Learning Symbolic Representations Through Joint GEnerative and DIscriminative Training

Emanuele Sansone
Department of Computer Science
KU Leuven
Leuven, 3001, Belgium
emanuele.sansone@kuleuven.be

Robin Manhaeve
Department of Computer Science
KU Leuven
Leuven, 3001, Belgium
robin.manhaeve@kuleuven.be

Abstract

We introduce **GEDI**, a Bayesian framework that combines existing self-supervised learning objectives with likelihood-based generative models. This framework leverages the benefits of both **GE**nerative and **DI**scriminative approaches, resulting in improved symbolic representations over standalone solutions. Additionally, GEDI can be easily integrated and trained jointly with existing neuro-symbolic frameworks without the need for additional supervision or costly pre-training steps. We demonstrate through experiments on real-world data, including SVHN, CIFAR10, and CIFAR100, that GEDI outperforms existing self-supervised learning strategies in terms of clustering performance by a significant margin. The symbolic component further allows it to leverage knowledge in the form of logical constraints to improve performance in the small data regime.

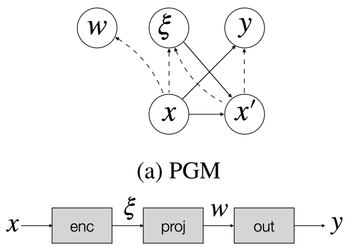
1 Introduction

Recently, neuro-symbolic learning has received attention as a new approach for integrating symbolic-based and sub-symbolic methods based on neural networks. This integration provides new capabilities in terms of perception and reasoning. Currently, neuro-symbolic solutions rely either on costly pre-training methods or on additional supervision at the symbolic representation level provided by the neural network, in order to effectively utilize subsequent learning feedback from the logical component (Manhaeve et al., 2018). This traditional top-down learning paradigm is subject to the problem of *representational collapse*. To gain a clearer understanding of the problem, imagine we have a tuple of three images, each of which contains a single digit (e.g., $< 3, 5, 8 >$). Along with this, we have information about the logical relationships between these digits (e.g., the third digit is the sum of the first two). Note that this task introduces less supervision compared to the digit addition experiment typically used in neuro-symbolic systems (i.e. the information about the sum is not provided). Current neuro-symbolic solutions can easily solve the task by mapping all input data onto the same symbol 0 and clearly solve the constrained task.

In this study, we present a bottom-up representation learning that can naturally integrate with, and leverage the information in logical constraints. We demonstrate that several existing self-supervised learning techniques and likelihood-based generative models can be unified within a coherent Bayesian framework called GEDI (Sansone & Manhaeve, 2022). The model leverages the complementary properties of discriminative approaches, which are suitable for representation learning, and of generative approaches, which capture information about the underlying density function generating the data, to learn better symbolic representations and support logical reasoning. Importantly, GEDI has two main advantages: it can be easily extended to the neuro-symbolic setting to address the collapse problem and it can also allow for learning symbolic representations in the small data regime, which is currently out of reach for existing self-supervised learning techniques.

2 GEDI Model

Model. Let us introduce the random quantities used in the model shown in Figure 1: (i) $x \in \Omega$, where Ω is a compact subset of $\mathbb{R}^d$, represents a data vector drawn independently from an unknown distribution $p(x)$ (for instance an image), (ii) $x' \in \Omega$ represents a transformed version of x using a stochastic data augmentation strategy $\mathcal{T}(x'|x)$ (obtained by adding for instance noisy or cropping the original image) (iii) $\xi \in \mathbb{R}^h$ is the latent representation of an input data point obtained from an encoder network (the latent representation of the original image) (iv) $w \in \mathcal{S}^{h-1}$, where $\mathcal{S}^{h-1}$ is a $h-1$ dimensional unit hypersphere, is the embedding vector of an input data point (obtained from the latent representation using a network called projection head), while (v) $y \in \{1, \dots, c\}$ is the symbolic representation of an input data point defined over c categories (namely the cluster label obtained by an output layer defined over the embedding representation). The corresponding probabilistic graphical model is given in Figure 1(a). Importantly, the generative process (solid arrows) is defined using the following conditional densities, namely: $p(w|x) = \mathcal{N}(w|0, I)$, viz. a multivariate Gaussian with zero mean and identity covariance, $p(\xi) = \mathcal{N}(\xi|0, I)$, $p(x'|x, \xi) = \mathcal{T}(x'|x)$ and $p(y|x) = \text{Softmax}(out(proj(enc(x))))$, where $enc : \Omega \to \mathbb{R}^h$ is an encoder used to compute the latent representation, $proj : \mathbb{R}^h \to \mathcal{S}^{h-1}$ is a projector head used to compute the embedding representation, and out computes the cosine similarity between the embedding representation w and the column vectors of a matrix of parameters $U \in \mathbb{R}^{h \times c}$ known as the cluster centers/prototypes (Caron et al., 2020). The inference process (dashed arrows) is given by the following conditional densities: $q(w|x) = \mathcal{N}(w|0, \Sigma)$, where $\Sigma = \sum_{i=1}^{n} (w_i - \bar{w})(w_i - \bar{w})^T + \beta I$ is an unnormalized sample covariance matrix computed over the embedding representations of n data points, β is a positive scalar used to ensure that Σ is positive-definite, $\bar{w} = 1/n \sum_{i=1}^{n} w_i$ is the mean of the embedding representations; $q(\xi|enc(x) - enc(x'), I)$ assesses the level of invariance between the latent representation of the input data and its augmented version; finally $q(y|x) = \text{SK}(out(proj(enc(x'))))$ defines a distribution over cluster/prototype assignments leveraging the Sinkhorn-Knopp algorithm (SK). Please refer to the work of Caron et al. (2020) for further details.



(a) PGM

(b) Block model

Figure 1: GEDI model. (a) shows the corresponding probabilistic graphical model (PGM). (b) shows the different modules of GEDI, namely the encoder, the projector head and an output module computing the cosine similarity between the embedding representation and the cluster centers.

Objective. Our training objective is based on an evidence lower bound on the negative entropy, derived from the probabilistic graphical model of Figure 1(a), namely:

$$E_{p(x)}\{\log p(x)\} \geq \underbrace{-CE(p, p_\Psi)}_{\text{Generative term}} + \underbrace{\mathcal{L}_{NF}(\Theta) + \mathcal{L}_{DI}(\Theta)}_{\text{Self-supervised learning terms}} \tag{1}$$

where $CE(p, p_\Psi)$ is the cross-entropy between the unknown distribution p and a generative model p_Ψ, equivalently seen as the negative data log-likelihood of the generative model p_Ψ. We define $p_\Psi(x) = e^{-u^T enc(x)}/\Gamma(\Psi)$ as an energy-based model, where Ψ includes both $u \in \mathbb{R}^h$ and the encoder parameters. Additionally,

$$\mathcal{L}_{NF}(\Theta) = -\underbrace{\mathbb{E}_{p(x)}\{KL(q(w|x)\|p(w))\}}_{\text{Decorrelation term}} - \underbrace{\mathbb{E}_{p(x)\mathcal{T}(x'|x)}\{KL(q(\xi|x, x')\|p(\xi))\}}_{\text{Invariance term}} \tag{2}$$

where the first and the second addends promote decorrelated features in the embedding representation and latent representations that are invariant to data augmentations, respectively. Finally,

$$\mathcal{L}_{DI}(\Theta) \geq \mathbb{E}_{p(x)\mathcal{T}(x'|x)}\{\mathbb{E}_{q(y|x')}\{\log p(y|x; \Theta)\} + H_q(y|x')\} \tag{3}$$

where $H_q(y|x')$ is the entropy computed over $q(y|x')$ and Θ includes all parameters of the encoder, projector head and the output layer of our model. Intuitively, the first addend in Eq. 3 forces the symbolic representations of the input data and its augmented version to be similar, whereas the second addend enforces uniformity on the cluster assignments, so as to avoid that all representations collapse to a single cluster. It is important to mention that the two objectives in Eqs. 2 and 3 are general enough to cover several proposed criteria in the literature of negative-free and cluster-based self-supervised learning (cf. Appendix A) (Sansone & Manhaeve, 2022). Interestingly, the objective in Eq. 1 provides a natural unification between generative and discriminative models based on self-supervised learning. Learning the GEDI model proceeds using standard gradient descent by maximizing Eq. 1 (more details about the training procedure are provided in Appendix C).

Table 1: Clustering performance in terms of normalized mutual information on test set (SVHN, CIFAR-10, CIFAR-100). Higher values indicate better clustering performance. We observe unstable training for SwAV on CIFAR-100. We report the best performance achieved out of 10 experiments.

Dataset	**JEM**	**Barlow**	**SwAV**	**GEDI**	**Gain**
SVHN	0.04	0.20	0.24	**0.39**	**+0.15**
CIFAR-10	0.04	0.22	0.39	**0.41**	**+0.02**
CIFAR-100	0.05	0.46	0.69*	**0.72**	**+0.03**

3 Experiments

We perform experiments to evaluate the discriminative performance of GEDI and its competitors, namely an energy-based model JEM (Grathwohl et al., 2020), which is trained with persistent contrastive divergence (similarly to our approach) and 2 self-supervised baselines, viz. a negative-free approach based on Barlow Twins (Zbontar et al., 2021) and a discriminative one based on SwAV (Caron et al., 2020). The whole analysis is divided into two main experimental settings, the first one based on real-world data, including SVHN, CIFAR-10 and CIFAR-100, and the second one based on a neural-symbolic learning task in the small data regime constructed from MNIST. We use existing code both as a basis to build our solution and also to run the experiments for the different baselines. In particular, we use the code from Duvenaud et al. (2021) for training energy-based models and the repository from da Costa et al. (2022) for all self-supervised baselines. Implementation details as well as additional experiments are reported in the Appendices.

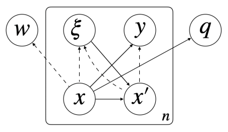
3.1 SVHN, CIFAR-10, CIFAR-100

We consider three well-known computer vision benchmarks, namely SVHN, CIFAR-10 and CIFAR-100. We use a simple 8-layer Resnet network for the backbone encoder for both SVHN and CIFAR-10 (around 1M parameters) and increase the hidden layer size for CIFAR-100 (around 4.1M parameters) following Duvenaud et al. (2021). We use a MLP with a single hidden layer for $proj$ (the number of hidden neurons is twice the size of the input vector), we choose $h = 256$ for CIFAR-100 and $h = 128$ for all other cases. Additionally, we use data augmentation strategies commonly used in the self-supervised learning literature, including color jitter, and gray scale conversion to name a few. We train JEM, Barlow, SwAV and GEDI for 100 epochs using Adam optimizer with learning rate $1e-4$ and batch size 64. Further details about the hyperparameters are available in Appendix F. We evaluate the clustering performance against the ground truth labels by using the Normalized Mutual Information (NMI) score.

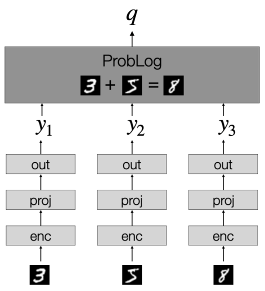
We report all quantitative performance in Table 1. Specifically, we observe that JEM fails to solve the clustering task for all datasets. This is quite natural, as JEM is a purely generative approach, mainly designed to perform implicit density estimation. Barlow Twins achieves inferior performance to SwAV, due to the fact that is not a cluster-based self-supervised learning approach. On the contrary, we observe that GEDI is able to outperform all other competitors, thanks to the exploitation of the complementary properties of both generative and self-supervised models. Indeed, the discriminative component in GEDI leverages the information about the underlying data manifold structure learnt by the generative part, thus improving the learning of the symbolic representation. In Appendix G we provide an ablation study to assess the importance of the different loss terms involved in Eq. 1. Additionally, we conduct experiments on linear probe evaluation, generation and OOD detection tasks commonly used in the literature of self-supervised learning and energy-based models. Results are reported in Appendix H.

3.2 Neural-symbolic setting

For the final task, we consider applying the proposed method to a neural-symbolic setting. For this, we borrow an experiment from DeepProbLog Manhaeve et al. (2018). In this task, each example consists of a three MNIST images such that the value of the last one is the sum of the first two, e.g. $+$ $=$. This can thus be considered a minimal neural-symbolic tasks, as it requires a minimal reasoning task (a single addition) on top of the image classification task. This task only contains positive examples and requires a minimal modification to the probabilistic graphical model, as shown in Figure 2. We use the inference mechanism from DeepProbLog to calculate the probability that this sum holds, and optimize this probability using the cross-entropy loss function, which is optimized along with the other loss functions. For this setting, this coincides with the Semantic Loss function (Xu et al., 2018). To be able to calculate the probability of this addition constraint, we need the classification probabilities for each digit.



(a) PGM



(b) Block model

Figure 2: Neuro-symbolic task requiring to learn the correct symbolic representation of the digits given only a tuple of images and the corresponding logical constraint. In this setting, $n = 3$ data points and q is a Boolean random variable used to detect if the logical constraint is satisfied.

It is a specifically interesting use case for neural-symbolic learning, since when only the probability is optimized, the neural network tends to collapse onto the trivial solution of classifying each digit as a 0 (i.e. $y_1 = y_2 = y_3 = 0$ in Figure 2). This is a logically correct but undesirable solution. Optimizing the discriminative objective should prevent this collapse. We hypothesize that a neural network can be trained to correctly classify MNIST digits by using both GEDI and the logical constraint. Since the MNIST is an easy dataset, we focus on the small data regime, and see whether the logical constraint is able to provide additional information. The hyperparameters are identical to those used in Section 3.1. Further details about the hyperparameters are dependent on the data regime, and are available in Appendix I.

We evaluate the model by measuring the accuracy and NMI of the ResNet model on the MNIST test dataset for different numbers of training examples. The results are shown in Table 2. Here, N indicates the number of addition examples, which each have 3 MNIST digits. As expected, the DeepProbLog baseline from Manhaeve et al. (2018) completely fails to classify MNIST images. It has learned to map all images to the class 0, as this results in a very low loss when considering only the logic, resultsing in an accuracy of 0.10 and an NMI of 0.0. The results also show that, without the NeSy constraint, the accuracy is low for all settings. The NMI is higher, however, and increases as there is more data available. This is expected, since the model is still able to learn how to cluster from the data. However, it is unable to correctly classify, as there is no signal in the data that is able to assign the correct label to each cluster. By including the constraint loss, the accuracy improves, as the model now has information on which cluster belongs to which class. Furthermore, it also has a positive effect on the NMI, as we have additional information on the clustering which is used by the model. These results show us that the proposed method is beneficial to learn to correctly recognize MNIST images using only a weakly-supervised constraint, whereas other NeSy methods fail without additional regularization. Furthermore, we show that the proposed method can leverage the information offered by the constraint to further improve the NMI and classification accuracy.

Table 2: The accuracy and NMI of GEDI on the MNIST test set after training on the addition dataset, both with and without the NeSy constraint. Additionally, we use DeepProbLog (Manhaeve et al., 2018) as a baseline without using our GEDI model. We trained each model 5 times and report the mean and standard deviation.

	Without GEDI		**Without constraint**		**With constraint**	
N	**Acc.**	**NMI**	**Acc.**	**NMI**	**Acc.**	**NMI**
100	0.10 ± 0.00	0.00 ± 0.00	0.08 ± 0.03	0.28 ± 0.03	$\mathbf{0.25 \pm 0.03}$	$\mathbf{0.41 \pm 0.03}$
1000	0.10 ± 0.00	0.00 ± 0.00	0.09 ± 0.02	0.47 ± 0.10	$\mathbf{0.52 \pm 0.26}$	$\mathbf{0.86 \pm 0.06}$
10000	0.10 ± 0.00	0.00 ± 0.00	0.17 ± 0.12	0.68 ± 0.09	$\mathbf{0.98 \pm 0.00}$	$\mathbf{0.97 \pm 0.01}$

Acknowledgments

This research is funded by TAILOR, a project from the EU Horizon 2020 research and innovation programme under GA No 952215. This research received also funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

References

A. Bardes, J. Ponce, and Y. Lecun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. In *ICLR*, 2022a.

A. Bardes, J. Ponce, and Y. LeCun. VICREGL: Self-Supervised Learning of Local Visual Features. In *NeurIPS*, 2022b.

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep Clustering for Unsupervised Learning of Visual Features. In *ECCV*, 2018.

M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *NeurIPS*, 2020.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, 2020.

M. Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *NeurIPS*, 2013.

V. G. T. da Costa, E. Fini, M. Nabi, N Sebe, and E. Ricci. Solo-learn: A Library of Self-supervised Methods for Visual Representation Learning. *JMLR*, 23(56):1–6, 2022.

A. Van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. In *arXiv*, 2018.

Y. Du and I. Mordatch. Implicit Generation and Generalization in Energy-Based Models. *arXiv*, 2019.

D. Duvenaud, J. Kelly, K. Swersky, M. Hashemi, M. Norouzi, and W. Grathwohl. No MCMC for Me: Amortized Samplers for Fast and Stable Training of Energy-Based Models. In *ICLR*, 2021.

A. Ermolov, A. Siarohin, E. Sangineto, and N. Sebe. Whitening for Self-Supervised Representation Learning. In *ICML*, 2021.

W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. Your Classifier is Secretly an Energy Based Model and You Should Treat It Like One. In *ICLR*, 2020.

B. Kim and J. C. Ye. Energy-Based Contrastive Learning of Visual Representations. In *NeurIPS*, 2022.

K. Lee. Prototypical Contrastive Predictive Coding. In *ICLR*, 2022.

X. Liu, Z. Wang, Y. Li, and S. Wang. Self-Supervised Learning via Maximum Entropy Coding. In *NeurIPS*, 2022.

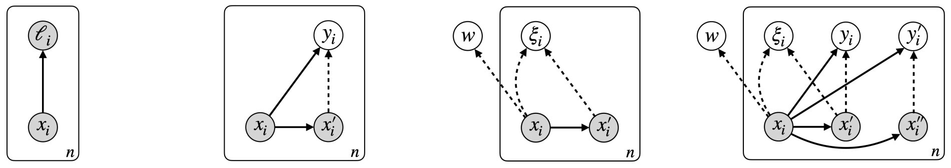
R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt. DeepProbLog: Neural Probabilistic Logic Programming. In *NeurIPS*, 2018.

E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? In *ICLR*, 2019.

E. Nijkamp, M. Hill, S. C. Zhu, and Y. N. Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *NeurIPS*, 2019.

E. Nijkamp, M. Hill, T. Han, S.-C. Zhu, and Y. N. Wu. On the Anatomy of MCMC-Based Maximum Likelihood Learning of Energy-Based Models. In *AAAI*, 2020.

Olivier O. Henaff. Data-Efficient Image Recognition with Contrastive Predictive Coding. In *ICML*, 2020.



(a) Contrastive (CT) (b) Discriminative (DI) (c) Negative-Free (NF) (d) GEDI

Figure 3: Probabilistic graphical models for the different classes of self-supervised learning approaches. White and grey nodes represent hidden and observed vectors/variables, respectively. Solid arrows define the generative process, whereas dashed arrows identify auxiliary posterior densities/distributions.

O.Chapelle, B. Schölkopf, and A. Zien. Semi-Supervised Learning. Adaptive Computation and Machine Learning. *Methods*, 2010.

S. Ozsoy, S. Hamdan, S. Arik, D. Yuret, and A. T. Erdogan. Self-Supervised Learning with an Information Maximization Criterion. In *NeurIPS*, 2022.

E. Sansone and R. Manhaeve. GEDI: GEnerative and DIscriminative Training for Self-Supervised Learning. In *arXiv*, 2022.

J. Xie, Y. Lu, S. C. Zhu, and Y. N. Wu. A Theory of Generative Convnet. In *ICML*, 2016.

H. Xu, H. Xiong, and G. J. Qi. K-Shot Contrastive Learning of Visual Features with Multiple Instance Augmentations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *ICML*, 2018.

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *ICML*, 2021.

A Bayesian Interpretation of Self-Supervised Learning Objectives

We distinguish self-supervised learning approaches according to three different classes: 1) contrastive, 2) cluster-based (or discriminative) and, 3) negative-free (or non-contrastive) methods. Fig. 3 shows the corresponding probabilistic graphical models for each of the self-supervised learning classes. From Fig. 3, we can also see how the different random quantities, we have introduced in Section 2, are probabilistically related to each other. Importantly, here we consider to have a set of input samples drawn independently from p and use it i to index different samples, viz. $i = \{1, \dots, n\}$

We are now ready to give an interpretation of the different SSL classes.

A.1 Contrastive SSL

Contrastive self-supervised learning can be interpreted in probabilistic terms using the graphical model in Fig. 3(a). In particular, we can define the following conditional density:

$$p(\ell_i|x_i;\Theta) = \frac{e^{sim(g(x_{\ell_i}),g(x_i))/\tau}}{\sum_{j=1}^{n} e^{sim(g(x_j),g(x_i))/\tau}} \tag{4}$$

where sim is a similarity function, $\tau > 0$ is temperature parameter used to calibrate the uncertainty for $p(\ell_i|x_i;\Theta)$ and $\Theta = \{\theta, \{x_i\}_i^n\}$ is the set of parameters, including the parameters of the embedding function and the observed data. The learning criterion can be obtained from the expected log-likelihood computed on the observed random quantities (and using the factorization provided by the graphical model), namely:

$$\mathbb{E}_{p(x_{1:n},\ell_{1:n})}\big\{ \log p(x_{1:n}, \ell_{1:n}; \Theta)\big\} \tag{5}$$

$$
\begin{aligned}
&= \mathbb{E}_{\prod_{j=1}^{n} p(x_j)\delta(\ell_j - j)}\Big\{ \log \prod_{i=1}^{n} p(x_i) p(\ell_i|x_i;\Theta)\Big\} \\
&= \sum_{i=1}^{n} \mathbb{E}_{p(x_i)}\big\{ \log p(x_i)\big\} + \mathbb{E}_{\prod_{j=1}^{n} p(x_j)\delta(\ell_j - j)}\left\{ \sum_{i=1}^{n} \log p(\ell_i|x_i;\Theta)\right\} \\
&= \underbrace{\sum_{i=1}^{n} \mathbb{E}_{p(x_i)}\big\{ \log p(x_i)\big\}}_{\text{Negative entropy term, } -H_p(x_{1:n})} \underbrace{+\mathbb{E}_{\prod_{j=1}^{n} p(x_j)}\left\{ \sum_{i=1}^{n} \log p(\ell_i = i|x_i;\Theta)\right\}}_{\text{Conditional log-likelihood term } \mathcal{L}_{CT}(\Theta)} \qquad (6)
\end{aligned}
$$

where δ in the second equality is a delta function and we use for instance $x_{1:n}$ as a compact way to express $x_1, \dots, x_n$. From Eq. (6), we observe that the expected log-likelihood can be rewritten as the sum of two quantities, namely a negative entropy and a conditional log-likelihood terms. However, only the second addend in Eq. (6) is relevant for maximization purposes over the parameters θ. Importantly, we can now show that the conditional log-likelikood term in Eq. (6), coupled with the definition provided in Eq. (4), corresponds to the notorious InfoNCE objective (den Oord et al., 2018). To see this, let us recall InfoNCE:

$$
\text{InfoNCE} \propto \mathbb{E}_{\prod_{j=1}^{n} p(x_j, z_j)}\left\{ \sum_{i=1}^{n} \log \frac{e^{f(x_i, x_i)}}{\sum_{k=1}^{n} e^{f(x_k, x_i)}}\right\}
$$

By choosing $p(x, z) = p(x)\delta(z - g(x))$ and $f(x, z) = sim(g(x), z)/\tau$ we recover the conditional log-likelihood term in Eq. (6). Importantly, other contrastive objectives, such as CPC (O. Henaff, 2020), SimCLR (Chen et al., 2020), ProtoCPC (Lee, 2022), KSCL (Xu et al., 2022) to name a few, can be obtained once we have a connection to InfoNCE.

A.2 DISCRIMINATIVE/CLUSTER-BASED SSL

Cluster-based SSL can be interpreted in probabilistic terms using the graphical model in Fig. 3(b). In particular, we can define the following conditional density:

$$
p(y_i|x_i;\Theta) = \frac{e^{U_{:y_i}^T G_{:i}/\tau}}{\sum_y e^{U_{:y}^T G_{:i}/\tau}} \qquad (7)
$$

where $U \in \mathbb{R}^{h\times c}$ is a matrix[1] of c cluster centers, $G = [g(x_1), \dots, g(x_n)]$ is a matrix of embeddings of size $h \times n$ and $\Theta = \{\theta, U\}$ is the set of parameters, including the ones for the embedding function and the cluster centers. The learning criterion can be obtained in a similar way to what we have done previously for contrastive methods. In particular, we have that

$$
\begin{aligned}
&\mathbb{E}_{p(x_{1:n})}\{\log p(x_{1:n};\Theta)\} \qquad (8) \\
&= -H_p(x_{1:n}) + \mathbb{E}_{p(x_{1:n})\mathcal{T}(x'_{1:n}|x_{1:n})}\left\{ \log \sum_{y_{1:n}} p(y_{1:n}|x_{1:n};\Theta)\right\} \\
&= -H_p(x_{1:n}) + \mathbb{E}_{p(x_{1:n})\mathcal{T}(x'_{1:n}|x_{1:n})}\Big\{ \log \sum_{y_{1:n}} \frac{q(y_{1:n}|x'_{1:n})}{q(y_{1:n}|x'_{1:n})} p(y_{1:n}|x_{1:n};\Theta)\Big\} \\
&\geq -H_p(x_{1:n}) - \mathbb{E}_{p(x_{1:n})\mathcal{T}(x'_{1:n}|x_{1:n})}\{KL(q(y_{1:n}|x'_{1:n})\|p(y_{1:n}|x_{1:n}))\} \\
&= \underbrace{\sum_{i=1}^{n} \mathbb{E}_{p(x_i)}\big\{ \log p(x_i)\big\}}_{\text{Negative entropy term, } -H_p(x_{1:n})} \\
&\quad + \underbrace{\sum_{i=1}^{n} \mathbb{E}_{p(x_i)\mathcal{T}(x'_i|x_i)}\{\mathbb{E}_{q(y_i|x'_i)} \log p(y_i|x_i;\Theta) + H_q(y_i|x'_i)\}}_{\text{Discriminative term } \mathcal{L}_{DI}(\Theta)} \qquad (9)
\end{aligned}
$$

[1]We use subscripts to select rows and columns. For instance, $U_{:y}$ identify $y-$th column of matrix U.

where $q(y_i|x_i')$ is an auxiliary distribution. Notably, maximizing the discriminative term is equivalent to minimize the KL divergence between the two predictive distributions $p(y_i|x_i)$ and $q(y_i|x_i')$, thus learning to predict similar category for both sample x_i and its augmented version x_i', obtained through $\mathcal{T}$. Importantly, we can relate the criterion in Eq. (9) to the objective used in optimal transport Cuturi (2013), by substituting Eq. (7) into Eq. (9) and adopting a matrix format, namely:

$$\mathcal{L}_{DI}(\Theta;\mathcal{T}) = \frac{1}{\tau}\left\{\mathbb{E}_{p(x_{1:n})\mathcal{T}(x_i'|x_i)}\{Tr(QU^TG)\} + \tau\mathbb{E}_{p(x_{1:n})\mathcal{T}(x_i'|x_i)}\{H_Q(y_{1:n}|x_{1:n}')\}\right\} \tag{10}$$

where $Q = [q(y_1|x_1'),\dots,q(y_n|x_n')]^T$ is a prediction matrix of size $n\times c$ and $Tr(A)$ is the trace of an arbitrary matrix A. Note that a naive maximization of $\mathcal{L}_{DI}(\Theta)$ can lead to obtain trivial solutions like the one corresponding to uniformative predictions, namely $q(y_i|x_i') = p_\gamma(y_i|x_i) = \text{Uniform}(\{1,\dots,c\})$ for all $i = 1,\dots,n$. Fortunately, the problem can be avoided and solved exactly using the Sinkhorn-Knopp algorithm, which alternates between maximizing $\mathcal{L}_{DI}(\Theta)$ in Eq. (10) with respect to Q and with respect to Θ, respectively. This is indeed the procedure used in several cluster-based SSL approaches, like DeepCluster Caron et al. (2018) and SwAV Caron et al. (2020), to name a few.

A.3 Non-contrastive/Negative-Free SSL

We can provide a probabilistic interpretation also for negative-free SSL using the graphical model shown in Fig. 3(c). Note that, for the sake of simplicity in the graph and in the following derivation, the latent variables (w and all ξ_i) are considered independent of each other only for the generation process. In reality, one should consider an alternative but equivalent model, using a generating process including also the edges $x_i\to w$, $\xi_i\to x_i'$ and defining $p(w|x_i)=p(w)$ and $p(x_i'|x_i,\xi_i) = \mathcal{T}(x_i'|x_i)$ for all $i = 1\dots,n$. Based on these considerations, we can define the prior and our auxiliary and inference densities for the model in the following way:

$$\begin{aligned} p(w) &= \mathcal{N}(w|0,I)\\ p(\xi_i) &= \mathcal{N}(\xi_i|0,I)\\ q(w|x_{1:n};\Theta) &= \mathcal{N}(w|0,\Sigma)\\ q(\xi_i|x_i,x_i';\Theta) &= \mathcal{N}(\xi_i|enc(x_i)-enc(x_i'),I) \end{aligned} \tag{11}$$

where $\mathcal{N}(\cdot|\mu,\Sigma)$ refers to a multivariate Gaussian density with mean μ and covariance Σ, I is an identity matrix, $\Sigma = \sum_{i=1}^n (g(x_i)-\bar{g})(g(x_i)-\bar{g})^T + \beta I$, β is positive scalar used to ensure the positive-definiteness of Σ, $\bar{g} = 1/n\sum_{i=1}^n g(x_i)$ and $\Theta = \{\theta\}$. Importantly, while $q(w|x_{1:n})$ in Eq. (11) is used to store the global statistical information of the data in the form of an unnormalized sample covariance, $q(\xi_i|x_i,x_i')$ is used to quantify the difference between a sample and its augmented version in terms of their latent representation. Similarly to previous SSL classes and by reusing definitions in Eq. (11), we can devise the learning criterion in the following way:

$$\begin{aligned} \mathbb{E}_{p(x_{1:n})}\{\log p(x_{1:n};\Theta)\} &\geq -H_p(x_{1:n}) - \mathbb{E}_{p(x_{1:n})}\{KL(q(w|x_{1:n};\Theta)\|p(w))\}\\ &\quad - \mathbb{E}_{p(x_{1:n})\mathcal{T}(x_{1:n}'|x_{1:n})}\{KL(q(\xi_{1:n}|x_{1:n},x_{1:n}';\Theta)\|p(\xi_{1:n}))\}\\ &= -H_p(x_{1:n}) - \mathbb{E}_{p(x_{1:n})}\{KL(q(w|x_{1:n};\Theta)\|p(w))\}\\ &\quad -\sum_{i=1}^n \mathbb{E}_{p(x_i)\mathcal{T}(x_i'|x_i)}\{KL(q(\xi_i|x_i,x_i';\Theta)\|p(\xi_i))\}\\ &\propto \underbrace{\sum_{i=1}^n \mathbb{E}_{p(x_i)}\big\{\log p(x_i)\big\}}_{\text{Negative entropy term, } -H_p(x_{1:n})} \underbrace{-\mathbb{E}_{p(x_{1:n})}\left\{\frac{Tr(\Sigma)}{2} - \frac{\log|\Sigma|}{2}\right\}}_{\text{Negative-free term, } \mathcal{L}_{NF}(\Theta) \text{ (first part)}}\\ &\quad \underbrace{-\sum_{i=1}^n \mathbb{E}_{p(x_i)\mathcal{T}(x_i'|x_i)}\left\{\frac{dist(x_i,x_i')}{2}\right\}}_{\text{Negative-free term, } \mathcal{L}_{NF}(\Theta) \text{ (second part)}} \end{aligned} \tag{12}$$

where $dist(x,x') = \|enc(x)-enc(x')\|^2$ and $|A|$ computes the determinant of an arbitrary matrix A. Notably, the maximization of $\mathcal{L}_{NF}(\Theta)$ promotes both decorrelated embedding features, as the first two addends in $\mathcal{L}_{NF}(\Theta)$ (obtained from the first KL term in Eq. (12)) force Σ to become an identity matrix, as well as representations that are invariant to data augmentations, thanks to the third addend in $\mathcal{L}_{NF}(\Theta)$. It is important to mention that $\mathcal{L}_{NF}(\Theta)$ in Eq. (12) recovers two recent negative-free criteria, namely CorInfoMax (Ozsoy et al., 2022) and MEC (Liu et al., 2022). We can also relate $\mathcal{L}_{NF}(\Theta)$ to other existing negative-free approaches, including Barlow Twins (Zbontar et al., 2021), VicReg (Bardes et al., 2022a;b) and W-MSE (Ermolov et al., 2021).

B Unifying Generative and SSL Models: A General Recipe (GEDI)

In all three classes of SSL approaches (see Eqs. (6),(9) and (12)), the expected data log-likelihood can be lower bounded by the sum of two contribution terms, namely a negative entropy $-H_p(x_{1:n})$ and a conditional log-likelihood term, chosen from $\mathcal{L}_{CT}(\Theta), \mathcal{L}_{DI}(\Theta)$ and $\mathcal{L}_{NF}(\Theta)$. A connection to generative models emerges by additionally lower bounding the negative entropy term, namely:

$$\begin{aligned} -H_p(x_{1:n}) &= \mathbb{E}_{p(x_{1:n})}\{\log p(x_{1:n})\} \\ &= \sum_{i=1}^{n} \mathbb{E}_{p(x_i)}\{\log p(x_i)\} \\ &= \sum_{i=1}^{n} \left[\mathbb{E}_{p(x_i)}\{\log p_\Psi(x_i)\} + KL(p(x_i)\|p_\Psi(x_i))\right] \\ &\geq \underbrace{\sum_{i=1}^{n} \mathbb{E}_{p(x_i)}\{\log p_\Psi(x_i)\}}_{-CE(p,p_\Psi)} \end{aligned} \tag{13}$$

where $p_\Psi(x)$ is a generative model parameterized by Ψ. Notably, the relation in (13) can be substituted in any of the objectives previously derived for the different SSL classes, thus allowing to jointly learn both generative and SSL models. This leads to a new GEnerative and DIscriminative family of models, which we call GEDI. Importantly, any kind of likelihood-based generative model (for instance variational autoencoders, normalizing flows, autoregressive or energy-based models) can be considered in GEDI. In this work, we argue that much can be gained by leveraging the GEDI integration. Notably, there has been a recent work EBCLR (Kim & Ye, 2022) integrating energy-based models with contrastive SSL approaches. Here, we show that EBCLR represents one possible instantiation of GEDI. For instance, let us consider contrastive SSL and observe that the conditional density in Eq. (4) can be decomposed into a joint and a marginal densities (similarly to what is done in (Grathwohl et al., 2020)):

$$\begin{aligned} p(\ell, x; \Theta) &= \frac{e^{sim(g(x_\ell),g(x))/\tau}}{\Gamma(\Theta)} \\ p(x; \Theta) &= \frac{\sum_{j=1}^{n} e^{sim(g(x_{\ell_j}),g(x))/\tau}}{\Gamma(\Theta)} \\ &= \frac{e^{-\underbrace{\left(-\log \sum_{j=1}^{n} e^{sim(g(x_{\ell_j}),g(x))/\tau}\right)}_{E(x;\Theta)}}}{\Gamma(\Theta)} \end{aligned} \tag{14}$$

where $E(x, \Theta)$ defines the energy score of the marginal density. Now, by choosing $p_\Psi(x) = p(x; \Theta)$ and $sim(z, z') = -\|z - z'\|^2$ in Eq. (14), one recovers the exact formulation of EBCLR (Kim & Ye, 2022).

C Whole Model and Training Algorithm

Figure 4 shows the whole GEDI architecture and how to compute the different losses. Importantly, GEDI training relies on a newly introduced data augmentation strategy, called DAM.

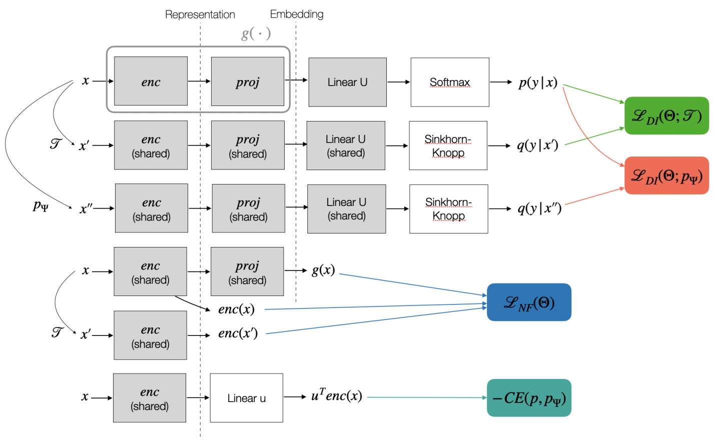


Figure 4: Diagram of the whole model. Grey boxes are shared among different rows.

Algorithm 1: Data Augmentation based on Manifold structure (DAM). Formulation based on a batch of samples of size n.

```
**Input:** $x_{1:n}, p_\Psi, \epsilon, T$;
**Output:** $x'_{1:n}$;
Sample $\Delta^0$ uniformly at random from $B_\epsilon$;
**For** $t = 0, \dots, T$;
    Evaluate $\nabla_x p_\Psi(x_i^t + \Delta^t)$ for all $i = 1, \dots, n$;
    Update $x_i^t$ using Eq. (15) for all $i = 1, \dots, n$;
    $\Delta^t \leftarrow \Delta^0$;
$x'_i \leftarrow x_i^T$ for all $i = 1, \dots, n$;
**Return** $x'_{1:n}$;
```

Data Augmentation based on Manifold structure (DAM). The routine uses generated samples obtained by "walking" on the approximated data manifold induced by the energy-based model and it assigns the same label of the original points to these samples through the discriminative loss $\mathcal{L}_{DI}(\Theta)$, thus enforcing the manifold assumption, commonly used in semi-supervised learning (O.Chapelle et al., 2010). DAM consists of an iterative procedure starting from an original training data point x, drawn from p, and generating new samples x' along the approximated data manifold induced by p_Ψ. At each iteration t, the algorithm performs two main operations: Firstly, it locally perturbs a sample x^t by randomly choosing a vector Δ^t on a ball of arbitrarily small radius $\epsilon > 0$, viz. B_ϵ, and secondly, it projects the perturbed sample $x^t + \Delta^t$ back onto the tangent plane of the approximated data manifold $\mathcal{M}_{p_\Psi}$ using the following update rule:

$$x^t \leftarrow x^t + \underbrace{\Delta^t - \left(\frac{\nabla_x p_\Psi(x^t+\Delta^t)^T\Delta^t}{\|\nabla_x p_\Psi(x^t+\Delta^t)\|^2}\right)\nabla_x p_\Psi(x^t+\Delta^t)}_{\Delta^t_\parallel} \tag{15}$$

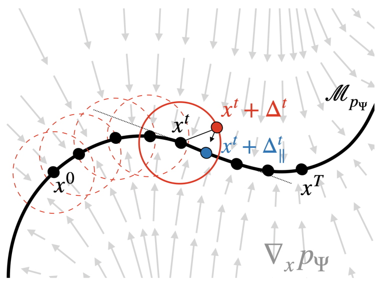


Figure 5: Data augmentation strategy exploiting the information about the manifold structure $\mathcal{M}_{p_\Psi}$ and the vector field $\nabla_x p_\Psi$ induced by the energy-based model p_Ψ. A local perturbation Δ^t of a point x^t is projected back onto the tangent plane

A visual interpretation as well as a complete description of the strategy are provided in Figure 5 and Algorithm 1, respectively.

Learning a GEDI model. The data augmentation strategy proves effective when the energy-based model approximates well the unknown data density $p(x)$. Consequently, we opt to train our GEDI model using a two-step procedure, where we first train the energy-based model to perform implicit density estimation and subsequently train the whole model by maximizing

Algorithm 2: GEDI Training. Formulation based on a batch of samples of size n.

```
**Input:** $x_{1:n}$, Iters$_1$, Iters$_2$, $T$, $\epsilon$, SGLD and Adam optimizer hyperparameters;
**Output:** Trained model $g$;
# Step 1;
**For** iter $= 1, \ldots,$ Iters$_1$;
    Generate samples from $p_\Psi$ using SGLD;
    $\Psi \leftarrow$ Adam maximizing $-CE(p, p_\Psi)$;
# Step 2;
**For** iter $= 1, \ldots,$ Iters$_2$;
    $x'_{1:n} \leftarrow$ DAM$(x_{1:n}, p_\Psi, \epsilon, T)$;
    Generate samples from $p_\Psi$ using SGLD;
    $\Psi, \Theta \leftarrow$ Adam maximizing Eq. (1);
**Return** $g$;
```

Table 3: Clustering performance in terms of normalized mutual information (NMI) on test set (moons and circles). Higher values indicate better clustering performance. Mean and standard deviations are computed from 5 different runs. GEDI uses $T = 10$ moves in DAM.

Dataset	**JEM**	**SwAV**	**GEDI**	**Gain**
Moons	0.0±0.0	0.8±0.1	**0.98±0.02**	**+0.18**
Circles	0.0±0.0	0.0±0.0	**1.00±0.01**	**+1.00**

the objective in Eq. (1). Regarding the first stage, we maximize only the generative term in Eq. (1) whose gradient is given by the following relation:

$$\begin{aligned}-\nabla_\Psi CE(p, p_\Psi) &= \mathbb{E}_{p(x)}\left\{\nabla_\Psi \log e^{u^T enc(x)}\right\} \\ &\quad - \nabla_\Psi \log \Gamma(\Psi) \\ &= \mathbb{E}_{p(x)}\{\nabla_\Psi \log e^{u^T enc(x)}\} \\ &\quad \mathbb{E}_{p_\Psi(x)}\left\{\nabla_\Psi \log e^{u^T enc(x)}\right\}\end{aligned} \tag{16}$$

where the first and the second expectations in Eq. (16) are estimated using the training and the generated data, respectively. To generate data from p_Ψ, we use a sampler based on Stochastic Gradient Langevin Dynamics (SGLD), thus following recent best practices to train energy-based models (Xie et al., 2016; Nijkamp et al., 2019; Du & Mordatch, 2019; Nijkamp et al., 2020).

Regarding the second stage, we maximize the whole objective in Eq. (1). Specifically, at each training iteration, we run the DAM routine to obtain the augmented sample x'. We also use the augmented samples from the stochastic data augmentation strategy $\mathcal{T}$. Both augmentations are used to compute $\mathcal{L}_{DI}(\Theta)$ through the differentiable clustering prodecure of SwAV (Caron et al., 2020). The learning process is summarized in Algorithm 2.

Computational requirements. Compared to traditional SSL training, and more specifically to SwAV, our learning algorithm requires additional operations and therefore increased computational requirements (but constant given T and Iters$_1$). Indeed, (i) we need an additional training step (Step 1) to pre-train a generative model to approximate the unknown data density and to ensure the proper working of DAM in the second step, (ii) we need to generate samples from p_Ψ to continue learning the generative model in Step 2 and (ii) we also need to run T additional forward and backward passes through the energy-based model to run the DAM strategy at each iteration of the GEDI training.

D MOTIVATING TOY EXAMPLES FOR DAM

We consider two well-known synthetic datasets, namely moons and circles. We use a multi-layer perceptron (MLP) with two hidden layers (100 neurons each) for enc and one with a single hidden layer (4 neurons) for $proj$, we choose $h = 2$ and we choose $\mathcal{T}(x'|x) = N(0, \sigma^2 I)$ with $\sigma = 0.03$ as our data augmentation strategy. We train JEM, SwAV and GEDI for $7k$ iterations using Adam optimizer with learning rate $1e-3$. Further details about the hyperparameters are available in the Appendix E. We evaluate the clustering performance both qualitatively, by visualizing the cluster assignments using different colors, as well as quantitatively, by using the Normalized Mutual Information (NMI) score. Furthermore, we conduct an ablation study for the different components of GEDI.

We report all quantitative performance in Table 3. Specifically, we observe that JEM fails to solve the clustering task for both datasets. This is quite natural, as JEM is a purely generative approach, mainly designed to perform implicit density estimation. SwAV can only solve the clustering task for the moons dataset, highlighting the fact that it is not able to exploit the information from the underlying density used to generate the data. However, GEDI can recover the true clusters in both datasets. This is due to the fact that GEDI uses the information from the generative component through DAM to inform the cluster-based one. Consequently, GEDI is able to exploit the manifold structure underlying data. Figure 6 provide some examples of predictions by SwAV and GEDI on the two datasets. From the figure, we can clearly see that GEDI can recover the two data manifolds up to a permutation of the labels.

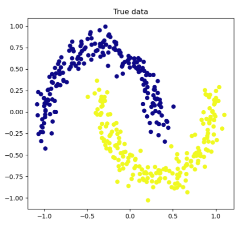
We conduct an ablation study to understand the impact of the different components of GEDI. We compare four different versions of GEDI, namely the full version (called simply *GEDI*), GEDI trained without $\mathcal{L}_{NF}(\Theta)$ (called *no NF*), GEDI trained without the first stage and also without $\mathcal{L}_{NF}(\Theta)$ (called *no NF, no train. 1*) and GEDI trained without $\mathcal{L}_{NF}(\Theta)$ using two different encoders for computing the discriminative and the generative terms in our objective (called *no NF, 2 enc.*). From the results in Figure 8, we can make the following observations: (i) DAM plays a crucial role to recover the manifold structure, as the performance increase with the number of steps T for *GEDI*, *no NF* and *no NF, 2 enc.*. However, the strategy is not effective when we don't use the first training stage, as demonstrated by the results obtained by *no NF, no train. 1* on circles. This confirms our original hypothesis that DAM requires a good generative model; (ii) When looking at the results obtained by *no NF* and *no NF, 2 enc.*, we observe that there is a clear advantage, especially on circles, by using a single encoder, highlighting the fact that the integration between generative and self-supervised models is effective not only at the objective level but also at the architectural one; (iii) From the comparison between *GEDI* and *no NF* on circles, we observe improved performance and reduced variance. This suggests that $\mathcal{L}_{NF}(\Theta)$ complements the other terms in our objective and therefore contributes to driving the learning towards desired solutions.

E HYPERPARAMETERS FOR SYNTHETIC DATA

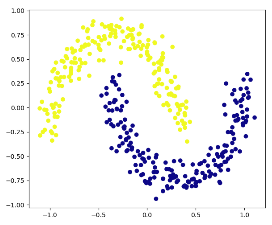
For the backbone enc, we use a MLP with two hidden layers and 100 neurons per layer, an output layer with 2 neurons and LeakyReLU activation functions. For the projection head $proj$, we use a MLP with one hidden layer and 4 and an output layer with 2 neurons (batch normalization is used in all layers) and final L_2 normalization. For JEM, we use an output layer with only one neuron. All methods use a batch size of 400. Baseline JEM (following the original paper):

- Number of iterations $100K$
- Learning rate $1e-3$
- Optimizer Adam $\beta_1 = 0$, $\beta_2 = 0.9$
- SGLD steps 1
- Buffer size 10000
- Reinitialization frequency 0.05
- SGLD step-size 1
- SGLD noise 0.01

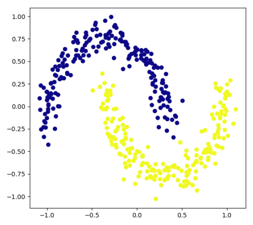
And for self-supervised learning methods, please refer to Table 5.



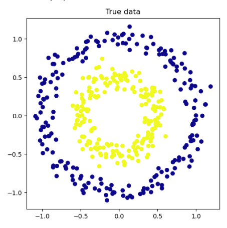
(a) Ground truth



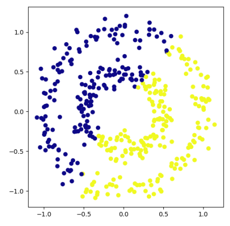
(b) SwAV



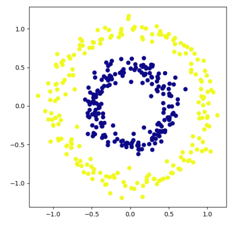
(c) GEDI



(d) Ground truth

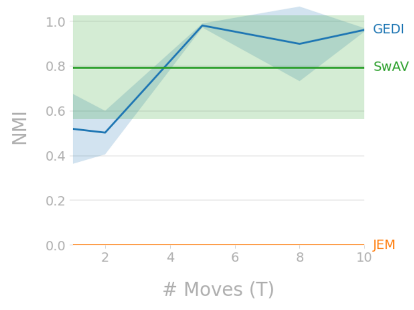


(e) SwAV

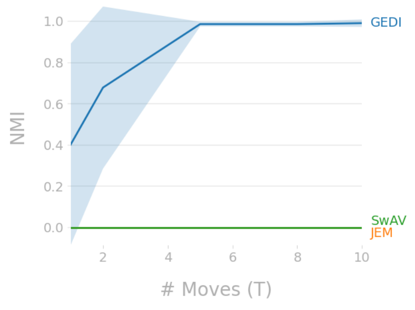


(f) GEDI

Figure 6: Qualitative visualization of the clustering performance for the different strategies (only SwAV and GEDI are shown) on moons (a-c) and on circles (d-f) datasets. Colors identify different cluster predictions.

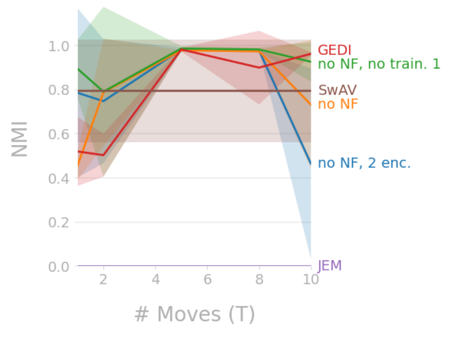


(a) Moons

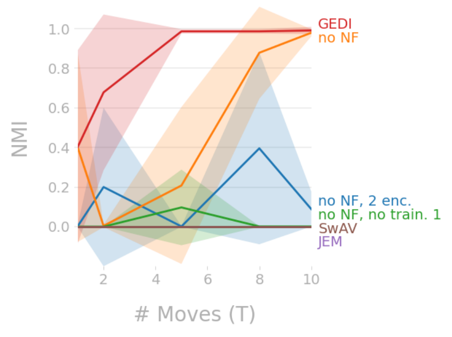


(b) Circles

Figure 7: NMI achieved by JEM, SwAV and GEDI on moons and circles dataset. The curves are obtained by choosing different values of T for DAM, namely $T \in \{1, 2, 5, 8, 10\}$.



(a) Moons



(b) Circles

Figure 8: Ablation study for GEDI on moons and circles dataset. The curves are obtained by choosing different values of T for DAM, namely $T \in \{1, 2, 5, 8, 10\}$.

F Hyperparameters for SVHN, CIFAR-10, CIFAR-100

Table 4: Resnet architecture. Conv2D(A,B,C) applies a 2d convolution to input with B channels and produces an output with C channels using stride (1, 1), padding (1, 1) and kernel size (A, A).

Name	Layer	Res. Layer

Block 1	Conv2D(3,3,F)
	LeakyRELU(0.2)
	Conv2D(3,F,F)
	AvgPool2D(2)
		AvgPool2D(2)
		Conv2D(1,3,F) no padding

	Sum	

Block 2	LeakyRELU(0.2)
	Conv2D(3,F,F)
	LeakyRELU(0.2)
	Conv2D(3,F,F)
	AvgPool2D(2)	

Block 3	LeakyRELU(0.2)
	Conv2D(3,F,F)
	LeakyRELU(0.2)
	Conv2D(3,F,F)	

Block 4	LeakyRELU(0.2)
	Conv2D(3,F,F)
	LeakyRELU(0.2)
	Conv2D(3,F,F)
	AvgPool2D(all)	

For the backbone enc, we use a ResNet with 8 layers as in Duvenaud et al. (2021), where its architecture is shown in Table 4. For the projection head $proj$, we use a MLP with one hidden layer and $2 * F$ neurons and an output layer with F neurons (batch normalization is used in all layers) and final L_2 normalization. $F = 128$ for SVHN, CIFAR-10 (1 million parameters) and $F = 256$ for CIFAR-100 (4.1 million parameters). For JEM, we use the same settings of Duvenaud et al. (2021). All methods use a batch size of 64. Baseline JEM (following the original paper):

- Number of epochs 100
- Learning rate $1e-4$
- Optimizer Adam
- SGLD steps 20
- Buffer size 10000
- Reinitialization frequency 0.05
- SGLD step-size 1
- SGLD noise 0.01
- Data augmentation (Gaussian noise) 0.03

And for self-supervised learning methods, please refer to Table 5.

G Ablation Study on SVHN, CIFAR-10, CIFAR-100

We conduct an ablation study to understand the impact of the different components of GEDI. We compare four different versions of GEDI, namely the full version (called simply *GEDI*), GEDI trained without $\mathcal{L}_{NF}(\Theta)$ (called *no NF*), GEDI trained without the first stage and also without

Table 5: Hyperparameters (in terms of sampling, optimizer, objective and data augmentation) used in all experiments.

Class	**Name param.**	**SVHN**	**CIFAR-10**	**CIFAR-100**	**Addition**
Data augment.	Color jitter prob.	0.1	0.1	0.1	0.1
	Gray scale prob.	0.1	0.1	0.1	0.1
	Random crop	Yes	Yes	Yes	Yes
	Additive Gauss. noise (std)	0.03	0.03	0.03	0.3
	Random horizontal flip	No	Yes	Yes	No
			Training 1		
SGLD	SGLD iters	20	20	20	10
	Buffer size	10k	10k	10k	10k
	Reinit. frequency	0.05	0.05	0.05	0.05
	SGLD step-size	1	1	1	1
	SGLD noise	0.01	0.01	0.01	0.01
Optimizer	Batch size	64	64	64	60
	Iters$_1$	100k	70k	70k	Sec. I
	Adam β_1	0.9	0.9	0.9	0.9
	Adam β_2	0.999	0.999	0.999	0.999
	Learning rate	$1e-4$	$1e-4$	$1e-4$	$1e-4$
			Training 2		
SGLD	SGLD iters	idem	idem	idem	idem
	Buffer size	idem	idem	idem	idem
	Reinit. frequency	idem	idem	idem	idem
	SGLD step-size	idem	idem	idem	idem
	SGLD noise	idem	idem	idem	idem
Optimizer	Batch size	idem	idem	idem	idem
	Iters$_2$	idem	idem	idem	idem
	Adam β_1	idem	idem	idem	idem
	Adam β_2	idem	idem	idem	idem
	Learning rate	idem	idem	idem	idem
DAM	ϵ	0.03	0.03	0.03	0.03
	T	10	10	10	10
Weights for losses	$-CE(p, p_\Psi)$	1	1	1	1
	$\mathcal{L}_{NF}(\Theta)$	1/batch	1/batch	1/batch	1/batch
	$\mathcal{L}_{DI}(\Theta, \mathcal{T})$	1000	1000	1000	1000
	$\mathcal{L}_{DI}(\Theta, p_\Psi)$	500	500	500	500
	$\mathcal{L}_{NeSY}(\Theta)$	-	-	-	0/3000

Table 6: Ablation study for clustering performance in terms of normalized mutual information on test set (SVHN, CIFAR-10, CIFAR-100). Higher values indicate better clustering performance.

Dataset	**no NF, 2 enc. (Ours)**	**no NF (Ours)**	**GEDI (Ourss)**	**Gain**
SVHN	0.21	0.31	**0.39**	**+0.08**
CIFAR-10	0.38	**0.40**	**0.41**	+0.00
CIFAR-100	**0.75**	**0.76**	0.72s	-0.04

$\mathcal{L}_{NF}(\Theta)$ (called *no NF, no train. 1*) and GEDI trained without $\mathcal{L}_{NF}(\Theta)$ using two different encoders for computing the discriminative and the generative terms in our objective (called *no NF, 2 enc.*). Results are shown in Table 6.

Table 7: Supervised linear evaluation in terms of accuracy on test set (SVHN, CIFAR-10, CIFAR-100). The linear classifier is trained for 100 epochs using SGD with momentum, learning rate $1e-3$ and batch size 100.

Dataset	**JEM**	**Barlow**	**SwAV**	**No NF, 2 enc.**	**No NF**	**GEDI**
SVHN	0.20	0.84	0.44	0.29	0.52	0.56
CIFAR-10	0.23	0.63	0.53	0.48	0.50	0.51
CIFAR-100	0.03	0.35	0.14	0.12	0.15	0.15

Table 8: Generative performance in terms of Frechet Inception Distance (FID) (SVHN, CIFAR-10, CIFAR-100). The lower the values the better the performance are.

Dataset	**JEM**	**Barlow**	**SwAV**	**No NF, 2 enc.**	**No NF**	**GEDI**
SVHN	166	454	489	**158**	173	208
CIFAR-10	250	413	430	**209**	265	236
CIFAR-100	240	374	399	**210**	237	244

Table 9: OOD detection in terms of AUROC on test set (CIFAR-10, CIFAR-100). Training is performed on SVHN.

Dataset	**JEM**	**Barlow**	**SwAV**	**No NF, 2 enc.**	**No NF**	**GEDI**
CIFAR-10	0.75	0.43	0.21	0.76	**0.97**	0.94
CIFAR-100	0.75	0.5	0.28	0.75	**0.94**	**0.93**

Table 10: OOD detection in terms of AUROC on test set (SVHN, CIFAR-100). Training is performed on CIFAR-10.

Dataset	**JEM**	**Barlow**	**SwAV**	**No NF, 2 enc.**	**No NF**	**GEDI**
SVHN	**0.43**	0.31	0.24	**0.43**	0.31	0.31
CIFAR-100	0.54	**0.56**	0.51	0.53	0.53	**0.55**

Table 11: OOD detection in terms of AUROC on test set (SVHN, CIFAR-10). Training is performed on CIFAR-100.

Dataset	**JEM**	**Barlow**	**SwAV**	**No NF, 2 enc.**	**No NF**	**GEDI**
SVHN	0.48	0.43	**0.50**	0.47	0.32	0.38
CIFAR-10	0.48	0.42	0.46	0.47	**0.49**	**0.50**

H Additional Experiments on SVHN, CIFAR-10, CIFAR-100

We conduct a linear probe evaluation of the representations learnt by the different models Table 7. These experiments provide insights on the capabilities of learning representations producing linearly separable classes. From Table 7, we observe a large difference in results between Barlow and SwAV. Our approach provides interpolating results between the two approaches.

We also evaluate the generative performance in terms of Frechet Inception Distance (FID). From Table 8, we observe that GEDI outperforms all self-supervised baselines by a large margin, achieving comparable performance to JEM.

Additionally, we evaluate the performance in terms of OOD detection, by following the same methodology used in Grathwohl et al. (2020). We use the OOD score criterion proposed in Grathwohl et al. (2020), namely $s(x) = -\|\frac{\partial \log p_{\Psi}(x)}{\partial x}\|_2$. From Table 9, we observe that GEDI achieves almost optimal performance. While these results are exciting, it is important to mention that they are not generally valid. Indeed, when training on CIFAR-10 and performing OOD evaluation on the other datasets, we observe that all approaches achieve similar performance both on CIFAR-100 and SVHN, suggesting that all datasets are considered in-distribution, see Table 10. A similar observation is obtained when training on CIFAR-100 and evaluating on CIFAR-10 and SVHN, see Table 11. Importantly, this is a phenomenon which has been only recently observed by the scientific community on generative models. Tackling this problem is currently out of the scope of this work. For further discussion about the issue, we point the reader to the works in Nalisnick et al. (2019).

I Details on the MNIST addition experiment.

We now discuss the details on the MNIST addition experiment.

I.1 Hyperparameters

For the backbone enc, we use a ResNet with 8 layers as in Duvenaud et al. (2021), where its architecture is shown in Table 4. For the projection head $proj$, we use a MLP with one hidden layer and 256 neurons and an output layer with 128 neurons (batch normalization is used in all layers) and final L_2 normalization. The number of epochs for both training phases for the three settings, i.e. 100 examples, 1000 examples and 10000 examples are 100, 30 and 5 epochs respectively. These were selected by the point at which the loss curves flatten out.

I.2 Data generation

The data was generated by uniformly sampling pairs a, b such that $0 \leq a \leq 9$, $0 \leq b \leq 9$ and $0 \leq a + b \leq 9$. For each triplet (a, b, c), we assigned to a, b, c, random MNIST images with corresponding labels, without replacement.

I.3 Calculating the constraint

To calculate the constraint, we group the three images of each triplet consecutively in the batches, hence why the batch size is a multiple of 3. To calculate the probability of the constraint, we used an arithmetic circuit compiled from the DeepProbLog program that implements this constraint Manhaeve et al. (2018).