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) - \log p_\zeta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}), \tag{24}$$

$$= \log\frac{\pi}{1-\pi} + \log p_{\mathrm{data}}(\boldsymbol{x}) - \delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) - \log Z_{\tilde{\boldsymbol{x}}} - \log p_\zeta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}). \tag{25}$$

There, we can replace q_θ by our definition of the tilted EBM. In addition, we purposefully choose to use $p_{\mathrm{data}}(\boldsymbol{x}) = p_\zeta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ as a noisy distribution. In other words, the de-augmenting EBM should not be mistaken with random samples from the data. Following the standard implementation of this model in Gutmann and Hyvärinen (2010) and the bias network solution used in Section 2.2.2, we model the conditional distribution using an explicit normalisation constant $\lambda_\varphi(\tilde{\boldsymbol{x}}) \approx \log Z_{\tilde{\boldsymbol{x}}}$. This gives:

$$\ell_{\theta,\varphi}(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \log\frac{\pi}{1-\pi} - \delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) - \lambda_\varphi(\tilde{\boldsymbol{x}}). \tag{26}$$

Plugging all into the noise contrastive estimation gives the following objective to *minimize*:

$$\mathcal{L}_{\mathrm{NCE}}(\theta,\varphi) = -\pi\mathbb{E}_{p_{\mathrm{data}}(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\log\sigma\left(\ell_{\theta,\varphi}(\boldsymbol{x},\tilde{\boldsymbol{x}})\right)\right] + (1-\pi)\mathbb{E}_{p_{\mathrm{data}}(\boldsymbol{x})p_{\mathrm{data}}(\tilde{\boldsymbol{x}})}\left[\log\sigma\left(-\ell_{\theta,\varphi}(\boldsymbol{x},\tilde{\boldsymbol{x}})\right)\right], \tag{27}$$

where σ is the sigmoid function. In practice, we sample n pairs $(\boldsymbol{x},\tilde{\boldsymbol{x}})$ per batch from the joint distribution. This entails that the ratio of positive pair to noise sampling, that is to say negative pairs, is $\frac{1}{n-1}$, *i.e* $\pi = \frac{1}{n}$.

2.3 Fisher-divergence

Another approach is to focus on matching the gradients of score of the distributions instead of their exact values. To that end, we can use the Fisher-Hyvärinen distance (Hyvärinen and Dayan, 2005) between $p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ and the model q_θ. The loss becomes:

$$\begin{aligned}\mathcal{L}_{\mathcal{F}}(\theta) &= \mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[D_{\mathcal{F}}(p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) \| q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}))\right] &(28)\\ &= \mathbb{E}_{p(\tilde{\boldsymbol{x}},\boldsymbol{x})}\left[\|\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} \log q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})\|^2\right], \\ &= \mathbb{E}_{p(\tilde{\boldsymbol{x}},\boldsymbol{x})}\left[\|\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})\|^2\right], &(29)\end{aligned}$$

We then remind that the gradient of sum is equivalent to the sum of gradients. We can therefore combine the two first terms and use Bayes' theorem. The Fisher loss is thus:

$$\mathcal{L}_{\mathcal{F}}(\theta) = \mathbb{E}_{p(\tilde{\boldsymbol{x}},\boldsymbol{x})}\left[\|\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) + \nabla_{\boldsymbol{x}}\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})\|^2\right]. \tag{30}$$

This loss is straightforward to interpret: any variation of the data $\boldsymbol{x}$ making the augmentation $\tilde{\boldsymbol{x}}$ unlikely should be reflected by an increase of distance, and conversely.

We see that the challenge has now shifted to the evaluation of the gradient of the augmentation probability with respect to its condition $\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$. This replaces the evaluation of the normalisation constant $Z_{\tilde{\boldsymbol{x}}}$ in other losses. In turn, we note that this loss is exactly *non-contrastive*: it does not require any negative pair, negative sampling, etc. This loss only compares a data sample with its respective augmentation.

Like the KL loss, the Fisher loss is also sensitive to singular collapses. When all representations $\psi_\theta(\boldsymbol{x})$ are mapped equal to the same constant, the distance becomes 0 everywhere. This implies a null gradient on the entire data space. In other words, once the model collapsed, the Fisher loss cannot pull back the model from this state. However, we may note that the null distance $\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) = 0$ is not the optimal solution because the Fisher loss remains positive. The global solution is $\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) = -\log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$, returning a null Fisher loss.

2.4 Solutions for linear parametrisations

Before delving into experiments, we inspect some conditions under which the proposed losses may lead to optimal solutions. To that end, we parametrise q_θ with linear models $\psi_\theta : \boldsymbol{x} \mapsto \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ and the Euclidean energy. Under such a setting, we may not care about the representations derived from ψ_θ, as the focus is shifted to learning a Gram matrix $\boldsymbol{G} = \boldsymbol{W}^\top\boldsymbol{W}$, which parametrises the distance δ. Implicitly, the contrastive learning task becomes a metric learning task. While this model is quite simplistic, we believe this can be a useful parallel that omits the encoder network, and focuses on the classical projection head, which plays a key role in contrastive learning (Jing et al., 2022; Xue et al., 2023; Wen and Li, 2022).

For any augmentation with defined moments and fast-decaying data distribution, we first show with Lemma 2.1 that there exists an optimal weight $\boldsymbol{W}$, up to orthogonal reparametrisations (Proof in Appendix B.1).

Lemma 2.1. *Let $\psi_\theta(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}+\boldsymbol{b}$ be a linear layer from dimension d to m, and $\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})$ the squared Euclidean distance between $\psi_\theta(\boldsymbol{x})$ and $\psi_\theta(\tilde{\boldsymbol{x}})$. Then, for any data distribution $p_{data}(\boldsymbol{x})$ that is smooth over $\mathcal{X}$ and decaying sufficiently fast, and for any augmentation $\tilde{\boldsymbol{x}} \mid \boldsymbol{x}$ with defined variance $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} = \mathbb{V}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}[\tilde{\boldsymbol{x}}]$ and mean $\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} = \mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}[\boldsymbol{x}]$, the global minimizer of the Fisher loss $\mathcal{L}_{\mathcal{F}}(\theta)$ is $\boldsymbol{W}$ such that:*

$$\boldsymbol{W}^\top\boldsymbol{W} = -\left(\int_{\mathcal{X}} (\nabla_{\boldsymbol{x}} p_{data}(\boldsymbol{x}))^\top \boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}^\top \mathrm{d}\boldsymbol{x}\right) \times \left(\mathbb{E}_{p_{data}(\boldsymbol{x})}\left[\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} + (\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} - \boldsymbol{x})(\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} - \boldsymbol{x})^\top\right]\right)^{-1}. \tag{31}$$

An immediate consequence of Lemma 2.1 is Corollary 2.2. When we restrict augmentations to Gaussian noise, the optimal weight $\boldsymbol{W}$ must be equal to the noise precision, up to a shift by the noise mean.

Corollary 2.2. *Let $\psi_\theta(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ be a linear layer from dimension d to m, and $\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})$ the squared Euclidean distance between $\psi_\theta(\boldsymbol{x})$ and $\psi_\theta(\tilde{\boldsymbol{x}})$. If $\tilde{\boldsymbol{x}} \mid \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}+\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}})$, then for any smooth and fast-decaying distribution $p_{data}(\boldsymbol{x})$, the global minimizer of the Fisher loss $\mathcal{L}_{\mathcal{F}}(\theta)$ is $\boldsymbol{W}$ such that:*

$$\boldsymbol{W}^\top\boldsymbol{W} = \left(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\right)^{-1}. \tag{32}$$

Proof. We begin by plugging $\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}$ in the left-hand integral of Equation (31). This gives:

$$\int_{\mathcal{X}} (\nabla_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}))^\top (\boldsymbol{x} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}})^\top \mathrm{d}\boldsymbol{x} = -\boldsymbol{I} + \boldsymbol{0}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top = -\boldsymbol{I}. \tag{33}$$

The expectation on the right-hand side of Equation (31) does not depend any longer on $\boldsymbol{x}$ as well:

$$\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} + (\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} - \boldsymbol{x})(\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} - \boldsymbol{x})^\top \right] = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}} \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top \right], \tag{34}$$

$$= \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}} \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top. \tag{35}$$

Plugging both expression in Equation (31) gives Equation (32). □

We may notice that both minimal solutions can be only reached if the product $\boldsymbol{W}^\top \boldsymbol{W}$ is full rank, which requires the output dimension m of the linear layer to be greater or equal to d. While this does not guarantee that $\boldsymbol{W}$ is of rank d, we may note that random matrices have a high probability of being full-rank (Feng and Zhang, 2007) in practice. It is interesting to remark in this context that the goal of self-supervised learning, which is to learn rich and insightful features from the data, is optimally achieved by learning a mapping that increases the number of features.

In the case where the output dimension of the linear layer m is smaller than d, we get from the Eckart-Young-Misky theorem (Eckart and Young, 1936; Mirsky, 1960) that the best m-rank approximation of the optimal solution is the top m eigenvalues. Noting the decomposition $(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}} \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)^{-1} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^\top$ and assuming that eigenvalues are decreasingly sorted, we have $\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{U}_{:m} \boldsymbol{\Lambda}_{:m} \boldsymbol{U}_{:m}^\top$. We can then identify the set of solutions $\boldsymbol{W} = \boldsymbol{Q} \boldsymbol{\Lambda}_{:m}^{\frac{1}{2}} \boldsymbol{U}_{:m}^\top$, with $\boldsymbol{Q}$ any $m \times m$ orthogonal projection.

Finally, Lemma 2.1 directly offers a perspective on cases that are sufficient for collapse in linear models, though not necessary. We show in Theorem 2.3 that the optimal weights $\boldsymbol{W}$ must have null singular values when the augmentations involve any function with a non-positive Jacobian determinant. This relates to Jing et al. (2022) who showed that the optimisation of the the InfoNCE loss is guided by the difference of two covariance matrices (one for the data, another for the augmentation). They showed that as soon as this difference provides negative eigenvalues, there will be dimensional collapse. Dimensional collapse occurs when the final representation has a lower intrinsic dimension than the full capacity of the neural network output. As an example, we can take the augmentation $\tilde{\boldsymbol{x}} = -\boldsymbol{x} + \boldsymbol{\varepsilon}$, i.e. $f(\boldsymbol{x}) = -\boldsymbol{x}$, which leads to collapse for any odd input space dimension d.

Theorem 2.3. *Let $\psi_\theta(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ be a linear layer from dimension d to m, and $\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ the squared Euclidean distance between $\psi_\theta(\boldsymbol{x})$ and $\psi_\theta(\tilde{\boldsymbol{x}})$. If $\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} = f(\boldsymbol{x})$ for any function f s.t. $\det\left(\mathbb{E}_{p_{data}(\boldsymbol{x})}[\nabla_{\boldsymbol{x}} f(\boldsymbol{x})]\right) \leq 0$, then the optimal weight $\boldsymbol{W}$ will collapse to null singular values.*

Proof. We simply need to compute the left-hand side of Equation (31), using:

$$-\left(\int_{\mathcal{X}} (\nabla_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}))^\top \boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}^\top \mathrm{d}\boldsymbol{x} \right) = -\left(\int_{\mathcal{X}} (\nabla_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}))^\top f(\boldsymbol{x})^\top \mathrm{d}\boldsymbol{x} \right) = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \right]. \tag{36}$$

The right-hand side of Equation (31) is always positive semi-definite, so its determinant is non-negative. The sign of the determinant of the product depends on $\det\left(\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \right]\right)$. If it is non-positive, then Equation (31) is non-positive. Consequently, and since the product $\boldsymbol{W}\boldsymbol{W}^\top$ is positive semi-definite, the closest matrix in norm to the optimal solution will be $\boldsymbol{W}\boldsymbol{W}^\top = \boldsymbol{0}$, hence collapsing. □

To move to the KL loss, we consider two cases. If the Gaussian noise does not have any shift, $\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} = \boldsymbol{0}$, then the optimal squared weight must be equal to the noise precision for fast-decaying data distribution following Theorem 2.4 (Proof in Appendix B.2). We show that, conversely, the solution $\boldsymbol{W}^\top \boldsymbol{W} = (\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}} \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)^{-1}$ can also be attained in the KL when the data distribution is Gaussian for any shift $\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}$, see Theorem 2.5 (Proof in Appendix B.3).

Theorem 2.4. *Let $\psi_\theta(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ be a linear layer from dimension d to m, and $\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ the squared Euclidean distance between $\psi_\theta(\boldsymbol{x})$ and $\psi_\theta(\tilde{\boldsymbol{x}})$. If $\tilde{\boldsymbol{x}} \mid \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}})$, then for any smooth and fast-decaying distribution $p_{data}(\boldsymbol{x})$, the global minimizer of the KL loss $\mathcal{L}_{KL}(\theta)$ is $\boldsymbol{W}$ such that $\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}$.*

Theorem 2.5. *Let $\psi_\theta(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ be a linear layer from dimension d to m, and $\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ the squared Euclidean distance between $\psi_\theta(\boldsymbol{x})$ and $\psi_\theta(\tilde{\boldsymbol{x}})$. For any Gaussian distribution $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\Sigma}_{\boldsymbol{x}})$ and any Gaussian augmentation $\tilde{\boldsymbol{x}} \mid \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}})$, the global minimum of the KL loss is reached for $\boldsymbol{W}^\top \boldsymbol{W} = (\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}} \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)^{-1}$.*

The most common proximity δ in self-supervised learning is the temperature-scaled cosine distance. We may consider drawing samples from any distribution, and passing them through a neural network ψ_θ with normalisation, as drawing samples directly on $m-1$ hypersphere. In this sense, Wang and Isola (2020) highlighted that contrastive learning favours uniform distribution of representations on the hypersphere while maintaining positive pairs close. Noting $\boldsymbol{z} = \psi_\theta(\boldsymbol{x})$ and $\tilde{\boldsymbol{z}} = \psi_\theta(\tilde{\boldsymbol{x}})$, we may approximate the change of variables by reparametrising $\boldsymbol{z} \sim \mathcal{U}(\mathbb{S}^{m-1})$ and augmentation $\tilde{\boldsymbol{z}} \mid \boldsymbol{z} \sim \text{vMF}(\boldsymbol{z} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \beta)$. This oversimplification shifts the goal of the neural network ψ_θ, which is now to optimise the concentration parameter β. For this approximation, Theorem 2.6 implies that minimising the KL loss leads to an inevitable increase in concentration β to infinity (Proof in Appendix B.4).

Theorem 2.6. *Let $\boldsymbol{z}$ be uniformly distributed on the $m-1$ hypersphere. Let $\tilde{\boldsymbol{z}} \mid \boldsymbol{z} \sim vMF(\boldsymbol{z} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \beta)$ be the augmentations of $\boldsymbol{z}$. Then, the KL loss $\mathcal{L}_{KL}(\beta)$ converges to its minimum as $\beta \to \infty$.*

To conclude, our theory goes along the results of Wen and Li (2022) who observed that freezing the projection head, even to the identity matrix, can lead models to learn good features nonetheless. From our perspective, this is equivalent to enforcing a given noise precision matrix for the augmentation distribution at the representation level of ψ_θ, right before the final linear layer.

3 A proof of concept

Before listing performance of the different losses in generic context, we focused here on a small tabular case with a clear objective. Inspired by Ohl et al. (2025), we sampled two concentric circles, and seek to disentangle them. The dataset is simply 200 samples taken from `scikit-learn`'s `make_circles` function with standard deviation of 0.05, and a scale factor of 0.1 between inner and outer circle.

To that end, we used for ψ_θ a simple two-layer MLP with 20 hidden nodes, 2 output nodes and ReLU activation. This allows to directly visualise if the circles were properly disentangled. Note that unlike the traditional protocol in contrastive learning, we did not use here a disposable projection head. For each run, we always started from the same initial weights. Each model is trained using an Adam optimizer over 10,000 iterations with a learning rate of 0.01. In practice, less than 2,000 iterations sufficed to disentangle the circles.

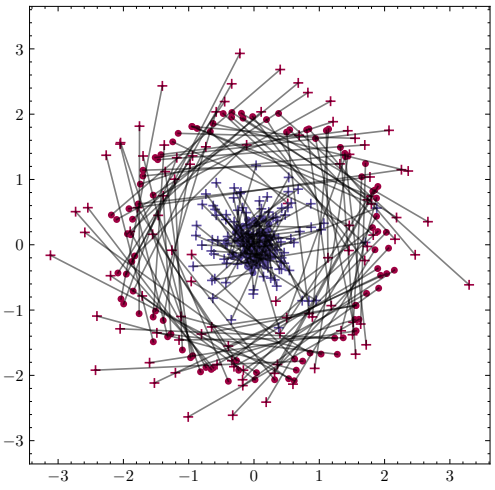
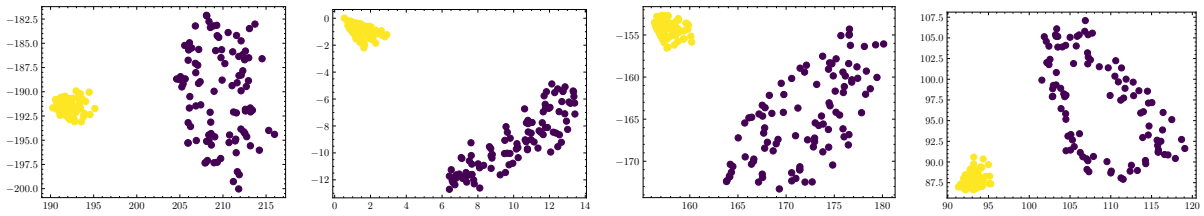


Figure 1: Concentric circle dataset with a random rotation augmentation and some Gaussian noise. Inner and outer circles are distinguished by colour. Dots represent the original samples whereas crosses represent the augmentation. Black lines link original samples to their respective augmentation.

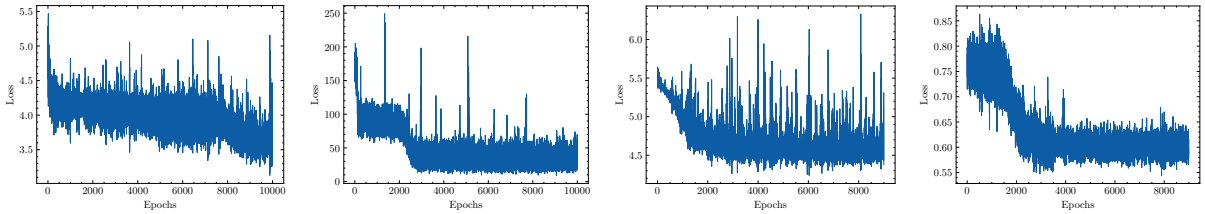
We chose a simple set of augmentations: the samples are first randomly rotated by either $-\frac{\pi}{2}$, 0 or $\frac{\pi}{2}$. We then added isotropic Gaussian noise with 0.1 standard deviation. We give a simple visualisation of the resulting dataset and augmentations in Figure 1. Importantly, we feed to the neural network both the original sample $\boldsymbol{x}$ and the augmentation $\tilde{\boldsymbol{x}}$, meaning that we never use paired augmentations as is customary in contrastive learning. We give further details on the implementation choices for the losses in Appendix D.

After training, we can directly visualise the two-dimensional output of the neural networks in Figure 2. All four losses yield a discriminative latent space that properly separates inner and outer circles. We provide for completeness the training loss curves in Figure 3. These curves are non-informative with regards to the success of the given task.



(a) KL loss (b) Fisher (c) SNL loss (d) NCE loss

Figure 2: Learned 2d representation by the neural network when disentangling the concentric circles according to different losses. For each loss, the neural network successfully separate the inner samples (yellow) from the outer samples (purple).



(a) KL loss (b) Fisher (c) SNL loss (d) NCE loss

Figure 3: Loss curves of each neural network depending on each training loss.

4 Experiments

The primary goal of our experiments is to demonstrate that all our proposed losses can be used for unsupervised representation learning. We did not seek to optimise hyperparameters for performance, but rather to observe if loss changes introduced significant differences. We begin by benchmarking performance on 18 tabular datasets selected from the Penn Machine Learning Benchmark (Romano et al., 2021) that have only continuous features, see Table 1, then follow with examples on image datasets using only CIFAR10 and CIFAR100. We tried both the Euclidean energy and cosine energy δ for tabular datasets, and kept only the cosine energy for image datasets because it is standard practice in high dimension. Still, we empirically found that collapses often occurred when combining Euclidean distance and likelihood-based losses, and suggest using it only for the Fisher loss. All models started from the exact same weights, unless told otherwise. Code is available at `https://anonymous.4open.science/r/EBMRL-B784/`.

4.1 General protocol

4.1.1 Comparisons

We identified 3 losses for comparison purposes: SimCLR (Chen et al., 2020), Spectral loss (HaoChen et al., 2021) and SimSiam (Chen and He, 2021). We used the two former for all experiments, and the latter only for image experiments as it requires a specific projection head, introducing therefore architectural differences between models. For the spectral loss, we followed HaoChen et al. (2021) and kept a hypersphere of squared radius $\mu = 10$.

4.1.2 Augmentations

Among the proposed methods, the Fisher loss is limited by the requirement of a gradient over the augmentation conditional densities. We propose two different augmentations, an exact augmentation for tabular datasets, and an approximate for image datasets. For the tractable tabular augmentation, we take inspiration from

Table 1: Summary of the datasets used from the Penn Machine Learning Benchmark (Romano et al., 2021). All features are continuous and were standard-scaled.

Name	Samples	Dimensions	Number of classes
analcatdata authorship	841	70	4
appendicitis	106	7	2
churn	5000	20	2
collins	485	23	13
led24	3200	24	10
letter	20000	16	26
optdigits	5620	64	10
satimage	6435	36	6
shuttle	58000	9	7
sonar	208	60	2
spambase	4601	57	2
spectf	349	44	2
texture	5500	40	11
vehicle	846	18	4
waveform 40	5000	40	3
wine quality red	1599	11	6
wine quality white	4898	11	7
yeast	1479	8	9

Ucar et al. (2021) and propose to add random Gaussian noise while randomly masking features:

$$\tilde{\boldsymbol{x}} = \boldsymbol{\alpha}\boldsymbol{x} + \boldsymbol{\varepsilon} \qquad \boldsymbol{\alpha} \sim \mathcal{B}(\rho), \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}), \tag{37}$$

where ρ is the preservation rate. For the image datasets, we leveraged a gradient estimate by exploiting the fast decay of Gaussian noise after image augmentations. In other words, we deterministically augmented the image using a function, then back-propagated through this function. Detailed derivations are provided in Appendix E.

For tabular datasets, we trained directly using an original view on the data and an augmented view for all losses. For image datasets, we sampled two augmented views of the datasets only for the SimCLR, SimSiam and spectral loss in respect of their original design.

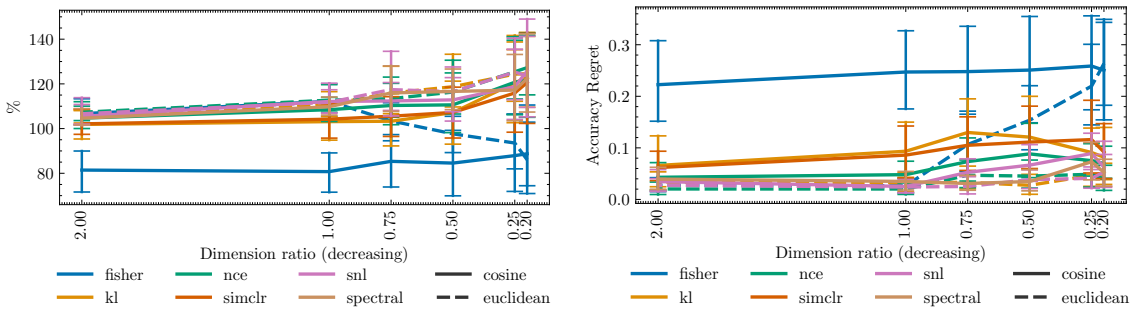
4.1.3 Evaluation

We mainly evaluated models using linear probing. We further completed these results by providing recent internal scores specialised for unsupervised learning, namely: CLID (Lu et al., 2023), RankMe (Garrido et al., 2023a), and the Alignment/Uniformity score (Wang and Isola, 2020). For the sake of brevity, we mainly report RankMe and linear probing accuracy, while delegating other metrics to appendices. We evaluated those metrics on the entire tabular datasets, and on the test sets of CIFAR10 and CIFAR100. We describe in detail the different metrics in Appendix A.3.

For linear probing, we used different models depending on tabular or image datasets. For tabular datasets, we used the scikit-learn (v1.8.0) logistic regression with 1000 iterations, and default parameters. For image datasets, we used 90 training epochs, and an SGD optimiser with momentum 0.9, no weight decay, and a cosine-annealed learning rate starting from 0.01. For tabular datasets, we trained on an 80% split and reported the score on the remaining 20%. For CIFAR10 and CIFAR100, we used the default splits.

4.2 Tabular datasets

Training In this experiment, all model started from the same weights, and differed only by their losses. Models were trained for 20,000 iterations with a batch size of 256 samples. We used an Adam optimiser with



(a) Relative improvement (↑) over initialisation

(b) Regret (↓) on best score

Figure 4: Accuracy performance of the same model trained with different losses after 20,000 iterations as the output decreases with respect to the number of initial features. The tabular augmentations preserves 80% of the initial features, before adding Gaussian noise. Error bars show standard errors. Dimension ratio is output over input m/d.

a learning rate of 0.01. This learning rate is warmed up during the 100 first iterations, then fades away owing to cosine annealing. To avoid collapse due to regularisation, we removed weight decay. We used a very simple MLP for the model:

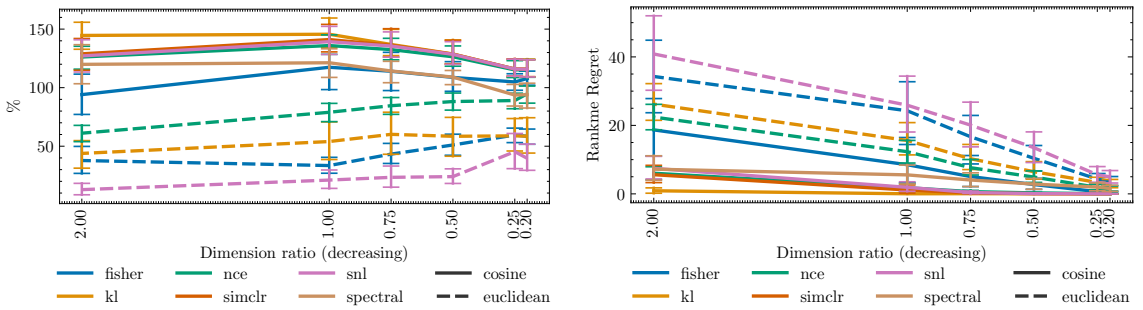
$$\psi_\theta(\boldsymbol{x}) = \boldsymbol{W}_2 \mathrm{ReLU}(\boldsymbol{W_1 x} + \boldsymbol{b}_1) + \boldsymbol{b}_2. \tag{38}$$

with d input features, a hidden dimension of $2d$ and an output dimension of m features. We did not use any projection head to keep the model simple. For the SNL loss, we implemented the bias λ_φ as 2-layer-MLP with m inputs, $\max(m/2, 2)$ hidden units and a scalar output, with batch norm and ReLU.

Setups We explored performance under two different setups. In the first setup, we fixed $\rho = 0.8$ (see Equation (37)), and varied m from $2d$ to $d/5$, with $\sigma = 0.1$. The purpose is to see whether, under limited but non-negligible masking, the models may learn a correct representation in spite of small or large output dimensions. We report accuracy in Figure 4 and RankMe in Figure 5. In the second setup, we fixed the output dimension m to $d/2$, and vary the preservation rate ρ, with $\sigma = 0.25$. The purpose it to see whether the lower-dimensional representations remain robust as the original features progressively get destroyed. We report accuracy in Figure 6 and RankMe in Figure 7. Additional metrics can be found in Appendix F.

Aggregating results To account for the variability among the different datasets, we used two aggregation methods to provide meaningful averages over fixed hyperparameters. First, we report the relative score, *i.e.* the performance of the model after training divided by the performance before training. For metrics that must be maximised, the relative score must exceed 100%, and conversely be below for metrics to minimize. Second, we report the regret score, which measures the gap between a score and the top-performing score among all models. The lower the better, indicating that a model is systematically close to top performance.

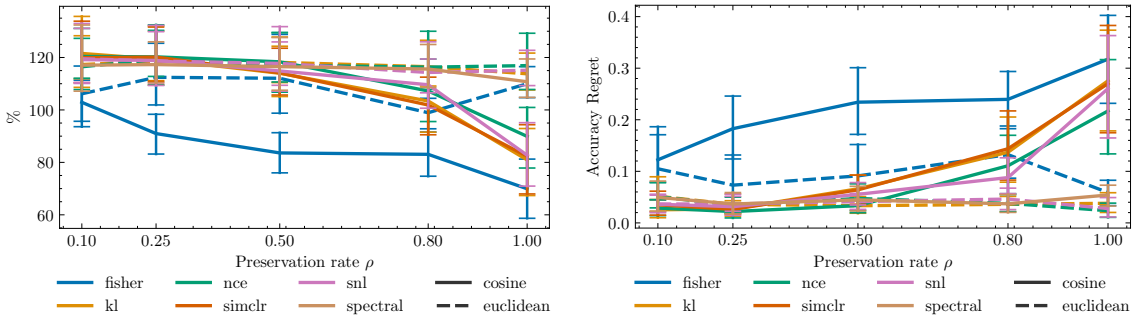
When using the Euclidean energy, we see that SNL is the strongest competitor as it simultaneously achieves one of the highest relative accuracy improvements while having the lowest RankMe score, using the fewest dimensions. This is the opposite behaviour to the Fisher loss with cosine energy, which barely improves the number of useful dimensions, as shown by a relative RankMe close to 100%. The simplest take-away from Figure 4 is that accuracy improvement spreads for all losses as the output dimension of the neural network decreases, except the Fisher loss with cosine energy. Moreover, for output dimensions smaller than the inputs, we see the Fisher loss also decrease in accuracy improvements with the Euclidean energy. We mitigate this loss with better results in Appendix F when using a learning rate of 1. Interestingly, while most losses achieve a nearly equal accuracy improvement, their RankMe scores behave differently in Figure 5 depending on the choice of energy. As the cosine energy forces representations to spread on the hypersphere, the performance improves in RankMe score, which highlights that the representations use all available dimensions equally. In contrast, the Euclidean energy does not constrain the output to lie on a specific manifold. Therefore, models tend to converge to non-detrimental dimensional collapses, as shown by relative RankMe scores lower than



(a) Relative improvement (↑) over initialisation

(b) Regret (↓) on best score

Figure 5: RankMe performance of the same model trained with different losses after 20,000 iterations as the output decreases with respect to the number of initial features. The tabular augmentations preserves 80% of the initial features, before adding Gaussian noise. Error bars show standard errors. Dimension ratio is output over input m/d.



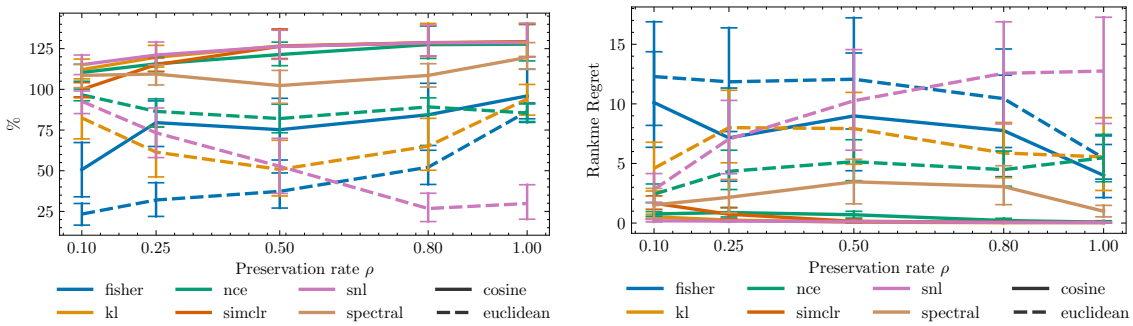
(a) Relative improvement (↑) over initialisation

(b) Regret (↓) on best score

Figure 6: Accuracy performance of the same model trained with different losses after 20,000 iterations. The tabular augmentations preserves only a fraction ρ of the initial features, before adding Gaussian noise. Error bars show standard errors.

100%, and regret scores away from zero. As the dimension ratio decreases, the number of available unused dimensions decreases, which tightens the gap between Euclidean and cosine energy in RankMe performance, deteriorating the relative accuracy. Finally, all likelihood-based losses exhibit the same fluctuations under the cosine energy.

When fixing the output to $m = d/2$, the strength of the augmentation, controlled by the preservation rate ρ, plays a crucial role in downstream performance. We first observe in Figure 6 that the more the features are preserved, *i.e.* the weaker the augmentation, the lower the accuracy with respect to initialisation, especially for the cosine energy. The only loss that manages in this setting to keep a competitive accuracy while using the cosine energy is the spectral loss. With the Euclidean energy, likelihood-based losses prove to be slightly better than the Fisher loss throughout all preservation rates. The RankMe scores now exhibit different behaviours in Figure 7. For all losses with cosine energy, weakening the augmentation, i.e. increasing the preservation rate ρ, increases the relative improvement and maintains a near-zero regret score. The Fisher loss is the worst in this case, as it both loses in accuracy and in number of useful features. For the Euclidean energy, the trends are loss-dependent. The augmentation strength seems to have little to no effect on the NCE loss. Weakening the augmentation decreased the relative RankMe score of the SNL loss, and conversely increased that of the Fisher loss.



(a) Relative improvement (↑) over initialisation

(b) Regret (↓) on best score

Figure 7: RankMe performance of the same model trained with different losses after 20,000 iterations. The tabular augmentations preserves only a fraction ρ of the initial features, before adding Gaussian noise. Error bars show standard errors.

Our take-away is that the choice of energy between Euclidean and cosine may not have an impact on linear probing performance for strong augmentations. However, if the dimension of the representation manifold matters, then the cosine energy should be picked for favouring hypersphere spreading, and the Euclidean energy should be used for low-dimensional representations. Among likelihood-based losses, only the spectral loss (with cosine energy) could maintain competitive performance against the Euclidean-energy losses.

4.3 Standard image datasets

Training We now switch to a ResNet18 following Chen et al. (2020, Appendix B.9). All models had an additional projection head of dimension 128 with 2 layers, except for the SimSiam loss, which required its own specific head (Chen and He, 2021). All models used a cosine energy with temperature $\tau = 0.1$. We used an SGD optimizer with learning rate 0.2, 10 epochs of linear warm-up, and cosine annealing over 1,000 epochs with batch size 512. We used a weight decay of 0.0005, except for the SNL and Fisher losses, where we set it to 0 to prevent collapses. Given the poor performance of the Fisher loss, we also trained it on an MLP with two hidden layers of size 1200 (Hinton et al., 2012) and no projection head. All ResNet18 start from the same initial weights. For SNL and NCE, we implemented a bias neural network λ_φ with dimensions [128, 256, 1].

Augmentations We used the augmentations from SimSiam (Chen and He, 2021), with a random resized crop, random horizontal flip, colour jittering, and gray scale. For all losses except the Fisher loss, we used Gaussian blur. We remind that for the SimCLR, SimSiam and spectral losses, we provide a positive pair of two augmentations from the same datum, whereas the KL, SNL and Fisher loss get a single augmentation.

Table 2 shows that the proposed losses in general do not perform better than losses that were specifically developed in deep learning contexts. Still, despite underperforming on CIFAR10, we observe that the NCE loss matches the baselines on CIFAR100 in linear probing. In addition, we may note that the trained models used a fair share of the 512 encoder dimensions, with RankMe scores above 300 for the KL loss in Table 3, and fewer but still above 200 for other likelihood-based losses. For the SNL loss, our attempts on CIFAR100 ended up in unfortunate degenerate collapses, despite identical parameters to the CIFAR10 setup. Such collapse originates from the high variance inherent to importance sampling in large dimension and was extensively described in the original paper (Senetaire et al., 2025). Similarly, the Fisher loss empirically proved sensitive and difficult to train, resulting in poor performance for both datasets.

Despite our attempts to provide gradient estimates for the Fisher loss, we did not manage to reach a linear probing equivalent to likelihood-based losses on images on CIFAR100. One of the brakes to these results that we empirically noticed was the involvement of batch norm layers in the neural network, leading to a different gradient norm between $\nabla_{\boldsymbol{x}}\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ and the augmentation gradient. Even though switching to an MLP with no batch normalisation enhanced the performance, it remains far from all other methods. Still, this follows our

Table 2: Downstream linear classification accuracy of Resnet18 models pretrained on CIFAR datasets depending on their loss and projection function. (After 1000 epochs at batch size 512). The naive baseline performance is the linear probing of the Resnet18 upon initialisation.

Loss	Naive baseline	SimCLR	Simsiam	Spectral	KL	NCE	SNL	Fisher	
								ResNet18	MLP
CIFAR10	31.2	84.6	86.5	89.2	76.6	78.6	77.5	14.4	24.6
CIFAR100	14.2	49.6	28.3	52.6	37.4	52.6	NaN	1.6	4.9

Table 3: RankMe score of Resnet18 models pretrained on CIFAR datasets depending on their loss and projection function. (After 1000 epochs at batch size 512). The naive baseline performance is the linear probing of the Resnet18 upon initialisation

Loss	Naive baseline	SimCLR	Simsiam	Spectral	KL	NCE	SNL	Fisher	
								ResNet18	MLP
CIFAR10	69	381	320	349	337	222	245	5	8
CIFAR100	69	388	312	370	345	225	NaN	2	7

earlier results were the Fisher loss with cosine energy showing significant performance downgrade in general. Finally, unlike tabular datasets, the Euclidean energy was not beneficial to the Fisher loss with a ResNet18 backbone. Empirically, we found that targeting a specific gradient norm with ResNet18s was challenging, and is one of the main contributing factors to the failure of this loss.

5 Conclusion

We revisited contrastive learning as an energy-based model that seeks to de-augment augmentations back to the original data distribution. By defining the energy through a distance between Siamese network outputs, we provided examples of novel losses for self-supervised learning. Among these losses, we showed that likelihood-based losses through KL divergence exhibit similar behaviours to classical losses like SimCLR. In particular, we leveraged through the Fisher divergence a score-based loss that does not require any contrasting term and can train the same model to similar performance, specifically when using the Euclidean distance between features on tabular datasets. Through simplistic experiments, we showed that the losses exhibit similar performance variations, up to few points of difference. However, there remains a large performance gap for the score-based loss when dealing with images, which is likely due to architectural design and the difficulty of computing an exact supervisory gradient with image augmentations.

This alternative writing of contrastive learning opens the path for multiple future works. First of all, more losses can be developed by considering other distances between the true decoding distribution and the model. Second, this may question how to evaluate the model, namely whether we are interested in the model q_θ or the tool for leveraging an energy ψ_θ. Finally, future works could explore how to improve performance of the Fisher loss, for instance by improving the conditional gradient of augmentations, or by directly modelling the score of the energy rather than computing it through automatic differentiation.

References

Han Bao, Yoshihiro Nagano, and Kento Nozawa. On the Surrogate Gap between Contrastive and Supervised Losses. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1585–1606. PMLR, July 2022.

David Barber and Felix Agakov. The IM Algorithm: A Variational Approach to Information Maximization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'03, pages 201–208, Cambridge, MA, USA, 2003. MIT Press.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. In *International Conference on Learning Representations*, 2022.

Suzanna Becker and Geoffrey E. Hinton. Self-Organizing Neural Network That Discovers Surfaces in Random-Dot Stereograms. *Nature*, 355(6356):161–163, January 1992. ISSN 1476-4687. doi: 10.1038/355161a0.

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.

Sungkyun Chang, Donmoon Lee, Jeongsoo Park, Hyungui Lim, Kyogu Lee, Karam Ko, and Yoonchang Han. Neural audio fingerprint for high-specific audio retrieval based on contrastive learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3025–3029. IEEE, 2021.

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.

Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, June 2021.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

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 *International Conference on Learning Representations*, 2021.

Andrew Draganov, Sharvaree Vadgama, Sebastian Damrich, Jan Niklas Böhm, Lucas Maes, Dmitry Kobak, and Erik J Bekkers. On the Importance of Embedding Norms in Self-Supervised Learning. In *International Conference on Machine Learning*, pages 14417–14438. PMLR, 2025.

Manuel S. Drehwald, Sagi Eppel, Jolina Li, Han Hao, and Alan Aspuru-Guzik. One-Shot Recognition of Any Material Anywhere Using Contrastive Learning with Physics-Based Rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23524–23533, October 2023.

Carl Eckart and Gale Young. The Approximation of One Matrix by Another of Lower Rank. *Psychometrika*, 1(3):211–218, 1936. doi: 10.1007/BF02288367.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the Intrinsic Dimension of Datasets by a Minimal Neighborhood Information. *Scientific Reports*, 7(1):12140, September 2017. ISSN 2045-2322. doi: 10.1038/s41598-017-11873-y.

Xinlong Feng and Zhinan Zhang. The rank of a random matrix. *Applied mathematics and computation*, 185 (1):689–694, 2007.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 6894–6910. Association for Computational Linguistics (ACL), 2021.

Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. RankMe: Assessing the Downstream Performance of Pretrained Self-Supervised Representations by Their Rank. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10929–10974. PMLR, July 2023a.

Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann LeCun. On the duality between contrastive and non-contrastive self-supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023b.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. In *International Conference on Learning Representations*, 2018.

Hariprasath Govindarajan, Per Sidén, Jacob Roll, and Fredrik Lindsten. On Partial Prototype Collapse in the DINO Family of Self-Supervised Methods. In *British Machine Vision Conference (BMVC) 2024*, 2024.

Florian Graf, Christoph Hofer, Marc Niethammer, and Roland Kwitt. Dissecting Supervised Contrastive Learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3821–3830. PMLR, July 2021.

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

Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schön. Learning proposals for practical energy-based regression. In *International Conference on Artificial Intelligence and Statistics*, pages 4685–4704. PMLR, 2022.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

Paul Hager, Martin J Menten, and Daniel Rueckert. Best of both worlds: Multimodal contrastive learning with tabular and imaging data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23924–23935, 2023.

Jeff Z. HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable Guarantees for Self-Supervised Deep Learning with Spectral Contrastive Loss. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 5000–5011. Curran Associates, Inc., 2021.

Md Abul Hayat, George Stein, Peter Harrington, Zarija Lukić, and Mustafa Mustafa. Self-supervised Representation Learning for Astronomical Images. *The Astrophysical Journal Letters*, 911(2):L33, April 2021.

Junlin He, Jinxiao Du, and Wei Ma. Preventing Dimensional Collapse in Self-Supervised Learning via Orthogonality Regularization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 95579–95606. Curran Associates, Inc., 2024. doi: 10.52202/079017-3028.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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.

Olivier Henaff. Data-Efficient Image Recognition with Contrastive Predictive Coding. In *International conference on machine learning*, pages 4182–4192. PMLR, 2020.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors, 2012.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

Jiabo Huang, Shaogang Gong, and Xiatian Zhu. Deep Semantic Clustering by Partition Confidence Maximisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8849–8858, 2020.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding Dimensional Collapse in Contrastive Self-supervised Learning. In *International Conference on Learning Representations*, 2022.

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. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc., 2020.

Beomsu Kim and Jong Chul Ye. Energy-based contrastive learning of visual representations. *Advances in Neural Information Processing Systems*, 35:4358–4369, 2022.

Alexander C. Li, Alexei A. Efros, and Deepak Pathak. Understanding Collapse in Non-contrastive Siamese Representation Learning. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 490–505, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19821-2.

Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive Clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8547–8555, May 2021. doi: 10.1609/aaai.v35i10.17037.

Yuchen Lu, Zhen Liu, Aristide Baratin, Romain Laroche, Aaron Courville, and Alessandro Sordoni. Using Representation Expressiveness and Learnability to Evaluate Self-Supervised Learning Methods. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Siwei Lyu. Interpretation and Generalization of Score Matching. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 359–366, Arlington, Virginia, USA, 2009. AUAI Press. ISBN 978-0-9749039-5-8.

Leon Mirsky. Symmetric Gauge Functions and Unitarily Invariant Norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, January 1960. ISSN 0033-5606. doi: 10.1093/qmath/11.1.50.

Louis Ohl, Pierre-Alexandre Mattei, and Frederic Precioso. A tutorial on discriminative clustering and mutual information. *ACM Comput. Surv.*, September 2025. ISSN 0360-0300. doi: 10.1145/3748255. Just Accepted.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On Variational Bounds of Mutual Information. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5171–5180. PMLR, June 2019.

Joseph D Romano, Trang T Le, William La Cava, John T Gregg, Daniel J Goldberg, Praneel Chakraborty, Natasha L Ray, Daniel Himmelstein, Weixuan Fu, and Jason H Moore. Pmlb v1.0: an open source dataset collection for benchmarking machine learning methods. *arXiv preprint arXiv:2012.00058v2*, 2021.

Michaël E Sander, Vincent Roulet, Tianlin Liu, and Mathieu Blondel. Joint learning of energy-based models and their partition function. *arXiv preprint arXiv:2501.18528*, 2025.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Hugo Senetaire, Paul Jeha, Pierre-Alexandre Mattei, and Jes Frellsen. Learning energy-based models by self-normalising the likelihood. *arXiv preprint arXiv:2503.07021*, 2025.

David Siegmund. Importance sampling in the monte carlo study of sequential tests. *The Annals of Statistics*, pages 673–684, 1976.

Zhiquan Tan, Yifan Zhang, Jingqin Yang, and Yang Yuan. Contrastive Learning is Spectral Clustering on Similarity Graph. In *The Twelfth International Conference on Learning Representations*, 2024.

Talip Ucar, Ehsan Hajiramezanali, and Lindsay Edwards. SubTab: Subsetting Features of Tabular Data for Self-Supervised Representation Learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18853–18865. Curran Associates, Inc., 2021.

Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv e-prints*, pages arXiv–1807, 2018.

Tongzhou Wang and Phillip Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR, July 2020.

Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5549–5560, 2022.

Yifei Wang, Jizhe Zhang, and Yisen Wang. Do generated data always help contrastive learning? In *The Twelfth International Conference on Learning Representations*, 2024a.

Yifei Wang, Qi Zhang, Yaoyu Guo, and Yisen Wang. Non-negative Contrastive Learning. In *The Twelfth International Conference on Learning Representations*, 2024b.

Zifeng Wang, Zhenbang Wu, Dinesh Agarwal, and Jimeng Sun. Medclip: Contrastive learning from unpaired medical images and text. In *Conference on Empirical Methods in Natural Language Processing*, volume 2022, page 3876, 2022.

Shoulin Wei, Yadi Li, Wei Lu, Nan Li, Bo Liang, Wei Dai, and Zhijian Zhang. Unsupervised Galaxy Morphological Visual Representation with Deep Contrastive Learning. *Publications of the Astronomical Society of the Pacific*, 134(1041):114508, November 2022. doi: 10.1088/1538-3873/aca04e.

Zixin Wen and Yuanzhi Li. The Mechanism of Prediction Head in Non-contrastive Self-supervised Learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24794–24809. Curran Associates, Inc., 2022.

Yihao Xue, Siddharth Joshi, Eric Gan, Pin-Yu Chen, and Baharan Mirzasoleiman. Which Features Are Learnt by Contrastive Learning? On the Role of Simplicity Bias in Class Collapse and Feature Suppression. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38938–38970. PMLR, July 2023.

Yihao Xue, Eric Gan, Jiayi Ni, Siddharth Joshi, and Baharan Mirzasoleiman. Investigating the Benefits of Projection Head for Representation Learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Han Yu, Hanrui Lyu, YiXun Xu, Charlie Windolf, Eric Kenji Lee, Fan Yang, Andrew M. Shelton, Olivier Winter, International Brain Laboratory, Eva L. Dyer, Chandramouli Chandrasekaran, Nicholas A. Steinmetz, Liam Paninski, and Cole Lincoln Hurwitz. In Vivo Cell-Type and Brain Region Classification Via Multimodal Contrastive Learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. Learning Diverse and Discriminative Representations Via the Principle of Maximal Coding Rate Reduction. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 978-1-7138-2954-6.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.

Chaoning Zhang, Kang Zhang, Chenshuang Zhang, Trung X. Pham, Chang D. Yoo, and In So Kweon. How Does SimSiam Avoid Collapse Without Negative Samples? A Unified Understanding with Self-supervised Contrastive Learning. In *International Conference on Learning Representations*, 2022.

Jianping Zhang, Lei Yang, Seyed Mahmoud Sajjadi Mohammadabadi, and Feng Yan. A Survey on Self-Supervised Learning: Recent Advances and Open Problems. *Neurocomputing*, page 131409, August 2025. ISSN 0925-2312. doi: 10.1016/j.neucom.2025.131409.

Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful Image Colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive Learning Inverts the Data Generating Process. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12979–12990. PMLR, July 2021.

A Related works

A.1 Architectures and losses in self-supervised learning

In the last decade, self-supervised learning emerged as a meaningful way to derive high-quality features from data. Given unlabelled data, a pretext task is derived from its structure to leverage a supervisory signal (Doersch et al., 2015; Zhang et al., 2016; Gidaris et al., 2018). In contrastive learning (Chen et al., 2020; HaoChen et al., 2021), the pretext task consists in producing two different augmented views $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_2$ from an original sample $\boldsymbol{x}$, and attempting to learn consistent features for both views, thereby achieving invariance to the chosen augmentation.

To train for an augmentation invariance, multiple variations of the contrastive loss have been proposed(Chen et al., 2020; He et al., 2020), often being connected to InfoNCE (Van den Oord et al., 2018) as we discussed in Section 2.2. More broadly, "contrasting" corresponds to the estimation of the normalisation constant of the EBM from Section 2.2. This notably gives an intuitive justification for the requirement of large batch sizes in contrastive learning (Chen et al., 2020): more samples allow for a better estimation of the normalisation constant. It was also noted that larger batch size may help achieve better downstream accuracy (Bao et al., 2022). Overall, self-supervised learning models attempting to have identical features between two augmented views can also be rewritten as the EBM:

$$q(\tilde{\boldsymbol{x}}_2 \mid \tilde{\boldsymbol{x}}_1, \boldsymbol{x}) \propto p_{\text{data}}(\tilde{\boldsymbol{x}}_2 \mid \boldsymbol{x}) \exp\left(-\delta(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2)\right). \tag{39}$$

In that sense, the data $\boldsymbol{x}$ becomes a guiding confounding variable for the de-augmenting distribution q, although it does not affect the distance δ. Moreover, by using the same augmentation process for both views $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_2$, the variables become interchangeable in q, which allows symmetrised losses as is often done in practice (Chen and He, 2021; Grill et al., 2020).

Traditionally, contrastive learning models stem from Siamese networks aiming for similar features between positive pairs (Becker and Hinton, 1992). Accounting for normalised features, their energy corresponds to the temperature-scaled cosine distance:

$$\delta_{\theta,\omega}(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2) = \frac{1}{\tau}\left(1 - \frac{\langle g_\omega \circ \psi_\theta(\tilde{\boldsymbol{x}}_1), g_\omega \circ \psi_\theta(\tilde{\boldsymbol{x}}_2)\rangle}{\|g_\omega \circ \psi_\theta(\tilde{\boldsymbol{x}}_1)\| \|g_\omega \circ \psi_\theta(\tilde{\boldsymbol{x}}_2)\|}\right). \tag{40}$$

In this formulation, the neural network comprises two parts: a backbone network ψ_θ, typically a ResNet (He et al., 2016) or ViT (Dosovitskiy et al., 2021), and a projection head g_ω. The latter is traditionally discarded after training, and only the features from the backbone are kept because they provide better performance for downstream tasks (Chen et al., 2020; HaoChen et al., 2021; Chen and He, 2021; Caron et al., 2021).

However, contrasting is expensive and often requires large batches of negative samples to work efficiently. Solutions have been proposed to tackle collapsing issues while avoiding contrasting samples. The most common approach consists in breaking the symmetry of the Siamese architecture, and offer different learning procedures for each network (Chen and He, 2021; Grill et al., 2020; Caron et al., 2021). This can be interpreted in our framework as the following asymmetric energy, with for instance cosine distance:

$$\delta_{\theta,\omega}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{\tau}\left(1 - \frac{\langle \psi_\theta(\boldsymbol{x}), \chi_\omega(\tilde{\boldsymbol{x}})\rangle}{\|\psi_\theta(\boldsymbol{x})\| \|\chi_\omega(\tilde{\boldsymbol{x}})\|}\right). \tag{41}$$

The most straightforward way to separate networks is to involve stop-gradient operations between identical architectures (Caron et al., 2021). In this case, the parameters of the network ω can be updated through exponential moving average from the weights θ (Caron et al., 2021; Grill et al., 2020; He et al., 2020). It is also possible to have different architectures, or partially similar. This is the case of BYOL (Grill et al., 2020), and SimSiam (Chen and He, 2021), for which the absence of collapse has been studied by Zhang et al. (2022). It is also possible to replace one of the neural networks by a set of learnable prototypes that enforce sparsity among features.

Implicitly, the carefully designed variations in architecture avoid the need for the estimation of a normalisation constant. Hence, these methods can be labelled as *non-contrastive* (Grill et al., 2020; Caron et al., 2021; He

et al., 2020). This stands in contrast to the Fisher loss established in Section 2.3, which is non-contrastive and maintains a Siamese architecture. Barlow Twins (Zbontar et al., 2021) and VICReg (Bardes et al., 2022) are similar examples of Siamese models yet involving an apparently non-contrastive loss. Such losses focusing on the covariance matrix of embeddings have been coined *dimension contrastive* by Garrido et al. (2023b), who showed their implicit equivalence to contrastive learning. Interestingly, we may as well note that some methods tried to apply contrastive loss on features rather than sample vectors (Huang et al., 2020; Li et al., 2021). Informally, for an embedding matrix, the contrastive loss is applied on columns rather than rows. However, such losses do not fit the self-supervised EBM design we propose because they consider different random variables (Ohl et al., 2025).

Parallels have also been established between contrastive losses and matrix factorisation (Wang et al., 2024b). Similarly, interpreting embedding matrices as connected graphs leverages spectral interpretation of losses. This notably connects with spectral clustering (Tan et al., 2024), and helps leverage the seminal spectral contrastive loss (HaoChen et al., 2021):

$$\mathcal{L}_{\text{spectral}}(\theta) = -2\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\mathbb{E}_{\tilde{\boldsymbol{x}}_1\tilde{\boldsymbol{x}}_2\sim p(\cdot\mid\boldsymbol{x})}\left[\psi_\theta(\tilde{\boldsymbol{x}}_1)^\top\psi_\theta(\tilde{\boldsymbol{x}}_2)\right] + \mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{x}_2\sim p_{\text{data}}}\mathbb{E}_{\tilde{\boldsymbol{x}}_1\sim p(\cdot\mid\boldsymbol{x}_1),\tilde{\boldsymbol{x}}_2\sim p(\cdot\mid\boldsymbol{x}_2)}\left[(\psi_\theta(\tilde{\boldsymbol{x}}_1)^\top\psi_\theta(\tilde{\boldsymbol{x}}_2))^2\right], \quad (42)$$

where the outputs $\psi_\theta(\cdot)$ are normalised.

To the best of our knowledge, the only parallel and attempt that has been yet done with EBM and contrastive learning is the work of Kim and Ye (2022). However, their choice was to view the untilted version of the model, *i.e* $E_\theta = \delta$ in Equation (4), as a joint distribution over $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, rather than a conditional. Consequently, their optimisation required the decomposition of the joint between a conditional distribution $p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ and a regularised marginal $\lambda p(\boldsymbol{x})$. In this design, their experiments showed that a low value of λ provided better results than $\lambda = 1$. This further comforts the idea that contrastive models can be perceived as conditional de-augmenting distributions, rather than joint models. In a similar fashion, Zimmermann et al. (2021) highlighted that contrastive learning models could reconstruct well the original latent features $\psi_\theta(\boldsymbol{x})$ if the data was generated from a hyperspherical prior.

Interestingly, for Siamese networks, the optimal solution $\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = -\log(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$ questions the relationship between the choice of augmentation $p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$, and the symmetric property of the distance δ. This apparent incompatibility could explain why asymmetrical architectures showed improved performance over Siamese networks.

A.2 Collapses in contrastive learning

Beside the motivation to find richer features in contrastive learning, the development of novel architectures and losses or regularisations is heavily driven by collapse avoidance. As methods improved, the taxonomy of collapses in self-supervised learning also expanded. For instance, there can be dimensional collapses (Wang and Isola, 2020), meaning that all features get flattened on a low-dimensional hyperplane smaller than the entire output space of the neural network. Prototype collapse (Govindarajan et al., 2024) occurs when the features of a model revolve around a subset of the prototypes used to leverage training. In supervised settings, class collapse (Graf et al., 2021) refers to all features of samples within a class becoming undistinguishable because they end up in the same location in a simplex. The most trivial collapse of all is the degenerate solution where *all* features become equal.

One of the main ways of addressing collapses is to tweak the backbone architecture. Indeed, Xue et al. (2024) analysed the role of the projection head and demonstrate that, for a 2-layer linear MLP, spectral losses allocate greater representational norm to augmentation-stable features. Their Theorem 3.5 shows that the dominant learned features are bounded by the rank of the first-layer weight matrix, linking dimensional expressivity directly to architectural constraints. For instance, He et al. (2024) introduced Orthogonality Regularisation (OR), which enforces orthogonality across convolutional and linear layers. Their results show that OR mitigates dimensional collapse not only in the learned representations but also in weight matrices and hidden features, leading to robust improvements. By drawing a parallel between contrastive objectives and non-negative matrix factorisation, Wang et al. (2024b) motivated the use of a differentiable ReLU layer at

the final network stage to induce sparsity and reduce collapse tendencies. At a larger scale, Zhang et al. (2025) showed that increasing network width, *e.g.* in ResNet-50 variants used in MoCo, SimCLR, and VICReg, systematically improves downstream performance, suggesting that overparameterisation may offer additional resilience against collapse by increasing the number of available representation directions.

Note that we have a very natural interpretation of collapses in contrastive models, or rather de-augmenting in our context. Indeed, for a neural network ψ_θ that collapses to a singular output, the distance δ becomes always 0, irrelevantly of the inputs. Consequently, the EBM is interpreted:

$$q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) \propto p_{\text{data}}(\boldsymbol{x}) \exp\left(-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})\right), \tag{43}$$

$$= p_{\text{data}}(\boldsymbol{x}). \tag{44}$$

In other words, the safest strategy to de-augment data in case of collapse is simply to sample from the dataset, ignoring thus all conditioning. This observation empirically advocates that "*contrastive learning needs stronger data augmentations than supervised learning*" (Chen et al., 2020). By ensuring that augmentations are somewhat strong, we can expect that there exists a notable difference between $p_{\text{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ and $p_{\text{data}}(\boldsymbol{x})$, which would encourage the EBM $q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ to avoid targetting $p_{\text{data}}(\boldsymbol{x})$. Note that in the case of methods that involve a double augmentation, we get: $q_\theta(\tilde{\boldsymbol{x}}_2 \mid \tilde{\boldsymbol{x}}_1, \boldsymbol{x}) = p_{\text{data}}(\tilde{\boldsymbol{x}}_2 \mid \boldsymbol{x})$, as mentioned in the previous section.

Nonetheless, a compromise needs to be found as too strong augmentations may lead as well to collapse. Wang et al. (2024a) showed for instance that an excessive amount of data augmentations taken from a generative model, e.g. DDPM (Ho et al., 2020), can be detrimental to the performance of contrastive learning models. Similarly, Wang and Qi (2022) proposed to balance weak and strong augmentations to avoid deterioration of model performance that may arise when augmentations severely affect the structure of the data inputs. Beyond the choice of augmentation, Li et al. (2022) investigated collapse in neural networks and observed that continual learning schedules can substantially reduce collapse relative to standard multi-epoch training. Still, we previously showed in Theorem 2.3 that, even for some simple settings, some choice of augmentation lead to collapse because the null matrix is part of the optima.

A.3 Evaluating the EBM

Assuming that the EBM has been trained by one of the losses from Section 2, the natural following step is to evaluate it. However, there is now a distinction between evaluating the learned features from ψ_θ, and evaluating if the model q_θ is good at its task: de-augmenting data. To the best of our knowledge, there does not exist metrics to evaluate how q_θ excels at de-augmenting, and we leave this question to future works.

The standard evaluation protocol for assessing features is linear probing, e.g. in Chen et al. (2020); HaoChen et al. (2021); Caron et al. (2021); Ucar et al. (2021). The parameters θ are frozen, and a logistic regression is fitted on the dataset $\{\psi_\theta(\boldsymbol{x}_i), y_i\}$. Then, the model with best testing accuracy is kept. However, such an evaluation requires labels, which are not necessarily available for the unsupervised task at hand, and it depends as well on additional stochastic parameters.

During the latest years, unsupervised scoring methods have been proposed to circumvent the need for labels when evaluating representation models. Wang and Isola (2020) suggested for instance *uniformity* and *alignment*. The former ensures that all features are evenly spaced on the hypersphere, which is related to the cosine energy δ, whereas the latter verifies that positive pairs $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}} \mid \boldsymbol{x}$ indeed have similar energy, in the sense of δ. Some efforts were also done to evaluate the actual dimensionality of the representation space. To that end, Garrido et al. (2023a) created RankMe, the entropy of the normalised singular value decomposition of the representation space. Ideally, all dimensions should equally informative, leading to a high entropy. Based on the work of Facco et al. (2017), Lu et al. (2023) approximate the intrinsic dimension of the representation manifold using a 2-nearest neighbours estimator. It is assumed that representations will use the greatest intrinsic dimension value in order to preserve information of fine-grained categories. They further proposed *cluster learnability*, where a KNN classifier assesses if it can recover in a validation set clusters derived from KMeans clustering. KMeans was also used in the MCR2 score (Yu et al., 2020), an information-theoretic measure seeking maximal gap between the global reduction rate of features and their cluster-wise reduction rate.

B Proofs of theorems

B.1 Proof of Lemma 2.1

We parametrise the model q_θ using the simplest representation function: a linear layer $\psi_\theta(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, with $\boldsymbol{W} \in \mathbb{R}^{m \times d}$. We consider as well the Euclidean energy:

$$\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{2}\|\psi_\theta(\boldsymbol{x}) - \psi_\theta(\tilde{\boldsymbol{x}})\|^2 = \frac{1}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2_{\boldsymbol{G}}, \tag{45}$$

where $\boldsymbol{G} = \boldsymbol{W}^\top \boldsymbol{W}$. We may note that the constant $\boldsymbol{b}$ is useless, meaning that the learnable parameters are $\theta = \{\boldsymbol{G}\}$. The EBM is hence defined as:

$$q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \frac{p_{\text{data}}(\boldsymbol{x})}{Z_{\tilde{\boldsymbol{x}}}} \exp\left(-\frac{1}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2_{\boldsymbol{G}}\right), \tag{46}$$

where:

$$Z_{\tilde{\boldsymbol{x}}} = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\exp\left(-\frac{1}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2_{\boldsymbol{G}}\right)\right]. \tag{47}$$

In the case where $\boldsymbol{G}$ is positive-definite, we may notice that the right-hand factor is the energy of a Gaussian distribution, where $\boldsymbol{G}$ acts as a precision matrix. This requires that the matrix $\boldsymbol{W}$ has rank at least d, which requires $m \geq d$. Assuming $\boldsymbol{G}$ positive-definite, we may interpret q_θ as:

$$q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \frac{p_{\text{data}}(\boldsymbol{x})}{Z_{\tilde{\boldsymbol{x}}}} \times \sqrt{\det(2\pi \boldsymbol{G}^{-1})}\mathcal{N}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}, \boldsymbol{G}^{-1}). \tag{48}$$

We choose any data distribution $p_{\text{data}}(\boldsymbol{x})$ that vanishes quickly as $\|\boldsymbol{x}\| \to \infty$. For the augmentation, we choose any distribution such that the conditional variance $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} = \mathbb{V}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}[\tilde{\boldsymbol{x}}]$ and mean $\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} = \mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}[\tilde{\boldsymbol{x}}]$ are defined. The loss is:

$$\begin{aligned}
\mathcal{L}_{\mathcal{F}}(\theta) &= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\|\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) + \nabla_{\boldsymbol{x}}\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})\|^2\right], &(49)\\
&= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\|\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) + (\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top \boldsymbol{G}\|^2\right]. &(50)
\end{aligned}$$

We can now derive this loss with respect to the parameter $\boldsymbol{G}$. We notice that the inner term is dominated by $\|\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})\|^2 + \|(\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top\|^2\|\boldsymbol{G}\|^2$, which has finite expectation under mild regularity conditions on the augmentations $p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$. We may therefore move the derivative with respect to $\boldsymbol{G}$ inwards, while expanding the squared norm:

$$\begin{aligned}
\nabla_{\boldsymbol{G}}\mathcal{L}_{\mathcal{F}}(\theta) &= \nabla_{\boldsymbol{G}}\mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\|\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) + (\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top \boldsymbol{G}\|^2\right], &(51)\\
&= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\nabla_{\boldsymbol{G}}\left(\|\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})\|^2 + \|(\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top \boldsymbol{G}\|^2 + 2\langle \nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}), \boldsymbol{G}(\boldsymbol{x} - \tilde{\boldsymbol{x}})\rangle\right)\right], &(52)\\
&= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\nabla_{\boldsymbol{G}}\|(\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top \boldsymbol{G}\|^2 + 2\nabla\boldsymbol{G}\langle \nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}), \boldsymbol{G}(\boldsymbol{x} - \tilde{\boldsymbol{x}})\rangle\right]. &(53)
\end{aligned}$$

Then, we can use the classical identities of matrix calculus:

$$\nabla_{\boldsymbol{G}}\mathcal{L}_{\mathcal{F}}(\theta) = \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[2\boldsymbol{G}(\boldsymbol{x} - \tilde{\boldsymbol{x}})(\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top\right] + 2\mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[(\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}))^\top (\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top\right]. \tag{54}$$

We can now simplify the first term. By decomposing the expectation over the data and the conditional augmentation, we have:

$$\mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[2\boldsymbol{G}(\boldsymbol{x}-\tilde{\boldsymbol{x}})(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\right] = 2\boldsymbol{G}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}\left[(\tilde{\boldsymbol{x}}-\boldsymbol{x})(\tilde{\boldsymbol{x}}-\boldsymbol{x})^\top\right]\right], \tag{55}$$

$$= 2\boldsymbol{G}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} + (\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}-\boldsymbol{x})(\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}-\boldsymbol{x})^\top\right]. \tag{56}$$

For the second term, we need the usual regularity assumptions on $p_{\text{data}}(\boldsymbol{x})$. We have:

$$\mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[(\nabla_{\boldsymbol{x}}\log p(\tilde{\boldsymbol{x}}\mid\boldsymbol{x}))^\top(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\right] = \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[(\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}}\log p_{\text{data}}(\boldsymbol{x}))^\top(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\right], \tag{57}$$

$$= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[(\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}}) - \nabla_{\boldsymbol{x}}\log p_{\text{data}}(\boldsymbol{x}))^\top(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\right], \tag{58}$$

$$= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[(\frac{\nabla_{\boldsymbol{x}}p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})}{p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})} - \frac{\nabla_{\boldsymbol{x}}p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})})^\top(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\right]. \tag{59}$$

We can then decompose the first expectation:

$$\mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\left(\frac{\nabla_{\boldsymbol{x}}p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})}{p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})}\right)(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\right] = \mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[\int_{\mathcal{X}}(\nabla_{\boldsymbol{x}}p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}}))^\top(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\mathrm{d}\boldsymbol{x}\right], \tag{60}$$

$$= \mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[-\boldsymbol{I} - \boldsymbol{0}\times\tilde{\boldsymbol{x}}^\top\right], \tag{61}$$

$$= -\boldsymbol{I}. \tag{62}$$

The second expectation also gets simplified on one of its terms:

$$\mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[-\left(\frac{\nabla_{\boldsymbol{x}}p(\boldsymbol{x})}{p(\boldsymbol{x})}\right)^\top(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\right] = -\int_{\mathcal{X}}\mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}\left[(\nabla_{\boldsymbol{x}}p_{\text{data}}(\boldsymbol{x}))^\top(\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top\right]\mathrm{d}\boldsymbol{x}, \tag{63}$$

$$= -\int_{\mathcal{X}}(\nabla_{\boldsymbol{x}}p_{\text{data}}(\boldsymbol{x}))^\top(\boldsymbol{x}-\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}})^\top\mathrm{d}\boldsymbol{x}, \tag{64}$$

$$= \boldsymbol{I} + \int_{\mathcal{X}}(\nabla_{\boldsymbol{x}}p_{\text{data}}(\boldsymbol{x}))^\top\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}^\top\mathrm{d}\boldsymbol{x}. \tag{65}$$

Plugging it all together, the derivative of the Fisher loss with respect to $\boldsymbol{G}$ is:

$$\nabla_{\boldsymbol{G}}\mathcal{L}_{\mathcal{F}}(\theta) = 2\boldsymbol{G}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} + (\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}-\boldsymbol{x})(\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}-\boldsymbol{x})^\top\right] + 2\int_{\mathcal{X}}(\nabla_{\boldsymbol{x}}p_{\text{data}}(\boldsymbol{x}))^\top\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}^\top\mathrm{d}\boldsymbol{x} \tag{66}$$

Setting the derivative to zero, we deduce that the optimal solution is:

$$\boxed{\boldsymbol{G} = -\left(\int_{\mathcal{X}}(\nabla_{\boldsymbol{x}}p_{\text{data}}(\boldsymbol{x}))^\top\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}^\top\mathrm{d}\boldsymbol{x}\right)\times\left(\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}} + (\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}-\boldsymbol{x})(\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}|\boldsymbol{x}}-\boldsymbol{x})^\top\right]\right)^{-1}}. \tag{67}$$

B.2 Proof of Theorem 2.4

We start by computing the actual value of the Fisher loss for any smooth data distribution $p_{\text{data}}(\boldsymbol{x})$. In addition, we assume that the augmentations are now Gaussian-distributed: $\tilde{\boldsymbol{x}}\mid\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{x}+\boldsymbol{\mu}_{\boldsymbol{\varepsilon}},\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}})$. We can unfold the value of the Fisher loss:

$$\mathcal{L}_{\mathcal{F}}(\theta) = \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\|\nabla_{\boldsymbol{x}}\log p(\tilde{\boldsymbol{x}}\mid\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})\|^2\right], \tag{68}$$

$$= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\|-(\boldsymbol{x}+\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}-\tilde{\boldsymbol{x}})^\top\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} + (\boldsymbol{x}-\tilde{\boldsymbol{x}})\boldsymbol{G}\|^2\right], \tag{69}$$

$$= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\|\boldsymbol{\varepsilon}^\top\left(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}-\boldsymbol{G}\right) - \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}\|\right]. \tag{70}$$

We can then develop and compute the expectation using the trick of the trace operator to replace the dot product. Thus:

$$\mathcal{L}_{\mathcal{F}}(\theta) = \mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[\boldsymbol{\varepsilon}^\top(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - \boldsymbol{G})^2\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^\top(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - \boldsymbol{G})\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} - \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - \boldsymbol{G})\boldsymbol{\varepsilon}\right] + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-2}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \tag{71}$$

$$= \mathrm{Tr}\left((\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - \boldsymbol{G})^2(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)\right) - \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - \boldsymbol{G})\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} - \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - \boldsymbol{G})\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-2}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \tag{72}$$

$$= \mathrm{Tr}\left((\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - \boldsymbol{G})^2\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}\right) + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\boldsymbol{G}^2\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} \tag{73}$$

$$= \mathrm{Tr}\left(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - 2\boldsymbol{G} + \boldsymbol{G}^2(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)\right) \tag{74}$$

Recalling Corollary 2.2, we know that the optimal solution to the Fisher loss is:

$$\boldsymbol{W}^\top\boldsymbol{W} = \boldsymbol{G} = (\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)^{-1} \tag{75}$$

leading to the minimal value:

$$\mathcal{L}_{\mathcal{F}}(\theta) = \mathrm{Tr}\left(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} - (\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)^{-1}\right). \tag{76}$$

Now, we define the random variables $\boldsymbol{u} = \boldsymbol{x} + \sqrt{t}\boldsymbol{w}$, with $t \geq 0$, for $\boldsymbol{W}$ any zero-mean white Gaussian noise. We note $p^t(\boldsymbol{u} \mid \tilde{\boldsymbol{x}})$ (resp. $q_\theta^t(\boldsymbol{u} \mid \tilde{\boldsymbol{x}})$) the distribution of $\boldsymbol{u}$ for a given t when $\boldsymbol{x}$ is sampled from $p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ (resp. $q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$. Then, following Lyu (2009, Theorem 1), we have $\forall\tilde{\boldsymbol{x}}$:

$$\frac{\mathrm{d}}{\mathrm{d}t}D_{\mathrm{KL}}(p^t(\boldsymbol{u} \mid \tilde{\boldsymbol{x}})\|q_\theta^t(\boldsymbol{u} \mid \tilde{\boldsymbol{x}}))\Big|_{t=0} = D_{\mathcal{F}}(p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})\|q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})). \tag{77}$$

We can wrap this expression in an expectation over all augmentations $\tilde{\boldsymbol{x}}$, then place the gradient operation outside. We recognize:

$$\mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[\frac{\mathrm{d}}{\mathrm{d}t}D_{\mathrm{KL}}(p^t(\boldsymbol{u} \mid \tilde{\boldsymbol{x}})\|q_\theta^t(\boldsymbol{u} \mid \tilde{\boldsymbol{x}}))\Big|_{t=0}\right] = \frac{\mathrm{d}}{dt}\mathcal{L}_{\mathrm{KL}}^t(\theta)\Big|_{t=0}, \tag{78}$$

and:

$$\mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[D_{\mathcal{F}}(p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})\|q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}))\right] = \mathcal{L}_{\mathcal{F}}(\theta). \tag{79}$$

We deduce:

$$\frac{\mathrm{d}}{dt}\mathcal{L}_{\mathrm{KL}}^t(\theta)\Big|_{t=0} = \mathcal{L}_{\mathcal{F}}(\theta). \tag{80}$$

The minimal value of the Fisher objective is by definition non-negative. To cancel it, we must impose $\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} = \boldsymbol{0}$. This implies:

$$\frac{\mathrm{d}}{dt}\mathcal{L}_{\mathrm{KL}}^t(\theta)\Big|_{t=0} = 0. \tag{81}$$

Since the variations of the KL objective are null, we reached its minimum. In other words, for any smooth data distribution decaying fast enough, and an augmentation $\tilde{\boldsymbol{x}} \mid \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}})$, the optimal solution:

$$\boldsymbol{G} = (\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)^{-1} = \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}, \tag{82}$$

is also a minimizer of the KL objective.

B.3 Proof of Theorem 2.5

The KL loss is expressed:

$$\mathcal{L}_{\mathrm{KL}}(\theta) = C + \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})\right] + \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\log Z_{\tilde{\boldsymbol{x}}}\right]. \tag{83}$$

The joint expectation of the energy is straightforward:

$$\begin{aligned}
\mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})\right] &= \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|^2_{\boldsymbol{G}}\right], &(84)\\
&= \frac{1}{2}\mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[\|\boldsymbol{\varepsilon}\|^2_{\boldsymbol{G}}\right], &(85)\\
&= \frac{1}{2}\operatorname{Tr}\left(\boldsymbol{G}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)\right). &(86)
\end{aligned}$$

We then remind that the normalising constant is an expectation over $p_{\mathrm{data}}(\boldsymbol{x})$, owing to the tilt:

$$Z_{\tilde{\boldsymbol{x}}} = \mathbb{E}_{p_{\mathrm{data}}(\boldsymbol{x})}\left[\exp\left(-\frac{1}{2}\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|^2_{\boldsymbol{G}}\right)\right]. \tag{87}$$

We can leverage a Gaussian integral by unfolding the definition of $p_{\mathrm{data}}(\boldsymbol{x})$, which is a Gaussian distribution:

$$\begin{aligned}
Z_{\tilde{\boldsymbol{x}}} &= \int_{\mathbb{R}^d} \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma}_{\boldsymbol{x}})}} \exp\left(-\frac{1}{2}\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|^2_{\boldsymbol{G}} - \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{x}}\|^2_{\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}}\right) \mathrm{d}\boldsymbol{x}, &(88)\\
&= \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma}_{\boldsymbol{x}})}} \int_{\mathbb{R}^d} \exp\left(-\left[\frac{1}{2}\boldsymbol{x}^\top(\boldsymbol{G}+\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})\boldsymbol{x} - (\tilde{\boldsymbol{x}}^\top\boldsymbol{G} + \boldsymbol{\mu}_{\boldsymbol{x}}^\top\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})\boldsymbol{x} + \frac{1}{2}\tilde{\boldsymbol{x}}^\top\boldsymbol{G}\tilde{\boldsymbol{x}} + \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}\right]\right) \mathrm{d}\boldsymbol{x}. &(89)
\end{aligned}$$

We note here that we do not even need $\boldsymbol{G}$ to be positive definite to carry the Gaussian integral. Indeed, if $\boldsymbol{G}$ is positive semi-definite, the sum $\boldsymbol{G}+\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}$ remains positive definite because the precision of the data is positive definite. The result of the integral is therefore:

$$\begin{aligned}
Z_{\tilde{\boldsymbol{x}}} = \frac{1}{\sqrt{\det(2\pi\boldsymbol{\Sigma}_{\boldsymbol{x}})}} &\sqrt{\det\left(2\pi(\boldsymbol{G}+\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})^{-1}\right)} \\
&\times \exp\left(\frac{1}{2}(\tilde{\boldsymbol{x}}^\top\boldsymbol{G} + \boldsymbol{\mu}_{\boldsymbol{x}}^\top\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})(\boldsymbol{G}+\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})^{-1}(\boldsymbol{G}\tilde{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}) - \frac{1}{2}\tilde{\boldsymbol{x}}^\top\boldsymbol{G}\tilde{\boldsymbol{x}} - \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}\right). \quad (90)
\end{aligned}$$

We begin by simplifying the determinants:

$$\begin{aligned}
\sqrt{\frac{\det\left(2\pi(\boldsymbol{G}+\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})^{-1}\right)}{\det(2\pi\boldsymbol{\Sigma}_{\boldsymbol{x}})}} &= \sqrt{\frac{\det((\boldsymbol{G}+\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})^{-1})}{\det(\boldsymbol{\Sigma}_{\boldsymbol{x}})}}, &(91)\\
&= \sqrt{\frac{1}{\det(\boldsymbol{\Sigma}_{\boldsymbol{x}})\det(\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}+\boldsymbol{G})}}, &(92)\\
&= \frac{1}{\sqrt{\det(\boldsymbol{I}+\boldsymbol{\Sigma}_{\boldsymbol{x}}\boldsymbol{G})}}. &(93)
\end{aligned}$$

For the exponent, we will reduce the expression by introducing the variable $\boldsymbol{A} = \boldsymbol{G} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}$. We can then conveniently express $\boldsymbol{G}$ using $\boldsymbol{A}$ and $\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}$. This means:

$$Z_{\tilde{\boldsymbol{x}}} \propto \exp\left(\frac{1}{2}(\tilde{\boldsymbol{x}}^\top \boldsymbol{G} + \boldsymbol{\mu}_{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})\boldsymbol{A}^{-1}(\boldsymbol{G}\tilde{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}) - \frac{1}{2}\tilde{\boldsymbol{x}}^\top \boldsymbol{G}\tilde{\boldsymbol{x}} - \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}\right), \tag{94}$$

$$= \exp\left(\frac{1}{2}(\tilde{\boldsymbol{x}}^\top (\boldsymbol{A} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}) + \boldsymbol{\mu}_{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})\boldsymbol{A}^{-1}((\boldsymbol{A} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})\tilde{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}) - \frac{1}{2}\tilde{\boldsymbol{x}}^\top \boldsymbol{G}\tilde{\boldsymbol{x}} - \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}\right), \tag{95}$$

$$= \exp\big(\frac{1}{2}\tilde{\boldsymbol{x}}^\top \boldsymbol{A}\tilde{\boldsymbol{x}} + \frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}})^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}}) + \frac{1}{2}\tilde{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}}) + \frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}})^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}}$$
$$- \frac{1}{2}\tilde{\boldsymbol{x}}^\top \boldsymbol{G}\tilde{\boldsymbol{x}} - \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}\big). \tag{96}$$

There, we can break down $\boldsymbol{A} = \boldsymbol{G} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}$ to further factorise:

$$Z_{\tilde{\boldsymbol{x}}} \propto \exp\left(\frac{1}{2}\tilde{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}} + \frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}})^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}}) + \frac{1}{2}\tilde{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}}) + \frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}})^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}} \right. \tag{97}$$
$$\left. - \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}\right),$$

$$= \exp\left(\frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}})^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}}) + \frac{1}{2}\tilde{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}} + \frac{1}{2}(\boldsymbol{\mu}_{\boldsymbol{x}} - \tilde{\boldsymbol{x}})^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\tilde{\boldsymbol{x}} - \frac{1}{2}\boldsymbol{\mu}_{\boldsymbol{x}}^\top \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu}_{\boldsymbol{x}}\right), \tag{98}$$

$$= \exp\left(\frac{1}{2}(\tilde{\boldsymbol{x}} - \boldsymbol{\mu}_{\boldsymbol{x}})^\top \left[\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\right](\tilde{\boldsymbol{x}} - \boldsymbol{\mu}_{\boldsymbol{x}})\right). \tag{99}$$

Now, the expectation of the log of the normalisation constant becomes easier:

$$\mathbb{E}_{p(\tilde{\boldsymbol{x}})}[\log Z_{\tilde{\boldsymbol{x}}}] = -\frac{1}{2}\log\det(\boldsymbol{I} + \boldsymbol{G}\boldsymbol{\Sigma}_{\boldsymbol{x}}) + \mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}(\tilde{\boldsymbol{x}} - \boldsymbol{\mu}_{\boldsymbol{x}})^\top \left[\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\right](\tilde{\boldsymbol{x}} - \boldsymbol{\mu}_{\boldsymbol{x}})\right], \tag{100}$$

$$= -\frac{1}{2}\log\det(\boldsymbol{I} + \boldsymbol{G}\boldsymbol{\Sigma}_{\boldsymbol{x}}) + \frac{1}{2}\operatorname{Tr}\left(\left[\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\right]\mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[(\tilde{\boldsymbol{x}} - \boldsymbol{\mu}_{\boldsymbol{x}})(\tilde{\boldsymbol{x}} - \boldsymbol{\mu}_{\boldsymbol{x}})^\top\right]\right), \tag{101}$$

$$= -\frac{1}{2}\log\det(\boldsymbol{I} + \boldsymbol{G}\boldsymbol{\Sigma}_{\boldsymbol{x}}) + \frac{1}{2}\operatorname{Tr}\left(\left[\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\right]\left[\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}} + (\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}} - \boldsymbol{\mu}_{\boldsymbol{x}})(\boldsymbol{\mu}_{\tilde{\boldsymbol{x}}} - \boldsymbol{\mu}_{\boldsymbol{x}})^\top\right]\right). \tag{102}$$

There, we remind that $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}} = \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\Sigma}_{\boldsymbol{x}}$, and the same expression goes for the means. This leads to:

$$\mathbb{E}_{p(\tilde{\boldsymbol{x}})}[\log Z_{\tilde{\boldsymbol{x}}}] = -\frac{1}{2}\log\det(\boldsymbol{I} + \boldsymbol{G}\boldsymbol{\Sigma}_{\boldsymbol{x}}) + \frac{1}{2}\operatorname{Tr}\left(\left[\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\right]\left[\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\right]\right). \tag{103}$$

The total loss is therefore:

$$\mathcal{L}_{\mathrm{KL}}(\theta) = \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\log\frac{p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}{p_{\mathrm{data}}(\boldsymbol{x})}\right] + \frac{1}{2}\operatorname{Tr}\left(\boldsymbol{G}[\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top]\right) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} + \boldsymbol{G}) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}_{\boldsymbol{x}})$$
$$+ \frac{1}{2}\operatorname{Tr}\left(\left[\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}(\boldsymbol{G} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1})^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\right]\left[\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\right]\right). \tag{104}$$

To find its minimizer, we will express the loss using the variable $\boldsymbol{A} = \boldsymbol{G} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}$ instead of $\boldsymbol{G}$. This notably facilitates the derivative of the log determinant. The newly expressed loss is:

$$\mathcal{L}_{\mathrm{KL}}(\theta) = \mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\log \frac{p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}{p_{\mathrm{data}}(\boldsymbol{x})}\right] + \frac{1}{2}\operatorname{Tr}\left([\boldsymbol{A} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}][\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top]\right) - \frac{1}{2}\log\det(\boldsymbol{A}) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}_{\boldsymbol{x}}) + \frac{1}{2}\operatorname{Tr}\left(\left[\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\right]\left[\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\right]\right). \quad (105)$$

The derivation with respect to $\boldsymbol{A}$ is then straightforward:

$$\nabla_{\boldsymbol{A}}\mathcal{L}_{\mathrm{KL}}(\theta) = \frac{1}{2}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top) - \frac{1}{2}\boldsymbol{A}^{-1} - \frac{1}{2}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\left[\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\right]\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1}. \quad (106)$$

Setting it to zero, we have the following equivalences:

$$\frac{1}{2}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top) - \frac{1}{2}\boldsymbol{A}^{-1} - \frac{1}{2}\boldsymbol{A}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\left[\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top\right]\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{A}^{-1} = \boldsymbol{0}, \quad (107)$$

$$\boldsymbol{A}^{-1}\left[\boldsymbol{A}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)\boldsymbol{A} - \boldsymbol{A} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}[\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top]\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1}]\right]\boldsymbol{A}^{-1} = \boldsymbol{0}. \quad \iff \quad (108)$$

To get it equal to 0, cancelling the inner factor suffices. There, we can unfold the definition of $\boldsymbol{A}$ and solve the equation on $\boldsymbol{G}$ instead:

$$\boldsymbol{A}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)\boldsymbol{A} - \boldsymbol{A} - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}[\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top]\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} = \boldsymbol{0}, \quad (109)$$

$$(\boldsymbol{G} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)(\boldsymbol{G} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}) - (\boldsymbol{G} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}) - \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}[\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top]\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} = \boldsymbol{0}, \quad \iff \quad (110)$$

$$\boldsymbol{G}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)\boldsymbol{G} + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)\boldsymbol{G} + \boldsymbol{G}(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} - \boldsymbol{G} - 2\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} = \boldsymbol{0}. \quad \iff \quad (111)$$

This is solved for:

$$\boxed{\boldsymbol{G} = (\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)^{-1}.} \quad (112)$$

We have thus again the same result as in all previous case, yet the interpretation of this EBM is closer to the Bayesian posterior, and strictly equal when $\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} = \boldsymbol{0}$ because of the tilt.

We showed that for both the Fisher and KL loss, the optimal solution for $\boldsymbol{G}$ is to be equal to $(\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}^\top)^{-1}$. For the Fisher loss, we did not require $\boldsymbol{G}$ to be positive definite to find the optimal solution. For the KL loss, the tilt introduced a precision matrix allowing us to compute the Gaussian integral, even if $\boldsymbol{G}$ were positive definite. Therefore, we have found the actual global optimum for both losses.

It follows that $\boldsymbol{G}$ cannot be equal to this global optimum if $\boldsymbol{W}$ is low-rank due to the output dimension p being smaller than the input dimension d.

B.4 Proof of Theorem 2.6

Even though the theorem was expressed using the representations $\boldsymbol{z}$, we will express the demonstration here using the input data $\boldsymbol{x}$ without loss of generality. The purpose is to make a better parallel with previous theorems.

We start by plugging all the definitions into the distance δ. We have $\boldsymbol{x} \sim \mathcal{U}(\mathbb{S}^{d-1})$, *i.e.* the uniform distribution on the hypersphere. The data is augmented with $\tilde{\boldsymbol{x}} \mid \boldsymbol{x} \sim \mathrm{vMF}(\boldsymbol{x} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \beta)$. By identification to the previous theorems, the neural network ψ_θ is the identity mapping. In such a setup, there are no parameters θ to optimise, and the temperature-scaled Euclidean distance is directly the cosine distance because all variables lie on the hypersphere:

$$\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{2\tau}\|\psi_\theta(\boldsymbol{x}) - \psi_\theta(\tilde{\boldsymbol{x}})\|^2, \tag{113}$$

$$= \frac{1}{2\tau}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2, \tag{114}$$

$$= \frac{1}{\tau}\left(1 - \boldsymbol{x}^\top \tilde{\boldsymbol{x}}\right). \tag{115}$$

We can then compute the KL loss. We start by the expectation of the distance over the joint expectation:

$$\mathbb{E}_{p(\boldsymbol{x},\tilde{\boldsymbol{x}})}[\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})] = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}\left[\tau^{-1}\left(1 - \boldsymbol{x}^\top \tilde{\boldsymbol{x}}\right)\right]\right], \tag{116}$$

$$= \tau^{-1} - \tau^{-1}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[A_d(\beta)\boldsymbol{x}^\top \boldsymbol{x}\right], \tag{117}$$

$$= \tau^{-1}\left(1 - A_d(\beta)\right), \tag{118}$$

where $A_d(\beta) = \frac{I_{\frac{d}{2}}(\beta)}{I_{\frac{d}{2}-1}(\beta)}$, with I the modified Bessel function of the first kind. We follow with the normalisation constant, where we can write the uniform distribution as von Mises Fisher distribution with concentration 0 for the data:

$$Z_{\tilde{\boldsymbol{x}}} = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\exp\left(-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})\right)\right], \tag{119}$$

$$= e^{-\frac{1}{\tau}}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[\exp\left(\frac{\boldsymbol{x}^\top \tilde{\boldsymbol{x}}}{\tau}\right)\right], \tag{120}$$

$$= e^{-\frac{1}{\tau}}\int_{\mathcal{X}} C_d(0)\exp\left(\frac{\boldsymbol{x}^\top \tilde{\boldsymbol{x}}}{\tau}\right)\mathrm{d}\boldsymbol{x}, \tag{121}$$

where:

$$C_d(\beta) = \frac{\beta^{\frac{d}{2}-1}}{(2\pi)^{\frac{d}{2}} I_{\frac{d}{2}-1}(\beta)}, \beta > 0, \qquad C_d(0) = \frac{\Gamma\left(\frac{d}{2}\right)}{(2\pi)^{\frac{d}{2}}}. \tag{122}$$

We may identify in the integral the density of a von Mises Fisher distribution, which concentration is controlled by the temperature τ. This allows us to directly compute the integral:

$$Z_{\tilde{\boldsymbol{x}}} = e^{-\frac{1}{\tau}}\int_{\mathcal{X}} C_d(0) \times \frac{C_d(\tau^{-1})}{C_d(\tau^{-1})}\exp\left(\frac{\boldsymbol{x}^\top \tilde{\boldsymbol{x}}}{\tau}\right)\mathrm{d}\boldsymbol{x}, \tag{123}$$

$$= e^{-\frac{1}{\tau}}\frac{C_d(0)}{C_d(\tau^{-1})}\int_{\mathcal{X}} f_{\text{vMF}}(\boldsymbol{x}; \tilde{\boldsymbol{x}}; \tau^{-1})\mathrm{d}\boldsymbol{x}, \tag{124}$$

$$= e^{-\frac{1}{\tau}}\frac{C_d(0)}{C_d(\tau^{-1})}. \tag{125}$$

We can finally deduce the expectation of the log constant, which is straightforward because it does not depend on the augmentation:

$$\mathbb{E}_{p(\tilde{\boldsymbol{x}})}\left[\log Z_{\tilde{\boldsymbol{x}}}\right] = -\frac{1}{\tau} + \log\Gamma\left(\frac{d}{2}\right) - \left(\frac{d}{2} - 1\right)\log\tau^{-1} + \log I_{\frac{d}{2}-1}(\tau^{-1}). \tag{126}$$

Therefore, the KL loss is:

$$\mathcal{L}_{\mathrm{KL}}(\beta) = \tau^{-1}\left(1 - A_d(\beta)\right) - \frac{1}{\tau} + \log\Gamma\left(\frac{d}{2}\right) - \left(\frac{d}{2} - 1\right)\log\tau^{-1} + \log I_{\frac{d}{2}-1}(\tau^{-1}). \tag{127}$$

As previously mentioned, there are no parameters of a model θ in this case. However, the function depends instead here the conditional concentration β of the augmentation. We may optimise the KL loss with respect to this concentration. The only term that depends on β is the first positive factor, which can be cancelled when $A_d(\beta) = 1$. This value is the limit of A_d when $\beta \longrightarrow \infty$.

C Noise contrastive estimation

C.1 Noise Contrastive Estimation (NCE)

In this section, we derive the noise contrastive estimator (NCE) for the conditional distribution $q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$. We follow the latent variable approach of Gutmann and Hyvärinen (2010), adapted to our conditional setting.

Let $\tilde{\boldsymbol{x}}$ be an anchor. We frame the estimation as a binary classification problem: given a sample $\boldsymbol{x}$, determine whether it was drawn from the true conditional $p_{\mathrm{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ or from a noise distribution $p_\zeta(\boldsymbol{x})$.

To that end, a binary latent variable $y \in \{0, 1\}$ is introduced with prior given by parameter π: $p(y = 1) = \pi$ and $p(y = 0) = 1 - \pi$. The joint distribution factorises as:

$$p(\boldsymbol{x}, y \mid \tilde{\boldsymbol{x}}) = p(y) \cdot p(\boldsymbol{x} \mid y, \tilde{\boldsymbol{x}}), \tag{128}$$

where the conditional likelihood is:

$$p(\boldsymbol{x} \mid y, \tilde{\boldsymbol{x}}) = \begin{cases} p_{\mathrm{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) & \text{if } y = 1, \\ p_\zeta(\boldsymbol{x}) & \text{if } y = 0. \end{cases} \tag{129}$$

Equivalently, we can write the joint as:

$$p(\boldsymbol{x}, y \mid \tilde{\boldsymbol{x}}) = \pi p_{\mathrm{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})\mathbb{1}[y = 1] + (1 - \pi)p_\zeta(\boldsymbol{x})\mathbb{1}[y = 0]. \tag{130}$$

Given a sample $\boldsymbol{x}$ and anchor $\tilde{\boldsymbol{x}}$, applying Bayes' theorem yields the posterior probability that $\boldsymbol{x}$ is a positive:

$$p(y = 1 \mid \boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{p(y = 1) \cdot p(\boldsymbol{x} \mid y = 1, \tilde{\boldsymbol{x}})}{p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}, \tag{131}$$

$$= \frac{\pi p_{\mathrm{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}{\pi p_{\mathrm{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) + (1 - \pi)p_\zeta(\boldsymbol{x})}. \tag{132}$$

We can express this compactly using the logit function. Define the true logit:

$$\ell(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \triangleq \log\frac{p(y = 1 \mid \boldsymbol{x}, \tilde{\boldsymbol{x}})}{p(y = 0 \mid \boldsymbol{x}, \tilde{\boldsymbol{x}})} = \log\frac{\pi}{1 - \pi} + \log\frac{p_{\mathrm{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}{p_\zeta(\boldsymbol{x})}. \tag{133}$$

Then we recover the posterior distributions for both classes $p(y = 1 \mid \boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sigma(\ell(\boldsymbol{x}, \tilde{\boldsymbol{x}}))$ and $p(y = 0 \mid \boldsymbol{x}, \tilde{\boldsymbol{x}}) = \sigma(-\ell(\boldsymbol{x}, \tilde{\boldsymbol{x}}))$, where $\sigma(u) = (1 + e^{-u})^{-1}$ is the sigmoid function.

We approximate $p_{\mathrm{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$ with a tilted EBM to mimic the approach used throughout the paper:

$$q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \frac{p_{\mathrm{data}}(\boldsymbol{x})\exp(-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}))}{Z_{\tilde{\boldsymbol{x}}}}. \tag{134}$$

For the noise distribution, we make the canonical choice $p_\zeta(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x})$. The parametric logit becomes:

$$\ell_\theta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \log \frac{\pi}{1-\pi} + \log \frac{q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}{p_\zeta(\boldsymbol{x})}, \tag{135}$$

$$= \log \frac{\pi}{1-\pi} + \log \frac{p_{\text{data}}(\boldsymbol{x}) \exp(-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}))/Z_{\tilde{\boldsymbol{x}}}}{p_{\text{data}}(\boldsymbol{x})}, \tag{136}$$

$$= \log \frac{\pi}{1-\pi} - \delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) - \log Z_{\tilde{\boldsymbol{x}}}. \tag{137}$$

Note that $\log p_{\text{data}}(\boldsymbol{x})$ cancels due to both the tilt structure and our choice of noise distribution. Following Gutmann and Hyvärinen (2010), we learn the log-normalising constant with a neural network $\lambda_\varphi(\tilde{\boldsymbol{x}}) \approx \log Z_{\tilde{\boldsymbol{x}}}$:

$$\ell_{\theta,\varphi}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \log \frac{\pi}{1-\pi} - \delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) - \lambda_\varphi(\tilde{\boldsymbol{x}}). \tag{138}$$

The NCE objective is the cross-entropy loss for the binary classification task. Given a sample $(\boldsymbol{x}, y)$ drawn from the generative model, the log-loss is:

$$-\log p_{\theta,\varphi}(y \mid \boldsymbol{x}, \tilde{\boldsymbol{x}}) = -y \log \sigma(\ell_{\theta,\varphi}(\boldsymbol{x}, \tilde{\boldsymbol{x}})) - (1-y) \log \sigma(-\ell_{\theta,\varphi}(\boldsymbol{x}, \tilde{\boldsymbol{x}})). \tag{139}$$

Taking the expectation over the joint distribution $p(\boldsymbol{x}, y \mid \tilde{\boldsymbol{x}})$:

$$\mathbb{E}_{p(\boldsymbol{x},y|\tilde{\boldsymbol{x}})} \left[-\log p_{\theta,\varphi}(y \mid \boldsymbol{x}, \tilde{\boldsymbol{x}})\right] = -\mathbb{E}_{p(y=1)p(\boldsymbol{x}|y=1,\tilde{\boldsymbol{x}})} \left[\log \sigma(\ell_{\theta,\varphi})\right] - \mathbb{E}_{p(y=0)p(\boldsymbol{x}|y=0,\tilde{\boldsymbol{x}})} \left[\log \sigma(-\ell_{\theta,\varphi})\right], \tag{140}$$

$$= -\pi \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\tilde{\boldsymbol{x}})} \left[\log \sigma(\ell_{\theta,\varphi})\right] - (1-\pi) \mathbb{E}_{p_\zeta(\boldsymbol{x})} \left[\log \sigma(-\ell_{\theta,\varphi})\right]. \tag{141}$$

Finally, taking the expectation over anchors $\tilde{\boldsymbol{x}} \sim p(\tilde{\boldsymbol{x}})$ and using our choice $p_\zeta(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x})$:

$$\boxed{\mathcal{L}_{\text{NCE}}(\theta, \varphi) = -\pi \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}, \tilde{\boldsymbol{x}})} \left[\log \sigma(\ell_{\theta,\varphi}(\boldsymbol{x}, \tilde{\boldsymbol{x}}))\right] - (1-\pi) \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}) p(\tilde{\boldsymbol{x}})} \left[\log \sigma(-\ell_{\theta,\varphi}(\boldsymbol{x}, \tilde{\boldsymbol{x}}))\right].} \tag{142}$$

The NCE objective trains a classifier to distinguish positive pairs $(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \sim p_{\text{data}}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ from negative pairs $(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \sim p_{\text{data}}(\boldsymbol{x}) p(\tilde{\boldsymbol{x}})$. The first term encourages high logits for positives; the second encourages low logits for negatives.

C.1.1 Convergence to maximum likelihood

As the number of noise samples $k = \frac{1-\pi}{\pi}$ increases, NCE converges to MLE. Let $\pi = \frac{1}{1+k}$, so $\log \frac{\pi}{1-\pi} = -\log k$ and the logit becomes:

$$\ell_{\theta,\varphi}(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = -\log k - \delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) - \lambda_\varphi(\tilde{\boldsymbol{x}}). \tag{143}$$

As $k \to \infty$, the logit $\ell \to -\infty$ for all $\boldsymbol{x}$. Using the asymptotic expansions $\sigma(u) \approx e^u$ for $u \ll 0$ and $\sigma(u) \approx 1$ for $u \gg 0$, the positive term becomes $-\log \sigma(\ell) \approx -\ell = \log k + \delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \lambda_\varphi(\tilde{\boldsymbol{x}})$, while the negative term becomes $-\log \sigma(-\ell) \approx e^\ell = \frac{e^{-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}) - \lambda_\varphi(\tilde{\boldsymbol{x}})}}{k}$.

Rescaling the objective $\frac{1}{\pi}\mathcal{L}_{\text{NCE}} = (1+k)\mathcal{L}_{\text{NCE}}$ leads to:

$$\frac{1}{\pi}\mathcal{L}_{\text{NCE}} \approx \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}, \tilde{\boldsymbol{x}})} \left[\log k + \delta + \lambda_\varphi\right] + k \cdot \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}) p(\tilde{\boldsymbol{x}})} \left[\frac{e^{-\delta - \lambda_\varphi}}{k}\right], \tag{144}$$

$$= \log k + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}, \tilde{\boldsymbol{x}})} \left[\delta + \lambda_\varphi\right] + \mathbb{E}_{p(\tilde{\boldsymbol{x}})} \left[\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[e^{-\delta - \lambda_\varphi}\right]\right]. \tag{145}$$

Taking the derivative with respect to $\lambda_\varphi(\tilde{\boldsymbol{x}})$ and setting to zero leads to the following equation:

$$1 - e^{-\lambda_\varphi(\tilde{\boldsymbol{x}})} \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})} \left[e^{-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}})}\right] = 0. \tag{146}$$

Solving in $\lambda^*_\varphi(\tilde{\boldsymbol{x}})$ gives:

$$\lambda^*_\varphi(\tilde{\boldsymbol{x}}) = \log \mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\left[e^{-\delta(\boldsymbol{x},\tilde{\boldsymbol{x}})}\right] = \log Z_{\tilde{\boldsymbol{x}}}. \tag{147}$$

Substituting the optimal normalizer and dropping an additive constant independent of θ, we recover negative log-likelihood (MLE objective):

$$\boxed{\lim_{k\to\infty}\left((k+1)\mathcal{L}_{\text{NCE}}(\theta,\varphi^*) - \log k\right) = \mathbb{E}_{p_{\text{data}}(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}) + \log Z_{\tilde{\boldsymbol{x}}}\right] = -\mathbb{E}_{p_{\text{data}}(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\log q_\theta(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})\right].} \tag{148}$$

C.2 Info-NCE

We now derive Info-NCE following the same derivation as NCE but with a *categorical* parametrisation.

Let $\tilde{\boldsymbol{x}}$ be a fixed anchor. We consider c candidate samples $\{\boldsymbol{x}_1,\dots,\boldsymbol{x}_c\}$, exactly one of which is the true positive datum that led to the augmentation $\tilde{\boldsymbol{x}}$. We frame estimation as a c-way classification problem: identify which sample is the positive.

We introduce a categorical latent variable $y \in \{1,\dots,c\}$ with uniform prior $p(y=i) = \frac{1}{c}$. The joint distribution factorises as:

$$p(\boldsymbol{x}_1,\dots,\boldsymbol{x}_c, y \mid \tilde{\boldsymbol{x}}) = p(y)\cdot p(\boldsymbol{x}_1,\dots,\boldsymbol{x}_c \mid y,\tilde{\boldsymbol{x}}), \tag{149}$$

where the conditional likelihood is:

$$p(\boldsymbol{x}_1,\dots,\boldsymbol{x}_c \mid y,\tilde{\boldsymbol{x}}) = p_{\text{data}}(\boldsymbol{x}_y \mid \tilde{\boldsymbol{x}})\cdot\prod_{j\neq y} p_\zeta(\boldsymbol{x}_j). \tag{150}$$

The generative process is:

1. Sample the positive index: $y \sim \text{Uniform}(\{1,\dots,c\})$
2. Draw the positive: $\boldsymbol{x}_y \sim p_{\text{data}}(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})$
3. Draw negatives independently: $\boldsymbol{x}_j \sim p_\zeta(\boldsymbol{x})$ for $j \neq y$

Given the batch $\boldsymbol{x}_{1:c} = \{\boldsymbol{x}_1,\dots,\boldsymbol{x}_c\}$ and anchor $\tilde{\boldsymbol{x}}$, applying Bayes' theorem:

$$p(y=i \mid \boldsymbol{x}_{1:c},\tilde{\boldsymbol{x}}) = \frac{p(y=i)\cdot p(\boldsymbol{x}_{1:c}\mid y=i,\tilde{\boldsymbol{x}})}{\sum_{k=1}^c p(y=k)\cdot p(\boldsymbol{x}_{1:c}\mid y=k,\tilde{\boldsymbol{x}})}, \tag{151}$$

$$= \frac{\frac{1}{c}\cdot p(\boldsymbol{x}_i\mid\tilde{\boldsymbol{x}})\cdot\prod_{j\neq i} p_\zeta(\boldsymbol{x}_j)}{\sum_{k=1}^c \frac{1}{c}\cdot p_{\text{data}}(\boldsymbol{x}_k\mid\tilde{\boldsymbol{x}})\cdot\prod_{j\neq k} p_\zeta(\boldsymbol{x}_j)}. \tag{152}$$

Dividing numerator and denominator by $\prod_{j=1}^c p_\zeta(\boldsymbol{x}_j)$:

$$p(y=i \mid \boldsymbol{x}_{1:c},\tilde{\boldsymbol{x}}) = \frac{p(\boldsymbol{x}_i\mid\tilde{\boldsymbol{x}})/p_\zeta(\boldsymbol{x}_i)}{\sum_{k=1}^c p_{\text{data}}(\boldsymbol{x}_k\mid\tilde{\boldsymbol{x}})/p_\zeta(\boldsymbol{x}_k)} = \frac{r_i}{\sum_{k=1}^c r_k}, \tag{153}$$

where $r_i = \frac{p(\boldsymbol{x}_i\mid\tilde{\boldsymbol{x}})}{p_\zeta(\boldsymbol{x}_i)}$ is the density ratio for sample i.

As in NCE, we choose the noise distribution to follow the data distribution $p_\zeta(\boldsymbol{x}) = p_{\text{data}}(\boldsymbol{x})$. The posterior simplifies to:

$$p(y=i \mid \boldsymbol{x}_{1:c},\tilde{\boldsymbol{x}}) = \frac{r_i}{\sum_{k=1}^c r_k}, \quad \text{where} \quad r_i = \frac{p(\boldsymbol{x}_i\mid\tilde{\boldsymbol{x}})}{p_{\text{data}}(\boldsymbol{x}_i)}. \tag{154}$$

We approximate $p(\boldsymbol{x}\mid\tilde{\boldsymbol{x}})$ with the same tilted EBM:

$$q_\theta(\boldsymbol{x}\mid\tilde{\boldsymbol{x}}) = \frac{p_{\text{data}}(\boldsymbol{x})\exp(-\delta(\boldsymbol{x},\tilde{\boldsymbol{x}}))}{Z_{\tilde{\boldsymbol{x}}}}. \tag{155}$$

The parametric density ratio becomes:

$$\frac{q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}{p_\zeta(\boldsymbol{x})} = \frac{q_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})}{p_{\text{data}}(\boldsymbol{x})} = \frac{\exp(-\delta(\boldsymbol{x}, \tilde{\boldsymbol{x}}))}{Z_{\tilde{\boldsymbol{x}}}}. \tag{156}$$

The parametric posterior is:

$$p_\theta(y = i \mid \boldsymbol{x}_{1:c}, \tilde{\boldsymbol{x}}) = \frac{e^{-\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}})}/Z_{\tilde{\boldsymbol{x}}}}{\sum_{k=1}^{c} e^{-\delta(\boldsymbol{x}_k, \tilde{\boldsymbol{x}})}/Z_{\tilde{\boldsymbol{x}}}} = \frac{e^{-\delta_i}}{\sum_{k=1}^{c} e^{-\delta_k}}, \tag{157}$$

where $\delta_i = \delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}})$.

Interestingly, the normalising constant $Z_{\tilde{\boldsymbol{x}}}$ cancels in the softmax. This is in contrast to NCE, where $Z_{\tilde{\boldsymbol{x}}}$ appears in the sigmoid and must be learned explicitly.

The Info-NCE objective is the cross-entropy loss for the c-way classification task. Given a batch $(\boldsymbol{x}_{1:c}, y)$ drawn from the generative model, the log-loss is:

$$-\log p_\theta(y \mid \boldsymbol{x}_{1:c}, \tilde{\boldsymbol{x}}) = -\log \frac{e^{-\delta_y}}{\sum_{k=1}^{c} e^{-\delta_k}} = \delta_y + \log \sum_{k=1}^{c} e^{-\delta_k}. \tag{158}$$

Taking the expectation over $y \sim \text{Uniform}(\{1, \ldots, c\})$ and the corresponding batch:

$$\mathbb{E}\left[-\log p_\theta(y \mid \boldsymbol{x}_{1:c}, \tilde{\boldsymbol{x}})\right] = \frac{1}{c} \sum_{i=1}^{c} \mathbb{E}\left[\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}) + \log \sum_{k=1}^{c} e^{-\delta(\boldsymbol{x}_k, \tilde{\boldsymbol{x}})} \,\middle|\, y = i\right]. \tag{159}$$

In practice, we work with batches of c positive pairs $\{(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)\}_{i=1}^{c}$. For each anchor $\tilde{\boldsymbol{x}}_i$, we treat $\boldsymbol{x}_i$ as the positive and $\{\boldsymbol{x}_j\}_{j \neq i}$ as negatives. Strictly, these in-batch negatives are drawn from $p_{\text{data}}(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}_j)$ rather than the marginal $p_{\text{data}}(\boldsymbol{x})$; since $\tilde{\boldsymbol{x}}_j \sim p(\tilde{\boldsymbol{x}})$, they remain marginally distributed as $p_{\text{data}}(\boldsymbol{x})$, so the batched objective is an unbiased estimator in expectation over the batch composition. The Info-NCE objective becomes:

$$\boxed{\mathcal{L}_{\text{InfoNCE}}(\theta) = -\frac{1}{c} \sum_{i=1}^{c} \log \frac{e^{-\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)}}{\sum_{j=1}^{c} e^{-\delta(\boldsymbol{x}_j, \tilde{\boldsymbol{x}}_i)}}.} \tag{160}$$

This formulation shows that the Info-NCE objective trains a classifier to identify the positive among c candidates. The first term $\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)$ penalizes large distances for positive pairs; the log-sum-exp term encourages large distances for negative pairs. It can be written in the same format as Poole et al. (2019) to reveal:

$$\boxed{\mathcal{L}_{\text{InfoNCE}}(\theta) = \frac{1}{c} \sum_{i=1}^{c} \left[\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) + \log \sum_{j=1}^{c} e^{-\delta(\boldsymbol{x}_j, \tilde{\boldsymbol{x}}_i)}\right].} \tag{161}$$

D Additional details for circle disentangling

We give here additional details for implementing our proof of concept on concentric circles. While the KL loss is the same most straightforward to implement, other losses require more care.

D.1 Bias function

Both the SNL and NCE losses require the parametric definition of a bias, which alleviates the explicit computation of the normalisation constant of the EBM. For both losses, we implemented the bias λ_φ as a neural network with 20 hidden nodes, a ReLU activation and a single node output. We used a separate Adam optimizer to update φ, with a learning rate of 0.1.

D.2 Supervisory gradient

The Fisher loss omits the computation of the normalisation constant, but requires in turn the gradient of the augmentation with respect to $\boldsymbol{x}$: $\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$.

We used two augmentations in a row: a rotation matrix $\boldsymbol{R}(u)$, with $u \sim \mathcal{U}(\{-\frac{\pi}{2}, 0, \frac{\pi}{2}\})$ and some noise addition $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}})$. We have:

$$\tilde{\boldsymbol{x}} = \boldsymbol{R}(u)\boldsymbol{x} + \boldsymbol{\varepsilon}. \tag{162}$$

Computing the probability of the augmentation is immediate, since it is a convolution with independent variable R and $\boldsymbol{\varepsilon}$:

$$p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \sum_{u \in \{-\frac{\pi}{2}, 0, \frac{\pi}{2}\}} p_u(u) \times p_{\boldsymbol{\varepsilon}}(\tilde{\boldsymbol{x}} - \boldsymbol{R}(u)\boldsymbol{x}), \tag{163}$$

$$= \frac{1}{3} \sum_{u \in \{-\frac{\pi}{2}, 0, \frac{\pi}{2}\}} \mathcal{N}(\tilde{\boldsymbol{x}} - \boldsymbol{R}(u)\boldsymbol{x} \mid \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}). \tag{164}$$

This is differentiable. This allows us to compute:

$$\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \frac{1}{p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})} \nabla_{\boldsymbol{x}} p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}), \tag{165}$$

$$= \frac{1}{3p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})} \sum_{u \in \{-\frac{\pi}{2}, 0, \frac{\pi}{2}\}} \nabla_{\boldsymbol{x}} \mathcal{N}(\tilde{\boldsymbol{x}} - \boldsymbol{R}(u)\boldsymbol{x} \mid \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}), \tag{166}$$

$$= \frac{1}{3p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})} \sum_{u \in \{-\frac{\pi}{2}, 0, \frac{\pi}{2}\}} \mathcal{N}(\tilde{\boldsymbol{x}} - \boldsymbol{R}(u)\boldsymbol{x} \mid \boldsymbol{\mu}_{\boldsymbol{\varepsilon}}, \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}) \times (\tilde{\boldsymbol{x}} - \boldsymbol{R}(u)\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^{-1} \boldsymbol{R}(u). \tag{167}$$

In practice, we computed $\log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$ and let automatic differentiation return the gradient.

E Augmentations for the Fisher loss

To train EBMs for representation learning with the Fisher loss, we need a supervisory gradient. In our experiments. We distinguish two different augmentations for our experiments based on the data type.

E.1 Tabular augmentations

For tabular data, we compute an exact supervisory gradient. We proposed the following augmentation:

$$\tilde{\boldsymbol{x}} = \boldsymbol{\alpha}\boldsymbol{x} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\alpha} \sim \mathcal{B}(\rho), \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}). \tag{168}$$

In other words, Gaussian noise will be added, whether or not the feature was masked. By considering independence between all features for the noise and masking process, we can derive this expression feature-wise. We first need the expression of the conditional distribution of the expression:

$$p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = p(\tilde{\boldsymbol{x}} \mid m = 0, \boldsymbol{x}) p(m = 0 \mid \boldsymbol{x}) + p(\tilde{\boldsymbol{x}} \mid m = 1, \boldsymbol{x}) p(m = 1 \mid \boldsymbol{x}), \tag{169}$$

$$= (1 - \rho)\mathcal{N}(\tilde{\boldsymbol{x}} \mid 0, \sigma^2) + \rho \mathcal{N}(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}, \sigma^2). \tag{170}$$

```python
# Make the input differentiable
x.requires_grad = True
# Pass through image augmentations
x_hat = A(x)
# Add Gaussian noise, but make sure that the output is detached for
    differentiation purposes
x_augmented = x_hat.detach() + std * torch.randn_like(x_hat)
# Compute the log prob approximate
log_p_aug_given_x = -0.5 * torch.sum(torch.square(x_augmented-x_hat))/(std**2)
# Backprop and get gradient (detach for Fisher loss)
log_p_aug_given_x.backward()
grad = x.grad.detach()
```

Listing 1: Approximating the gradient of the conditional augmentation when using image augmentation in PyTorch.

Then, the gradient with respect to the log-pdf is immediate:

$$\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \frac{1}{p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})} \times \rho \frac{1}{\sigma^2}(\tilde{\boldsymbol{x}} - \boldsymbol{x}) \times \mathcal{N}(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}, \sigma^2). \tag{171}$$

A more practical expression, which avoids computing explicitly the conditional density, can be accessed by dividing by the right-side Gaussian distribution. This gives:

$$\nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \frac{\frac{\rho}{\sigma^2}(\tilde{\boldsymbol{x}} - \boldsymbol{x})}{1 + \frac{1-\rho}{\rho} \exp\left(\frac{\boldsymbol{x}}{2\sigma^2}(\boldsymbol{x} - \tilde{\boldsymbol{x}})\right)}. \tag{172}$$

While we did not explore this in our experiments, different preservation rates and noise scales may be considered for each feature.

E.2 Image augmentations

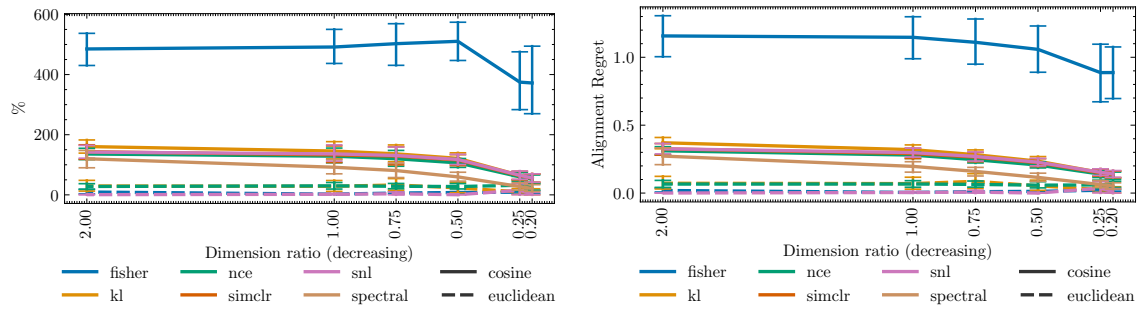
Unlike our choice of augmentations for tabular data, image datasets usually involve multiple transformations in a row, some of which can be invertible. For most of these transformations, *e.g.* black and white colouring or random cropping, the set of possible outputs is countable and finite. Consequently, computing all possible outcomes of a chain of transformations can quickly explode, and is in practice too long to be worth implementing. That is why we will approximate the log probability, then compute the gradient of this approximation with respect to the data $\boldsymbol{x}$.

Let T be the number of transformations in a row. For each augmentation i, there are O_i possible outcomes, including that of not applying the transformation. For instance, black and white colouring is either used or not, so $O = 2$. For an augmentation cropping to one of the quadrant of an image, there are $O = 5$ possible outcomes: not applying it and one per quadrant. We note $\mathcal{A}_{a_i} \dots \mathcal{A}_{a_T}$ each individual augmentation, where a_i is an integer indicating the outcome used for the i-th augmentation. The sequence $\boldsymbol{a}$ thus encodes the entire augmentation in a deterministic manner:

$$\hat{\boldsymbol{x}} = \mathcal{A}_{\boldsymbol{a}}(\boldsymbol{x}) = \mathcal{A}_{a_T} \circ \dots \circ \mathcal{A}_{a_1}(\boldsymbol{x}). \tag{173}$$

To sample an augmentation is then equivalent to sampling the sequence $\boldsymbol{a}$, where each component is either drawn from a Bernoulli distribution or a categorical distribution with fixed user-defined parameters. Finally, we leverage a distribution with similar support to $\boldsymbol{x}$ by adding Gaussian noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \beta^{-1}\boldsymbol{I})$:

$$\tilde{\boldsymbol{x}} = \hat{\boldsymbol{x}} + \boldsymbol{\varepsilon}. \tag{174}$$



(a) Relative improvement (↓) over initialisation

(b) Regret (↓) on best score

Figure 8: Alignment performance of the same model trained with different losses after 20,000 iterations as the output decreases with respect to the number of initial features. The tabular augmentations preserves 80% of the initial features, before adding Gaussian noise. Error bars show standard errors.

By convolution, we have the conditional density of the augmentations:

$$p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = \mathbb{E}_{p(\boldsymbol{a})} \left[\mathcal{N}(\tilde{\boldsymbol{x}} - \mathcal{A}_{\boldsymbol{a}}(\boldsymbol{x}) \mid \boldsymbol{0}, \beta^{-1}\boldsymbol{I}) \right]. \tag{175}$$

To avoid the computationally expensive search over all states $\boldsymbol{a}$, we simply propose to exploit the fast decaying nature of the Gaussian distribution. By taking a set of augmentations such that all deterministic outputs are far away in data space, only one term in the expectation will have a non-negligible value. For instance, we can use horizontal flipping combined with Sobel filtering. However, Gaussian blur is not recommanded to these regards as it may be too close to Gaussian noise. We approximate:

$$p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) \approx \mathcal{N}(\tilde{\boldsymbol{x}} - \mathcal{A}_{\boldsymbol{a}'}(\boldsymbol{x}) \mid \boldsymbol{0}, \beta^{-1}\boldsymbol{I}) p(\boldsymbol{a}'), \tag{176}$$

where $\boldsymbol{a}'$ was the sampled vector state for augmenting. Once we pass the approximation in a log, the probability of this vector state does not matter. Then, automatic differentiation tools can easily leverage a gradient with respect to $\boldsymbol{x}$. We give a simple implementation in PyTorch in Listing 1.

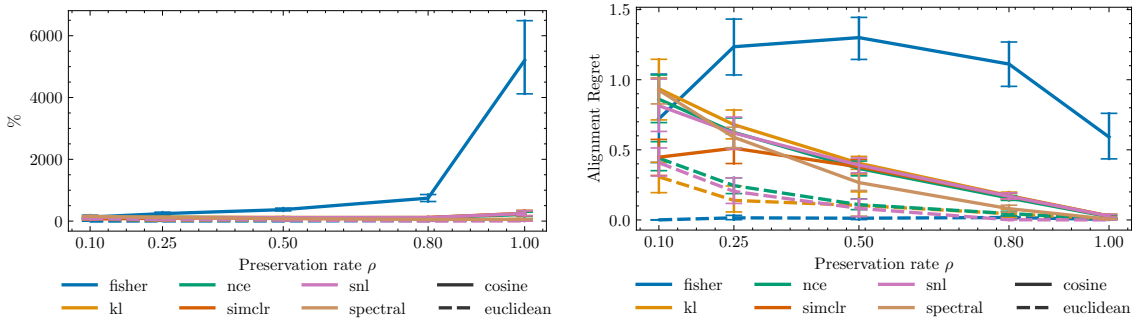
F Extended results for tabular datasets

F.1 Additional metrics

Our extended results using alignment and uniformity metrics (Wang and Isola, 2020) can be respectively found in Figure 8 and Figure 10 when varying output dimensions, and Figure 9 and Figure 11 when varying the feature preservation rate. We remind that the alignment score is in $[0, 1]$, and the lower the better. This means that the relative improvement over the initialisation score must be below 100%. Conversely, the uniformity score must have a relative improvement greater than 100% because it is a negative score for which the lower the better. For completeness, we reported the scores for the Euclidean energy, even though alignment and uniformity are not adequate measures because that energy does not enforce representations to lie on a hypersphere, *e.g.* Figure 2. We focus our analysis on the cosine energy only.

Aligned with our results from Section 4, the Fisher loss with cosine energy is once again the worst performing solution. It neither succeeds at keeping features aligned, as its alignment score increases in figures 9 and 8, nor manages to spread those features with broken relationships on the hypersphere because its uniformity score hardly moves in Figure 11.

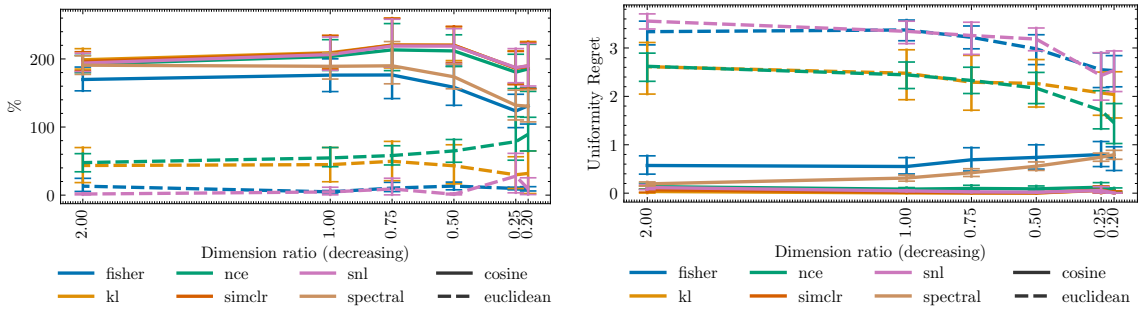
For other losses, we mainly note that NCE and SNL give the best spread of features over the hypersphere, as highlighted by the best uniformity regret in all settings in Figure 11 and Figure 10. Except for strong augmentations with $\rho = 0.1$, SIMCLR is also a strong competitor. However, its alignment is worse than all



(a) Relative improvement (↓) over initialisation

(b) Regret (↓) on best score

Figure 9: Alignment performance of the same model trained with different losses after 20,000 iterations. The tabular augmentations preserves only a fraction ρ of the initial features, before adding Gaussian noise. Error bars show standard errors.



(a) Relative improvement (↑) over initialisation

(b) Regret (↓) on best score

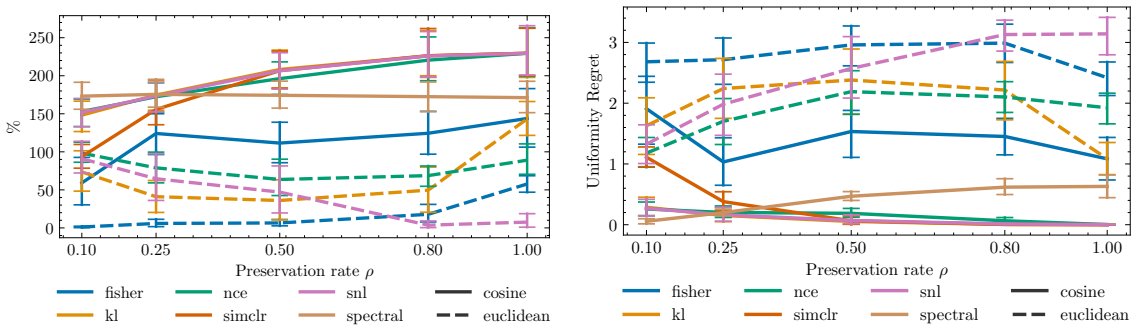
Figure 10: Uniformity performance of the same model trained with different losses after 20,000 iterations as the output decreases with respect to the number of initial features. The tabular augmentations preserves 80% of the initial features, before adding Gaussian noise. Error bars show standard errors.

other likelihood-based methods. Compiling both metrics, the spectral loss achieves the best trade-off between aligmnent and uniformity on the hypersphere. Overall, all losses, except the Fisher loss, follow the exact same variations throughout the experimental settings, and our suggestion of SNL, NCE and KL losses in this context are on par with other likelihood-based losses.

F.2 Performance for a learning rate of 1

As mentioned in the experiments, Section 4, we obtained different results for the Fisher loss when using a learning rate of 1 for the Adam optimiser. To effectively show the difference, we present in Figure 12 and Figure 13 the difference of score between a learning rate of 1 and a learning rate of 0.01. All parameters except learning rate are left unchanged, and the models once again start from the exact same initial weights.

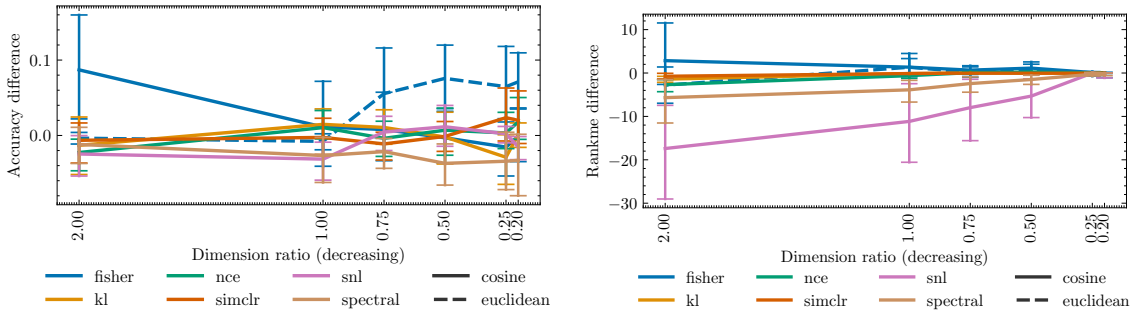
In both figures, a positive score indicates the the learning rate of 1 led to improved performance over the initial learning rate. Unsurprisingly, increasing the learning rate led to a performance decrease across all setups. In particular, that decrease is minimal for all dimension ratios with at most 10% in accuracy and 10 points in RankMe, except for the SNL loss depending on the energy. All Euclidean energy scenarios, except Fisher, more heavily suffered in accuracy and RankMe with a greater learning rate when varying the strength of augmentations in Figure 13.



(a) Relative improvement (↑) over initialisation

(b) Regret (↓) on best score

Figure 11: Uniformity performance of the same model trained with different losses after 20,000 iterations. The tabular augmentations preserves only a fraction ρ of the initial features, before adding Gaussian noise. Error bars show standard errors.

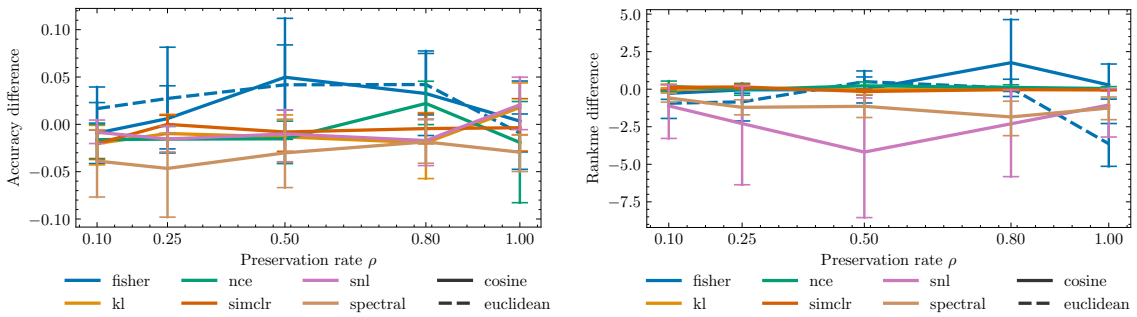


(a) Accuracy (↑) difference

(b) RankMe (↑) difference

Figure 12: Difference in final scores between models trained with a learning of 1 minus and models trained with a learning of 0.01 depending on the dimension ratio of the output. A positive value indicates better performance with a learning rate of 1.

In contrast, we observe that the Fisher loss, independently of the energy gets either increase or no improvement in most cases. The only notable decrease concerns RankMe when the preservation rate is equal to 1 in Figure 13. If we focus on Figure 12, we may note that the Euclidean energy is beneficial to the Fisher loss under a high learning rate for smaller output dimensions. This compensates our initial observations from Figure 4 where the exact same loss was instead dropping in performance. Our take-away is that a higher learning rate is beneficial for the Fisher loss, and that this loss should rather be used with an Euclidean energy.



(a) Accuracy (↑) difference

(b) RankMe (↑) difference

Figure 13: Difference in final scores between models trained with a learning of 1 minus and models trained with a learning of 0.01 depending on the preservation rate ρ of the input features. A positive value indicates better performance with a learning rate of 1.