Energy-Based Contrastive Learning of Visual Representations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Contrastive learning is a method of learning visual representations by training Deep Neural Networks (DNNs) to increase the similarity between representations of positive pairs (transformations of the same image) and reduce the similarity between representations of negative pairs (transformations of different images). Here we explore Energy-Based Contrastive Learning (EBCLR) that leverages the power of generative learning by combining contrastive learning with Energy-Based Models (EBMs). EBCLR can be theoretically interpreted as learning the joint distribution of positive pairs, and it shows promising results on small and medium-scale datasets such as MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. Specifically, we find EBCLR demonstrates from $\times 4$ up to $\times 20$ acceleration compared to SimCLR and MoCo v2 in terms of training epochs. Furthermore, in contrast to SimCLR, we observe EBCLR achieves nearly the same performance with 254 negative pairs (batch size 128) and 30 negative pairs (batch size 16) per positive pair, demonstrating the robustness of EBCLR to small numbers of negative pairs. Hence, EBCLR provides a novel avenue for improving contrastive learning methods that usually require large datasets with a significant number of negative pairs per iteration to achieve reasonable performance on downstream tasks.

1 Introduction

In computer vision, supervised learning requires a large-scale human-annotated dataset of images to train accurate deep neural networks (DNNs). However, acquiring labels for millions of images can be difficult or impossible in practice. This has led to the rise of self-supervised learning, which learns useful visual representations by forcing DNNs to be invariant or equivariant to image transformations. Among self-supervised learning algorithms, contrastive methods are rapidly gaining popularity for their superb performance.

Specifically, contrastive learning methods [1, 2, 3, 4, 5] train DNNs by increasing the similarity between representations of *positive pairs* (transformations of the same image) and decreasing the similarity between representations of *negative pairs* (transformations of different images). The negative pairs prevent DNNs from collapsing to the trivial solution, i.e., the constant function. There are numerous variants of contrastive learning methods, such as SimCLR [4], Momentum Contrast (MoCo) [5], etc.

Despite this flurry of research in contrastive learning, contrastive methods require large datasets and a large number of negative pairs per positive pair to achieve reasonable performance on downstream tasks. Although there are recently proposed non-contrastive methods such as BYOL [6] and SimSiam [7] that do not rely on negative pairs, they require heuristic techniques such as stop-gradient to avert collapsing to the trivial solution. There has been an effort to explain the dynamics of non-contrastive methods with linear neural networks [3], but it is unclear how the analyses generalize to DNNs.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

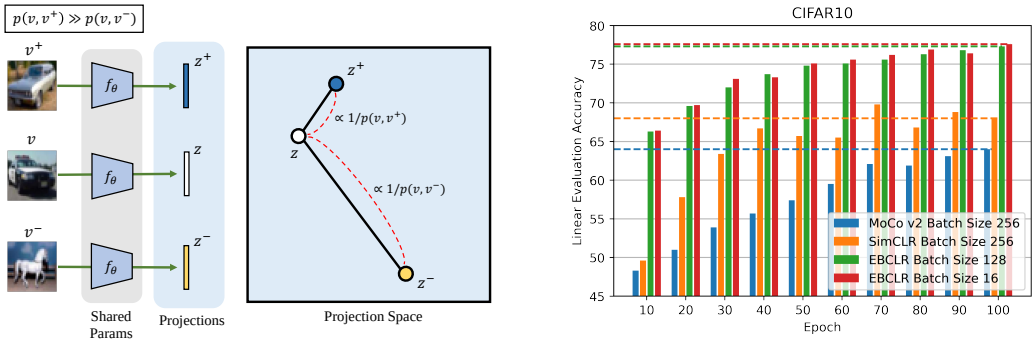


Figure 1: **Left:** An illustration of EBCLR. Here, $\propto$ means "is a monotonically increasing function of". We use $p(v, v')$, the joint distribution of positive pairs, as a measure of semantic similarity of images. Specifically, $p(v, v')$ will be high when v and v' are semantically similar, and low otherwise. A DNN f_θ is trained such that the distance in the projection space is controlled by $1/p(v, v')$. **Right:** Comparison of EBCLR, SimCLR, and MoCo v2 on CIFAR10 in terms of linear evaluation accuracy. EBCLR at epoch 10 beats MoCo v2 at epoch 100, and EBCLR at epoch 20 beats SimCLR and MoCo v2 at epoch 100. Moreover, EBCLR shows identical performance regardless of whether we use 254 negative pairs (batch size 128) or 30 negative pairs (batch size 16) per positive pair.

In this paper, we explore a novel avenue in visual representation learning: Energy-Based Contrastive Learning (EBCLR) which leverages the power of generative learning [8, 9, 10] by combining contrastive learning with energy-based models (EBMs). EBCLR complements the contrastive learning loss with a generative loss, and it can be interpreted as learning the joint distribution of positive pairs. In fact, we demonstrate that the existing contrastive loss is a special case of the EBCLR loss if the generative term is not used. Although EBMs are notorious for being difficult to train due to their reliance on Stochastic Gradient Langevin Dynamics (SGLD) [11], another important contribution of this work is that we overcome this by appropriate modifications to SGLD.

Extensive experiments on a variety of small and medium-scale datasets demonstrate that EBCLR is robust to small numbers of negative pairs, and it outperforms SimCLR and MoCo v2 [12] in terms of sample efficiency and linear evaluation accuracy. Hence, EBCLR opens up a new research direction for alleviating the dependence of contrastive methods on large datasets and large batches.

Our contributions can be summarized as follows:

- We propose a novel contrastive learning method called EBCLR which learns the joint distribution of positive pairs. We show that EBCLR loss is equivalent to a combination of a contrastive term and a generative term (Section 3). To the best of our knowledge, this is the first work to apply EBMs to contrastive learning of visual representations.
- We show that EBCLR offers two advantages over conventional contrastive learning methods: EBCLR is several times more sample efficient (Section 4.1) and robust to small batch sizes (Section 4.2). These factors lead to a non-trivial performance gain for EBCLR.
- We perform thorough ablation studies of the components of EBCLR: effect of changing the weight of the generative term (Section 4.3), effect of projection space dimension (Section 4.3), and the effect of the proposed SGLD modifications (Section 4.4).

2 Related Works

In this section, we go over related works necessary for understanding EBCLR. In Appendix A, we give a more extensive review of relevant works for those not familiar with EBMs, contrastive learning, or generative models.

2.1 Contrastive Learning

For a given batch of images $\{x_n\}_{n=1}^N$ and two image transformations t, t', contrastive learning methods first create two views $v_n = t(x_n)$, $v'_n = t'(x_n)$ of each instance x_n. Here, the pair (v_n, v'_m)

is called a *positive pair* if $n = m$ and a *negative pair* if $n \neq m$. Given a DNN f_θ, the views are then embedded into the projection space by passing the views through f_θ and normalizing.

Contrastive methods train f_θ to increase agreement between projections of positive pairs and decrease agreement between projections of negative pairs. Specifically, f_θ is trained to maximize the InfoNCE objective [1]. After training, outputs from the final layer or an intermediate layer of f_θ are used for downstream tasks.

There are numerous variants of contrastive methods. For instance, SimCLR [4] uses a composition of random cropping, random flipping, color jittering, color dropping, and blurring as the image transformation. Negative pairs are created by transforming different images within a batch. On the other hand, MoCo [12] maintains a queue of negative samples, so negative samples are not limited to views of images from the same batch.

2.2 Energy-Based Models

Given a scalar-valued *energy function* $E_\theta(v)$ with parameter θ, an energy-based model (EBM) [13] defines a distribution by the formula

$$q_\theta(v) \coloneqq \frac{1}{Z(\theta)} \exp\{-E_\theta(v)\} \tag{1}$$

where $Z(\theta)$ is the *partition function* which guarantees q_θ integrates to 1. Since there are essentially no restrictions on the choice of the energy function, EBMs have great flexibility in modeling distributions. Hence, EBMs have been applied to a wide variety of machine learning tasks, such as dimensionality reduction via autoencoding [14], learning generative classifiers [8, 9, 10, 15], generating images [16], and training regression models [17, 18]. Wang et al. [19] have explored connections between EBMs and InfoNCE to enhance generative performance of EBMs. However, to the best of our knowledge, this paper is the first to combine EBMs with contrastive learning for representation learning.

Given a target distribution, an EBM can be used to estimate its density p when we can only sample from p. One way of achieving this is by minimizing the Kullback-Leibler (KL) divergence between q_θ and p that maximizes the expected log-likelihood of q_θ under p [20]:

$$\max_\theta \ \mathbb{E}_p[\log q_\theta(v)]. \tag{2}$$

Stochastic gradient ascent can be used to solve (2) [20]. Specifically, the gradient of the expected log-likelihood with respect to the parameters θ can be shown to be

$$\nabla_\theta \mathbb{E}_p[\log q_\theta(v)] = \mathbb{E}_{q_\theta}[\nabla_\theta E_\theta(v)] - \mathbb{E}_p[\nabla_\theta E_\theta(v)]. \tag{3}$$

Hence, updating θ with (3) amounts to pushing up on the energy for samples from q_θ and pushing down on the energy for samples from p. This optimization method is also known as contrastive divergence [21].

While the second term in (3) can be easily calculated as we have access to samples from p, the first term requires sampling from q_θ. Previous works [16, 22, 10, 15] have used Stochastic Gradient Langevin Dynamics (SGLD) [11] to generate samples from q_θ. Specifically, given a sample v_0 from some proposal distribution q_0, the iteration

$$v_{t+1} = v_t - \frac{\alpha_t}{2} \nabla_{v_t} E_\theta(v_t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_t^2) \tag{4}$$

guarantees that the sequence $\{v_t\}$ converges to a sample from q_θ assuming $\{\alpha_t\}$ decays at a polynomial rate [11].

However, SGLD requires an infinite number of steps until samples from the proposal distribution converge to samples from the target distribution. This is unfeasible, so in practice, only a finite number of steps along with constant step size, i.e. $\alpha_t = \alpha$ and constant noise variance $\sigma_t = \sigma^2$ are used [16, 22, 10, 15]. Moreover, Yang and Ji [15] noted SGLD often generates samples with extreme pixel values that cause EBMs to diverge during training. Hence, they have proposed *proximal SGLD* which clamps gradient values into an interval $[-\delta, \delta]$ for a threshold $\delta > 0$. Then, the update equation becomes

$$v_{t+1} = v_t - \alpha \cdot \mathrm{clamp}\{\nabla_{v_t} E_\theta(v_t), \delta\} + \epsilon \tag{5}$$

for $t = 0, \ldots, T-1$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $\mathrm{clamp}\{\cdot, \delta\}$ clamps each element of the input vector into $[-\delta, \delta]$. In our work, we introduce additional modifications to SGLD which accelerate the convergence of EBCLR.

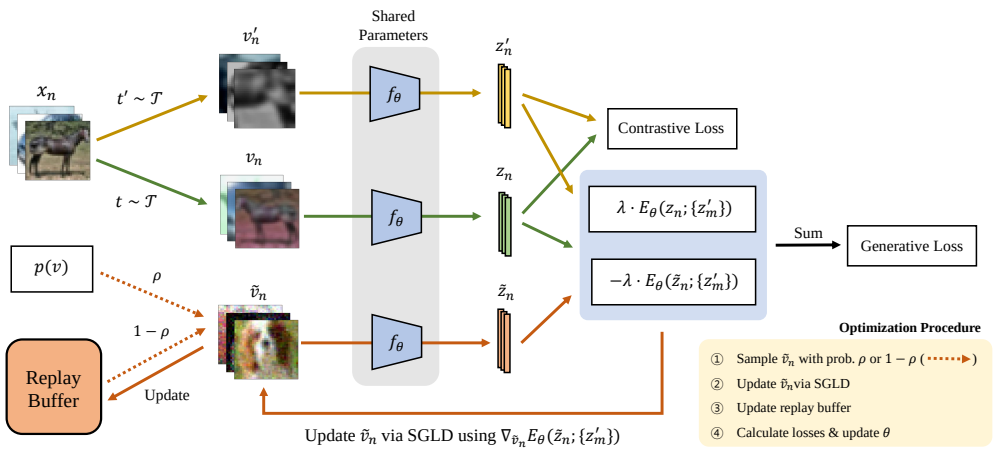


Figure 2: An illustration of the learning process of EBCLR.

3 Theory

3.1 Energy-Based Contrastive Learning

Let $\mathcal{D}$ be a distribution of images and $\mathcal{T}$ a distribution of stochastic image transformations. Given $x \sim \mathcal{D}$ and i.i.d. $t, t' \sim \mathcal{T}$, our goal is to approximate the joint distribution of the views

$$p(v, v'), \quad \text{where } \; v = t(x), \; v' = t'(x)$$

using the model distribution

$$q_\theta(v, v') := \frac{1}{Z(\theta)} \exp\{-\|z - z'\|^2/\tau\}. \tag{6}$$

where $Z(\theta)$ is a normalization constant, $\tau > 0$ is a temperature hyper-parameter, and z and z' are projections computed by passing the views v and v' through the DNN f_θ and then normalizing to have unit norm. We now explain the intuitive meaning of matching q_θ to p.

Our key idea is to use $p(v, v')$ as a measure of semantic similarity of v and v'. If two images v and v' are semantically similar, they are likely to be transformations of similar images. So, $p(v, v')$ will be high when v and v' are semantically similar and low otherwise. Suppose q_θ successfully approximates p. If we equate $p(v, v')$ to $q_\theta(v, v')$ in (6) and solve for $\|z - z'\|$, we see that the distance between z and z' will become a monotone increasing function of $1/p(v, v')$, which is the inverse of semantic similarity of v and v'. So, semantically similar images will have nearby projections, and dissimilar images will have distant projections. This idea is illustrated in Figure 1.

To approximate p using q_θ, we train f_θ to maximize the expected log-likelihood of q_θ under p:

$$\max_\theta \; \mathbb{E}_p[\log q_\theta(v, v')]. \tag{7}$$

In order to solve this problem with stochastic gradient ascent, we could naively extend (3) to the setting of joint distributions to obtain the following result.

Proposition 1. *The the joint distribution* (6) *can be formulated as an EBM*

$$q_\theta(v, v') = \frac{1}{Z(\theta)} \exp\{-E_\theta(z, z')\}, \qquad E_\theta(v, v') = \|z - z'\|^2/\tau \tag{8}$$

and the gradient of the objective of (7) *is given by*

$$\nabla_\theta \mathbb{E}_p[\log q_\theta(v, v')] = \mathbb{E}_{q_\theta}[\nabla_\theta E_\theta(v, v')] - \mathbb{E}_p[\nabla_\theta E_\theta(v, v')]. \tag{9}$$

However, computing the first expectation in (9) requires sampling pairs of views (v, v') from $q_\theta(v, v')$ via SGLD, which could be expensive. To avert this problem, we use Bayes' rule to decompose

$$\mathbb{E}_p[\log q_\theta(v, v')] = \mathbb{E}_p[\log q_\theta(v' \mid v)] + \mathbb{E}_p[\log q_\theta(v)] \quad \text{where} \quad q_\theta(v) = \int q_\theta(v, v')\, dv'. \tag{10}$$

In the first equation of (10), the first and second terms at the RHS will be referred to as discriminative and generative terms, respectively, throughout the paper. A similar decomposition was used by Grathwohl et al. [10] in the setting of learning generative classifiers.

Furthermore, we add a hyper-parameter λ to balance the strength of the discriminative term and the generative term. The advantage of this modification will be discussed in Section 4.3. This yields our Energy-Based Contrastive Learning (EBCLR) objective

$$\mathcal{L}(\theta) \coloneqq \mathbb{E}_p[\log q_\theta(v' \mid v)] + \lambda \mathbb{E}_p[\log q_\theta(v)]. \tag{11}$$

The discriminative term can be easily differentiated since the partition function $Z(\theta)$ cancels out when $q_\theta(v, v')$ is divided by $q_\theta(v)$. However, the generative term still contains $Z(\theta)$. We now present our key result, which addresses this problem by relating the generative term to EBMs. The proof is deferred to Appendix C.1.

Theorem 2. *The marginal distribution in* (10) *can be formulated as an EBM*

$$q_\theta(v) = \frac{1}{Z(\theta)} \exp\{-E_\theta(v)\}, \qquad E_\theta(v) \coloneqq -\log \int e^{-\|z-z'\|^2/\tau}\, dv' \tag{12}$$

where $Z(\theta)$ is the partition function in (6)*, and the gradient of the generative term is given by*

$$\nabla_\theta \mathbb{E}_p[\log q_\theta(v)] = \mathbb{E}_{q_\theta(v)}[\nabla_\theta E_\theta(v)] - \mathbb{E}_p[\nabla_\theta E_\theta(v)]. \tag{13}$$

Thus, the gradient of the EBCLR objective is

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_p[\nabla_\theta \log q_\theta(v' \mid v)] + \lambda \mathbb{E}_{q_\theta(v)}[\nabla_\theta E_\theta(v)] - \lambda \mathbb{E}_p[\nabla_\theta E_\theta(v)] \tag{14}$$

Theorem 2 suggests that the EBM for the joint distribution can be learned by computing the gradients of the discriminative term and the EBM for the marginal distribution. Moreover, we only need to sample v from $q_\theta(v)$ to compute the second expectation in (14).

3.2 Approximating the EBCLR Objective

To implement EBCLR, we need to approximate expectations in (11) with their empirical means. Suppose samples $\{(v_n, v'_n)\}_{n=1}^N$ from $p(v, v')$ are given, and let $\{(z_n, z'_n)\}_{n=1}^N$ be the corresponding projections. As the learning goal is to make $q_\theta(v_n, v'_n)$ approximate the joint probability density function $p(v_n, v'_n)$, the empirical mean $\widehat{q}_\theta(v_n)$ can be defined as:

$$\widehat{q}_\theta(v_n) = \frac{1}{N'} \sum_{v'_m : v'_m \neq v_n} q_\theta(v_n, v'_m) \tag{15}$$

where the sum is over the collection of v'_m defined as

$$\{v'_m : v'_m \neq v_n\} \coloneqq \{v_k\}_{k=1}^N \cup \{v'_k\}_{k=1}^N - \{v_n\} \tag{16}$$

and $N' \coloneqq |\{v'_m : v'_m \neq v_n\}| = 2N - 1$. One could also use a simpler form of the empirical mean:

$$\widehat{q}_\theta(v_n) = \frac{1}{N} \sum_{m=1}^N q_\theta(v_n, v'_m) \tag{17}$$

Similarly, $q_\theta(v'|v)$ in (11), which should approximate the conditional probability density $p(v'|v)$, can be represented in terms of $q_\theta(v_n, v'_n)$. Specifically, we have

$$q_\theta(v'_n \mid v_n) \simeq \frac{q_\theta(v_n, v'_n)}{\widehat{q}_\theta(v_n)} = \frac{q_\theta(v_n, v'_n)}{\frac{1}{N'} \sum_{v'_m : v'_m \neq v_n} q_\theta(v_n, v'_m)} = \frac{e^{-\|z_n - z'_n\|^2/\tau}}{\frac{1}{N'} \sum_{v'_m : v'_m \neq v_n} e^{-\|z_n - z'_m\|^2/\tau}} \tag{18}$$

It is then immediately apparent that the empirical form of the discriminative term using (18) is a particular instance of the contrastive learning objective such as InfoNCE and SimCLR. Hence, EBCLR can be interpreted as complementing contrastive learning with a generative term defined by an EBM. We will demonstrate in Section 4.1 that the generative term offers significant advantages over other contrastive learning methods.

For the second term, we use the simpler form of the empirical mean in (17):

$$\widehat{q}_\theta(v_n) = \frac{1}{N}\sum_{m=1}^{N} q_\theta(v_n, v'_m) = \frac{1}{Z(\theta)} \cdot \frac{1}{N}\sum_{m=1}^{N} \exp\{-\|z_n - z'_m\|^2/\tau\} \tag{19}$$

We could also use (15) as the empirical mean, but either choice showed identical performance (c.f. Appendix E.3). So, we have found (15) to be not worth the additional complexity, and have resorted to the simpler approximation (17) instead. In Appendix C.2, we theoretically justify that EBCLR will work as intended even with the approximations (15) or (17). If we compare (19) with (12), we can see that this approximation of $q_\theta(v)$ yields the energy function (after ignoring the constant $\log N$)

$$E_\theta(v; \{v'_m\}_{m=1}^N) := -\log\left(\sum_{m=1}^{N} e^{-\|z - z'_m\|^2/\tau}\right). \tag{20}$$

3.3 Modifications to SGLD

According to Theorem 2, we need samples from the marginal $q_\theta(v)$ to calculate the second expectation in (14). Hence, we apply proximal SGLD (5) with the energy function (20) to sample from $q_\theta(v)$ as

$$\tilde{v}_{t+1} = \tilde{v}_t - \alpha \cdot \text{clamp}\{\nabla_{\tilde{v}_t} E_\theta(\tilde{v}_t; \{v'_m\}_{m=1}^N), \delta\} + \epsilon \tag{21}$$

for $t = 0, \ldots, T-1$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We make three additional modifications to proximal SGLD to expedite the training process. From here on, we will be referring to proximal SGLD in (5) when we say SGLD.

First, we initialize SGLD from generated samples from previous iterations, and with probability ρ, we reinitialize SGLD chains from samples from a proposal distribution q_0. This is achieved by keeping a replay buffer $\mathcal{B}$ of SGLD samples from previous iterations. This technique of maintaining a replay buffer has also been used in previous works and has proven to be crucial for stabilizing and accelerating the convergence of EBMs [16, 10, 15].

Second, the proposal distribution q_0 is set to be the data distribution $p(v)$. This choice differs from those of previous works [16, 10, 15] which have either used the uniform distribution or a mixture of Gaussians as the proposal distribution.

Finally, we use *multi-stage SGLD (MSGLD)*, which adaptively controls the magnitude of noise added in SGLD. For each sample $\tilde{v}$ in the replay buffer $\mathcal{B}$, we keep a count $\kappa_{\tilde{v}}$ of number of times it has been used as the initial point of SGLD. For samples with a low count, we use noise of high variance, and for samples with a high count, we use noise of low variance. Specifically, in (5), we set

$$\sigma = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min}) \cdot [1 - \kappa_{\tilde{v}}/K]_+. \tag{22}$$

where $[\cdot]_+ := \max\{0, \cdot\}$, $\sigma^2_{\max}$ and $\sigma^2_{\min}$ are the upper and lower bounds on the noise variance, respectively, and K controls the decay rate of noise variance. The purpose of this technique is to facilitate quick exploration of the modes of q_θ and still guarantee SGLD generates samples with sufficiently low energy. The pseudocodes for MSGLD and EBCLR are given in Algorithms 1 and 2, respectively, in Appendix B, and the overall learning flow of EBCLR is described in Figure 2.

4 Experiments

We now describe the experimental settings. A complete description is deferred to Appendix D.

Baseline methods and datasets. The baseline methods are SimCLR, MoCo v2, SimSiam, and BYOL. The hyper-parameters are chosen closely following the original works [4, 12, 7, 6]. We use four datasets: MNIST [23], Fashion MNIST (FMNIST) [24], CIFAR10, and CIFAR100 [25].

DNN architecture. We decompose $f_\theta = \pi_\theta \circ \phi_\theta$ where ϕ_θ is the encoder network and π_θ is the projection network. Rather than using the output of f_θ for downstream tasks, we follow previous works [4, 5, 1, 2, 3, 6, 7] and use the output of ϕ_θ instead. In our experiments, we set ϕ_θ to be a ResNet-18 [26] up to the global average pooling layer and π_θ to be a 2-layer MLP with output dimension 128. However, we remove batch normalization because batch normalization hurts SGLD [16]. We also replace ReLU with leaky ReLU to expedite the convergence of SGLD. For the baselines, we use settings proposed in the original works while keeping the backbone fixed to be ResNet-18.

Evaluation. We evaluate the representations by training a linear classifier on top of frozen ϕ_θ.

Dataset	MNIST		FMNIST		CIFAR10		CIFAR100	
Statistic	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.
SimSiam	98.6	0.1	87.4	0.1	70.4	0.25	38.3	0.1
BYOL	**99.3**	0.4	89.0	0.2	70.9	0.25	41.7	0.2
SimCLR	99.0	0.1	88.5	0.15	68.0	0.15	43.1	0.25
MoCo v2	98.1	0.05	87.8	0.1	64.0	0.1	38.2	0.1
EBCLR	**99.3**	–	**90.1**	–	**77.3**	–	**49.1**	–

Table 1: Linear evaluation accuracy and efficiency relative to EBCLR. Efficiency of a method relative to EBCLR is calculated by the following formula: (number of epochs used by EBCLR to reach the final accuracy of the method) / (total number of training epochs).

4.1 Comparison with Baselines

We use batch size 128 for EBCLR and batch size 256 for the baseline methods following Wang et. al [27] and train each method for 100 epochs. Table 1 shows the result of training each method for 100 epochs. Observe that EBCLR consistently outperforms all baseline methods in terms of linear evaluation accuracy. Moreover, relative efficiency indicates EBCLR is capable of achieving the same level of performance as the baseline methods with much fewer training epochs. Concretely, we observe at least $\times 4$ acceleration in terms of epochs compared to contrastive methods. Hence, EBCLR is a much more desirable choice than SimCLR or MoCo v2 for learning visual representations when we have a small number of training samples.

Direction	**M** $\rightarrow$ **FM**	**FM** $\rightarrow$ **M**	**C10** $\rightarrow$ **C100**	**C100** $\rightarrow$ **C10**
SimSiam	86.9	97.2	39.5	64.0
BYOL	**87.3**	97.8	42.3	70.2
SimCLR	86.9	97.4	39.9	67.3
MoCo v2	85.3	97.1	36.2	62.9
EBCLR	**87.4**	**98.5**	**46.9**	**72.4**

Table 2: Comparison of transfer learning results in the linear evaluation setting. Left side of the arrow is the dataset than the encoder was pre-trained on, and right side of the arrow is the dataset that linear evaluation was performed on. We use the following abbreviations. **M** : MNIST, **FM** : FMNIST, **C10** : CIFAR10, **C100** : CIFAR100.

We also investigate the transfer learning performance of EBCLR. Table 2 compares the transfer learning accuracies. EBCLR always outperforms the baseline methods, and the performance gap is especially large on CIFAR10 and CIFAR100. This indicates EBCLR learns visual representations that generalize well across datasets. Repeating the above experiments with longer training or KNN classification led to similar conclusions (c.f. Appendixes E.1 and E.4, respectively).

4.2 Effect of Reducing Negative Pairs

We compared the performances of EBCLR and SimCLR as we reduced the number of negative pairs per positive pair. For MoCo v2, the negative samples are provided by a queue updated by a momentum encoder. On the other hand, for EBCLR and SimCLR, negative samples come from the same batch as the positive pair. So, we did not have a way of fairly comparing EBCLR and SimCLR with MoCo v2. Hence, we excluded MoCo v2 from this experiment.

We note that, according to (18), given a batch of size N, we obtain $2N - 2$ negative pairs for each positive pair. SimCLR also has $2N - 2$ negative pairs for each positive pair. Hence, we can conveniently compare the sensitivity of EBCLR and SimCLR to the number of negative pairs by varying the batch size.

Table 3 shows the result of training each method for 100 epochs with batch sizes in $\{16, 64, 128\}$. We make three important observations. First, EBCLR consistently beats SimCLR in terms of linear evaluation accuracy for every batch size. Second, EBCLR is invariant to the choice of batch size. This contrasts with SimCLR whose performance degrades as batch size decreases. Consequently, EBCLR with batch size 16 beats SimCLR with batch size 128. Finally, as a byproduct of the second observation, the efficiency of EBCLR relative to SimCLR increases as batch size decreases. These properties make EBCLR suitable for situations where we cannot use large batch sizes, e.g., when we

Dataset	MNIST			FMNIST			CIFAR10			CIFAR100		
Batch Size	16	64	128	16	64	128	16	64	128	16	64	128
SimCLR	98.7	99.1	99.1	87.1	88.0	88.2	65.2	67.6	69.0	36.9	39.1	43.0
EBCLR	**99.4**	**99.3**	**99.3**	**89.6**	**90.4**	**90.1**	**77.6**	**78.2**	**77.3**	**48.8**	**49.8**	**49.1**
Rel. Eff.	0.05	0.1	0.15	0.05	0.15	0.1	0.1	0.15	0.2	0.1	0.15	0.25

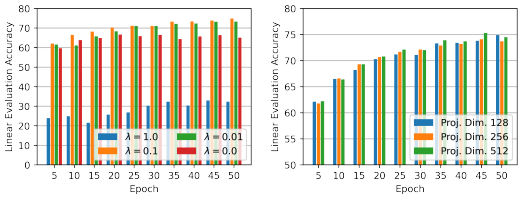
Table 3: Linear evaluation accuracies and efficiencies relative to EBCLR with various batch sizes. Efficiency of SimCLR relative to EBCLR is calculated by the following formula: (number of epochs used by EBCLR with the same batch size to reach the final accuracy of SimCLR) / (total number of training epochs).

have a small number of GPUs. Repeating the experiments with longer training or KNN classification again led to similar conclusions (c.f. Appendixes E.2 and E.4, respectively).

4.3 Effect of λ and Projection Dimension

We explored the effect of changing the hyperparameter λ which controls the importance of the generative term relative to the discriminative term (c.f. Equation (11)). Figure 3a shows the performance of EBCLR with various values of λ as training progresses. We observe that naively using $\lambda = 1.0$ leads to poor results. The performance peaks at $\lambda = 0.1$, and then degrades as we further decrease λ.

This result has two crucial implications. First, the generative term plays a non-trivial role in EBCLR. Second, we need to strike a right balance between the discriminative term and the generative term to achieve good performance on downstream tasks[1].



(a) Effect of λ. (b) Effect of proj. dim.

Figure 3: Effect of λ and projection dimension (output dimension of π_θ, demonstrated on CIFAR10.

We also investigated the effect of varying the output dimension of π_θ. Figure 3b shows linear evaluation results for projection dimensions in $\{128, 256, 512\}$. We observe that the projection dimension has essentially no influence on the training process. In this respect, EBCLR resembles SimCLR which is also invariant to the output dimension (c.f. Figure 8 in the work by Chen et. al [4]).

4.4 Effect of SGLD Modifications

We now study the roles of the three SGLD modifications proposed in Section 3.3. Figure 4 shows the results of varying one parameter of MSGLD while keeping the others fixed.

Effect of reintialization frequency ρ. Figure 4a displays linear evaluation results for $\rho \in \{0.0, 0.2, 1.0\}$. We note that setting $\rho = 1.0$ is equivalent to removing the replay buffer. Also, setting $\rho = 0.0$ is equivalent to never reinitializing SGLD chains.

Initially, $\rho = 0.0$ shows the best performance, as SGLD quickly reaches samples of lower energy. However, learning then slows down because of the lack of diversity of samples in the replay buffer $\mathcal{B}$. This implies that it is necessary to set $\rho > 0$ in order to learn good representations.

On the other hand, $\rho = 1.0$ shows slow convergence in the beginning because samples in the replay buffer are not given enough iterations to reach low energy. Although it does beat $\rho = 0.0$ at latter epochs, it still often performs worse than $\rho = 0.2$. Moreover, it is not sample-efficient compared to $\rho = 0.2$ since we have to provide an entire batch of new samples for reinitializing SGLD chains at each iteration.

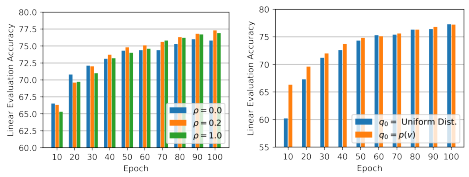
[1] Interestingly, we observed a similar phenomenon when we used models trained with EBCLR to generate images. For more details, we refer the readers to Appendix E.5.

Given the above observations, it is clear why the intermediate value 0.2 is the best choice out of $\rho \in \{0.0, 0.2, 1.0\}$. $\rho = 0.2$ allows enough time for samples in the replay buffer to reach low energy while still maintaining the diversity of samples in $\mathcal{B}$. Also, it is sample-efficient compared to $\rho = 1.0$.

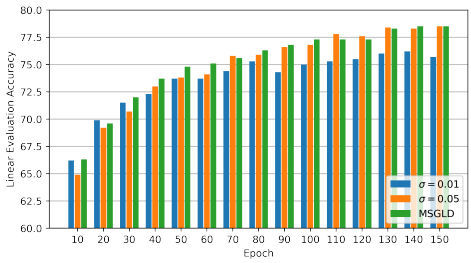
Effect of proposal distribution q_0. Figure 4b compares linear evaluation accuracies with q_0 as the uniform distribution and $q_0 = p(v)$. We observe prominent acceleration in the initial epochs for $q_0 = p(v)$. Hence, we can conclude that this choice of proposal distribution is crucial for the high efficiency of EBCLR compared to the baseline methods in Tables 1 and 3.

We believe this acceleration effect can be explained by the work of Hinton [21]. Specifically, let us observe that the EBM update equation (3) pushes up the energy on the model distribution q_θ. In the implementation of EBCLR with $q_0 = p(v)$, however, q_θ is replaced by the distribution of samples created by a finite number of (noisy) gradient steps on real data points (c.f. Section 3.3). Hence, the modified EMB update equation contains the curvature information of the data manifold. This curvature information may expedite the training process of EBCLR. For a detailed discussion on this, we refer the readers to Section 3 of the work by Hinton [21].

Comparison of SGLD and MSGLD. Figure 4c shows results with SGLD with $\sigma \in \{0.01, 0.05\}$ and MSGLD with $\sigma_{\min} = 0.01$ and $\sigma_{\max} = 0.05$. We note that setting $\sigma_{\min} = \sigma_{\max}$ reduces MSGLD to SGLD. We observe $\sigma = 0.01$ initially shows fast convergence but then saturates due to the lack of diversity of generated samples. On the other hand, $\sigma = 0.05$ initially has the worst performance but eventually beats $\sigma = 0.01$ since $\sigma = 0.05$ quickly explores the modes of q_θ. MSGLD inherits the best of both settings. Specifically, MSGLD is as fast as $\sigma = 0.01$ in the beginning, and it does not suffer from the saturation problem.



(a) Effect of varying ρ. (b) Effect of varying q_0.



(c) SGLD with $\sigma \in \{0.01, 0.05\}$ and MSGLD.

Figure 4: Ablation study of SGLD modifications on CIFAR10.

5 Limitations and Societal Impacts

Limitations. The main limitation of our work is of scale. While EBCLR demonstrates superior sample efficiency, it requires inner SGLD iterations (which cannot be parallelized) and a replay buffer $\mathcal{B}$. These two components increase the computational burden of EBCLR. So, we found it difficult to apply EBCLR to large-scale data such as ImageNet. However, we note that inner SGLD iterations and the replay buffer are not particular limitations of EBCLR, but limitations of EBMs in general. Given the increasing efforts to overcome these limitations such as Proximal-YOPO-SGLD (for more discussion, see Appendix F), we believe EBCLR will eventually be applicable to larger data.

Social Impacts. We generally expect positive outcomes from this research. Further development of EBCLR can mitigate the need for large amount of data and large batch sizes to learn good representations and ultimately lead to a reduction in resource consumption.

6 Conclusion

In this work, we proposed EBCLR which combines contrastive learning with EBMs. This amalgamation of ideas has led to both theoretical and practical contributions. Theoretically, EBCLR associates distance in the projection space with the density of positive samples. Since the distribution of positive samples reflects the semantic similarity of images, EBCLR is capable of learning good visual representations. Practically, EBCLR is several times more sample-efficient than conventional contrastive and non-contrastive learning approaches and is robust to small numbers of negative pairs. Hence, EBCLR is applicable even in scenarios with limited data or devices. We believe that EBCLR makes representation learning available to a wider range of machine learning practitioners.

References

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

[2] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv:1906.05849*, 2019.

[3] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *NeurIPS*, 2020.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[6] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv:2006.07733*, 2020.

[7] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

[8] Julia A. Lasserre, Christopher M. Bishop, and Thomas P. Minka. Principled hybrids of generative and discriminative models. In *CVPR*, 2006.

[9] Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *ICML*, 2008.

[10] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swerwky. Your classifier is secretly and energy based model and you should treat it like one. In *ICLR*, 2020.

[11] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

[12] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020.

[13] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. *Predicting Structured Data*, 2006.

[14] Marc'Aurelio Ranzato, Y-Lan Boureau, Sumit Chopra, and Yann LeCun. A unified energy-based framework for unsupervised learning. 2007.

[15] Xiulong Yang and Shihao Ji. Jem++: Improved techniques for training jem. In *ICCV*, 2021.

[16] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. In *NeurIPS*, 2019.

[17] Fredrik K. Gustafsson, Martin Danelljan, Radu Timofte, and Thomas B. Schön. How to train your energy-based model for regression. In *BMVC*, 2020.

[18] Fredrik K. Gustafsson, Martin Danelljan, Goutam Bhat, and Thomas B. Schön. Energy-based models for deep probabilistic regression. In *ECCV*, 2020.

[19] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. A unified contrastive energy-based model for understanding the generative ability of adversarial training. In *ICLR*, 2022.

[20] Yang Song and Diederik P. Kingma. How to train your energy-based models. *arxiv preprint arXiv:2101.03288*, 2021.

[21] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 2002.

[22] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On learning non-convergent short-run mcmc toward energy-based model. In *NeurIPS*, 2019.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[24] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel dataset for benchmarking machine learning algorithms. *arxiv preprint arXiv:1708.07747*, 2017.

[25] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[27] Xudong Wang, Ziwei Liu, and Sella X. Yu. Unsupervised feature learning by cross-level instance-group discrimination. In *CVPR*, 2021.

[28] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020.

[29] Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In *ICML*, 2021.

[30] Hao Chen, Yaohui Wang, Benoit Lagadec, Antitza Dantcheva, and Francois Bremond. Joint generative and contrastive learning for unsupervised person re-identification. In *CVPR*, 2021.

[31] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 2011.

[32] Bradley Efron. The efficiency of logistic regression compared to normal discriminant analysis. *JASA*, 1975.

[33] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NeurIPS*, 2001.

[34] Yoshua Bengio, Pascal Lamblin, Dan Popovoci, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NeurIPS*, 2006.

[35] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *JMLR*, 2010.

[36] Stephen Zhao, Jörn-Henrik Jacobsen, and Will Grathwohl. Joint energy-based models for semi-supervised classification. In *ICML Workshop*, 2020.

[37] Wenzheng Zhang and Karl Stratos. Understanding hard negatives in noise contrastive estimation. 2021.

[38] Diedrik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[39] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[40] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. 2010.

[41] Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No mcmc for me: Amortized sampling for fast and stable training of energy-based models. In *ICLR*, 2021.

[42] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *JMLR*, 2005.

[43] Benjamin Rhodes, Kai Xu, and Michael U. Gutmann. Telescoping density-ratio estimation. 2020.

[44] Ruiqi Gao, Erik Nijkamp, Diederik P. Kingma, Zhen Xu, Andrew M. Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *CVPR*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Each EBM was trained once due to computation costs.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional Background

A.1 Contrastive Divergence and Contrastive Learning

Contrastive divergence, initially proposed by Hinton et al. [21], is a particular method for solving the maximum-likelihood estimation problem for EBMs. Intuitively, it proceeds by minimizing the energy function on data and maximizing the energy function on (possibly short-run) MCMC samples of the current EBM distribution through gradient ascent. In fact, we use an instance of contrastive divergence to optimize the generative term in our EBCLR objective (11).

We also disambiguate contrastive learning from contrastive divergence. They are similar in the sense that they contrast certain pairs of data. Contrastive learning contrasts positive pairs and negative pairs. Contrastive divergence contrasts data and MCMC samples. However, they differ in the aspect that contrastive learning is inherently a discriminative learning method (in contrastive learning, the model "predicts" the positive pair among negative pairs) and contrastive divergence is a generative learning method.

In the context of unsupervised visual representation learning, there are several works which connect contrastive learning with generative learning [28, 29, 30]. For instance, Chen et al. [30] combine contrastive learning with GANs to improve person re-identification. Zimmermann et al. [29] and Wang et al. [28] study the distributional properties of embeddings found by contrastive learning. Our work, for the first time, explores the synergy between contrastive learning (the discriminative term in (11)) and EBMs / contrastive divergence (the generative term in (11)) for learning visual representations.

A.2 Relation to Contrastive Energy-Based Models (CEMs) [19]

According to the terms introduced in the work of Wang et al. [19], our EBCLR model distribution (6) with $\tau = 1$ is identical to the model distribution of Non-Parametric CEM ((4) in [19]). This is because the squared ℓ_2 norm between two unit-norm vectors z and z' is equivalent to the inner product between z and z' up to an additive constant. However, our work differs from the work of Wang et al. [19] in three aspects.

Objective function. Wang et al. [19] optimizes the log-likelihood of the joint model distribution $q_\theta(v, v')$ directly whereas we use Bayes' rule to decompose the objective into two terms, the discriminative term and the generative term. This allows us to reweight the generative term to learn better visual representations (as explored in Section 4.3).

Optimization method. Wang et al. [19] maximizes the log-likelihood of the joint model distribution directly via adversarial training. On the other hand, we use gradient ascent to optimize the discriminative term and contrastive divergence to optimize the generative term. Computation costs are similar, as adversarial training also requires inner iterations to generate adversarial examples at every model update step.

Application of interest. Wang et al. [19] focuses on enhancing image generation, while we focus on learning visual representations useful for downstream tasks.

Wang et al. [19] also establishes a connection between contrastive divergence and InfoNCE (Section 5.1 in [19]). Specifically, Wang et al. [19] show that when we approximate the model marginal $q_\theta(v)$ with the data distribution $p(v)$, MLE gradient for $q_\theta(v, v')$ becomes the InfoNCE loss gradient. Surprisingly, we find this holds in our framework as well. If we set $q_\theta(v) = p(v)$ in (11), the generative term in (11) becomes independent of θ, so we end up optimizing only the discriminative term. We have shown that the discriminative term can be approximated by a contrastive learning objective, so we recover the result of Wang et al. [19]. In this sense, the derivations in Section 3 of our paper generalizes the connection discovered in the work of Wang et al. [19], as we work with weaker assumptions (we do not set $q_\theta(v) = p(v)$ in (11)).

A.3 EBMs for Learning Useful Representations

Ranzato et al. [14] proposes to train an EBM with an autoencoder structure by minimizing the sum of an encoding loss and a decoding loss to learn an energy surface. A byproduct is that the EBM learns a compressed representation of the data, which can be used for, e.g., denoising. Our work is similar in the aspect that we use a particular EBM architecture (a triplet network) and training the EBM with our proposed method results in a compressed representation of the data which is useful for downstream tasks, e.g., classification and transfer learning. In that sense, our work is also roughly related to Denoising Autoencoders (DAEs) [31] and Restricted Boltzmann Machines (RBMs) as well. Both are EBMs whose goal is to extract useful representations from data. However, we cannot say Ranzato et al. [2], RBMs, or DAEs precede our work in relating contrastive learning to EBMs, because they do not mention anything about contrasting positive pairs against negative pairs.

A.4 Sample Efficiency of Generative Models

We motivate the sample efficiency of EBCLR through the lens of discriminative and generative learning. We note that standard contrastive learning (e.g., SimCLR) optimizes a discriminative objective (the model must predict positive pairs) and EBCLR optimizes a generative objective (given a pair of images, the model must output how likely it is to be a positive pair). So, we can compare the performance of discriminative models and generative models.

There is a long line of works which show (theoretically and / or empirically) that generative models can outperform discriminative ones. For instance, Efron [32] has theoretically shown that Normal Discriminant Analysis can be more efficient than logistic regression. Ng et al. [33] has also theoretically shown that in the small sample regime, naïve Bayes can be more resistant to overfitting than logistic regression (i.e., find better solutions when given a small number of training samples).

There are also works [8, 9, 10] that consider optimizing a weighted combination of a discriminative loss and a generative loss, just like our EBCLR objective (11). They find that with an appropriate weight, the model can outperform discriminative models. In particular, Larochelle et al. [9] finds this hybrid approach to be beneficial in the limited training data setting. Grathwohl et al. [10] uses this hybrid approach to train a classifier and find that the classifier exhibits excellent calibration, unlike other calibration methods that require additional training data.

We can also address the efficiency of generative models from the perspective of generative pre-training. Bengio et al. [34] finds that pre-training a network with a generative loss brings the network weights towards a good local minimum, thus bringing better generalization. Erhan et al. [35] finds that generative pre-training acts as a regularizer and that this regularization effect can be beneficial when we have a small training set.

Another explanation specific to EBCLR is that the generative loss has an effect orthogonal to the effect of the discriminative loss. To minimize the generative loss, the model needs to place low energy on data and high energy elsewhere. This can be achieved only when the model has features that accurately capture the manifold of the data. When we have a small training set, the model may easily solve the discriminative objective and overfit. By providing an auxiliary generative task, the overfitting can be mitigated. An analogous effect with Joint Energy-Based Models is demonstrated by Zhao et al. [36].

B Psuedocodes

Algorithm 1 MSGLD

Input: Energy E_θ, initialization point $\tilde{v}$, count $\kappa_{\tilde{v}}$, step size α, number of iterations T, proximity parameter δ, decay parameter K, variance bounds $\sigma^2_{\min}$, $\sigma^2_{\max}$
$\sigma = \sigma_{\min} + (\sigma_{\max} - \sigma_{\min}) \cdot [1 - \kappa_{\tilde{v}}/K]_+$
for $t = 0, 1, 2, \ldots T-1$ **do**
    Sample $\epsilon \sim \mathcal{N}(0, \sigma^2)$
    $\tilde{v} \leftarrow \tilde{v} - \alpha \cdot \text{clamp}\{\nabla_{\tilde{v}} E_\theta(\tilde{v}), \delta\} + \epsilon$
end for
$\kappa_{\tilde{v}} \leftarrow \kappa_{\tilde{v}} + 1$

Algorithm 2 EBCLR

Input: DNN f_θ, batch size N, generative term weight λ, replay buffer $\mathcal{B}$, reinit. frequency ρ
while not converged **do**
    Sample $\{(v_n, v'_n)\}_{n=1}^N$ from $p(v, v')$
    Calculate the disc. term with Eq. (18)
    Calculate gradient of the disc. term
    Sample $\{\tilde{v}_n\}_{n=1}^N$ from $\mathcal{B}$ with prob. $1-\rho$ and from $p(v)$ with prob. ρ
    Calculate $E_\theta(v; \{v'_m\}_{m=1}^N)$ with Eq. (20)
    Update $\{\tilde{v}_n^{(2)}\}_{n=1}^N$ with MSGLD in Algo. 1
    Calculate gradient of the gen. term with Eq. (3)
    Update θ via gradient ascent
    Update $\mathcal{B}$ with MSGLD samples $\{\tilde{v}_n\}_{n=1}^N$
end while

C Missing Proofs

C.1 Proof of Theorem 2

Proof. We first observe that

$$\begin{aligned}\frac{1}{Z(\theta)}\exp\{-E_\theta(v)\} &= \frac{1}{Z(\theta)}\int e^{-\|z-z'\|^2/\tau}\,dv' \\ &= \int \frac{1}{Z(\theta)} e^{-\|z-z'\|^2/\tau}\,dv' \\ &= \int q_\theta(v,v')\,dv' \\ &= q_\theta(v)\end{aligned}$$

which proves the relation (12). We also have

$$\log q_\theta(v) = -\log Z(\theta) - E_\theta(v)$$

and so

$$\begin{aligned}\nabla_\theta \log q_\theta(v) &= -\frac{1}{Z(\theta)}\nabla_\theta Z(\theta) - \nabla_\theta E_\theta(v) \\ &= -\frac{1}{Z(\theta)}\nabla_\theta \int \exp\{-E_\theta(v)\}\,dv - \nabla_\theta E_\theta(v) \\ &= -\frac{1}{Z(\theta)}\int \nabla_\theta \exp\{-E_\theta(v)\}\,dv - \nabla_\theta E_\theta(v) \\ &= -\frac{1}{Z(\theta)}\int \{-\nabla_\theta E_\theta(v)\}\cdot\exp\{-E_\theta(v)\}\,dv - \nabla_\theta E_\theta(v) \\ &= \int \{\nabla_\theta E_\theta(v)\}\cdot\frac{1}{Z(\theta)}\exp\{-E_\theta(v)\}\,dv - \nabla_\theta E_\theta(v) \\ &= \int \{\nabla_\theta E_\theta(v)\}\cdot q_\theta(v)\,dv - \nabla_\theta E_\theta(v) \\ &= \mathbb{E}_{q_\theta}[\nabla_\theta E_\theta(v)] - \nabla_\theta E_\theta(v).\end{aligned}$$

Taking expectation w.r.t. $p(v)$ establishes the relation (13). □

C.2 Another Theoretical Justification for EBCLR

Here we provide another theoretical justification on how solving the EBCLR objective (11) with $\lambda = 1$ causes $q_\theta(v, v')$ to approximate $p(v, v')$. It is known that optimizing the first term of (11) with (18) in place of $q_\theta(v' \mid v)$ will cause (18) to approximate $p(v' \mid v)$ [37]. Also, optimizing the second term of (11) with (15) or (17) in place of $q_\theta(v)$ will cause (15) or (17) to be proportional to $p(v)$ (indeed, in Table 10, we see SGLD samples of the EBCLR marginal $q_\theta(v)$ approximated by (17) achieve a non-trivial FID score). So, the product of (15) or (17) with (18) will be proportional to $p(v, v')$. Moreover, by construction, the product of (15) or (17) with (18) is (approximately) $q_\theta(v, v')$. Hence, optimizing the EBCLR objective (11) will cause $q_\theta(v, v')$ to model $p(v, v')$.

D Complete Training and Evaluation Details

Baseline methods and datasets. The baseline methods are SimCLR, MoCo v2, SimSiam, and BYOL. The hyper-parameters are chosen closely following the original works [4, 12, 7, 6]. We use four datasets: MNIST [23], Fashion MNIST (FMNIST) [24], CIFAR10, and CIFAR100 [25].

Data transformation. For fair comparison, we use the stochastic data transformation proposed by Chen et. al [4] for all methods. In the case of grayscale images, we only use the random cropping augmentation. For EBCLR, we also add small Gaussian noise $\mathcal{N}(0, 0.03^2)$ to stabilize training [16, 10, 15]. We remark that Gaussian noise of standard deviation 0.03 is nearly invisible to the human eye.

DNN architecture. We decompose $f_\theta = \pi_\theta \circ \phi_\theta$ where ϕ_θ is the encoder network and π_θ is the projection network. Rather than directly using the output of f_θ for downstream tasks, we follow previous works [4, 5, 1, 2, 3, 6, 7] and use the output of ϕ_θ instead.

In our experiments, we set ϕ_θ to be a ResNet-18 [26] up to the global average pooling layer. The architecture of π_θ and π_θ to be a 2-layer MLP with output dimension 128. However, we remove batch normalization because batch normalization hurts SGLD [16]. We also replace ReLU activations with leaky ReLU to expedite the convergence of SGLD. For the baseline methods, we use the settings proposed in the original works while keeping the backbone fixed to be ResNet-18.

SGLD settings. The default parameters for MSGLD are $|\mathcal{B}| = 50k$, $\rho = 0.2$, $T = 10$, $\alpha = 0.05$, $\delta = 1.0$, $K = 3$, $\sigma_{\min} = 0.01$, and $\sigma_{\max} = 0.05$. We use $\rho = 0.4$ for MNIST, $\rho = 0.6$ for FMNIST, and $\rho = 0.2$ for CIFAR10 and CIFAR100. At the beginning of training, we fill the buffer $\mathcal{B}$ entirely with proposal distribution samples.

Training. The temperature is $\tau = 0.1$ for EBCLR. For EBCLR, as we cannot use batch normalization, it is difficult to apply SGD with momentum. Hence, we use the Adam optimizer [38]. The learning rate is 0.0002 if the batch size is 128 and 0.0001 if the batch size is smaller than 128. We also weakly regularize the squared ℓ_2-norm of the outputs of f_θ with weight 0.001. For the baseline methods, we closely follow the settings of previous works [4, 12, 27]. Specifically, we use SGD with momentum 0.9 and learning rate of $0.03 \cdot (\textit{batch size}/256)$ and weight decay 0.0001. Each model is trained with a single GeForce RTX 3090.

Evaluation. We assess the quality of the representations by training a linear classifier on top of frozen ϕ_θ. The linear classifier is trained with Adam for 200 epochs with batch size 512. The learning rate η is found by grid search for $\log_{10} \eta$ in $[-4, -1]$.

E Additional Experiments

E.1 Section 4.1 with Longer Training

We repeat the experiments in Section 4.1 by training the models for 200 epochs. For EBCLR, the number of SGLD iterations T is increased by 5 when SGLD is unable to generate samples with sufficiently high energy. Tables 4 and 5 describe the results. We again note that EBCLR consistently beats all baseline methods and demonstrates high efficiency.

Dataset	**MNIST**		**FMNIST**		**CIFAR10**		**CIFAR100**	
Statistic	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.
SimSiam	98.0	0.025	87.3	0.05	72.8	0.175	42.9	0.125
BYOL	**99.4**	0.5	89.5	0.25	75.3	0.325	45.7	0.2
SimCLR	99.1	0.05	88.9	0.05	72.7	0.175	41.9	0.05
MoCo v2	98.9	0.05	89.4	0.25	67.4	0.075	45.0	0.2
EBCLR	**99.4**	–	**90.8**	–	**80.0**	–	**51.6**	–

Table 4: Linear evaluation accuracy and efficiency relative to EBCLR. Efficiency of a method relative to EBCLR is calculated by the following formula: (number of epochs used by EBCLR to reach the final accuracy of the method) / (total number of training epochs).

Direction	**M** $\to$ **FM**	**FM** $\to$ **M**	**C10** $\to$ **C100**	**C100** $\to$ **C10**
SimSiam	86.5	95.2	43.2	67.0
BYOL	87.1	97.3	47.7	73.6
SimCLR	86.9	97.2	45.7	65.4
MoCo v2	86.2	97.6	41.2	68.7
EBCLR	**87.3**	**98.5**	**48.4**	**74.2**

Table 5: Comparison of transfer learning results in the linear evaluation setting. Left side of the arrow is the dataset than the encoder was pre-trained on, and right side of the arrow is the dataset that linear evaluation was performed on. We use the following abbreviations. **M** : MNIST, **FM** : FMNIST, **C10** : CIFAR10, **C100** : CIFAR100.

E.2 Section 4.2 with Longer Training

We repeat the experiments in Section 4.2 by training the models for 200 epochs. For EBCLR, the number of SGLD iterations T is increased by 5 when SGLD is unable to generate samples with sufficiently high energy. Table 6 describes the results. We observe similar patterns as in the case of training for 100 epochs. In the case of CIFAR10, SimCLR seems to have a pattern agnostic to the batch size. However, we note that there is a significant gap between the performance of SimCLR with batch size 256 (c.f. Table 4) and batch sizes in $\{16, 64, 128\}$.

Interestingly, for SimCLR with small batch sizes, we observe that training longer with small batch sizes leads to worse performance than training only for 100 epochs. This is particularly prominent on CIFAR10 and CIFAR100 (compare Tables 3 and 6). We speculate that this is because, with small batches, SimCLR is unable to provide hard negatives, and this leads to overfitting of the DNN. On the other hand, EBCLR does not suffer from this phenomenon. This again demonstrates the importance of the generative term.

Dataset	**MNIST**			**FMNIST**			**CIFAR10**			**CIFAR100**		
Batch Size	16	64	128	16	64	128	16	64	128	16	64	128
SimCLR	98.7	98.7	99.0	86.9	88.5	88.9	65.3	64.0	63.8	33.4	39.6	43.0
EBCLR	**99.4**	**99.4**	**99.4**	**89.8**	**90.2**	**90.2**	**79.2**	**80.2**	**80.0**	**51.0**	**52.4**	**51.6**
Rel. Eff.	0.3	0.05	0.45	0.05	0.1	0.1	0.05	0.05	0.05	0.025	0.1	0.15

Table 6: Linear evaluation accuracies and efficiencies relative to EBCLR with various batch sizes. Efficiency of SimCLR relative to EBCLR is calculated by the following formula: (number of epochs used by EBCLR with the same batch size to reach the final accuracy of SimCLR) / (total number of training epochs).

E.3 Comparison of Approximations

We can either use Equation (15) or Equation (17) when approximating the generative term. Figure 5 shows results for both approximations on CIFAR10. We observe identical performance for both approximations, which implies that the generative term is robust to the number of samples used in the calculation of the empirical mean. This justifies our choice of using the simpler Equation (17) to approximate the marginal distribution $q(v)$.

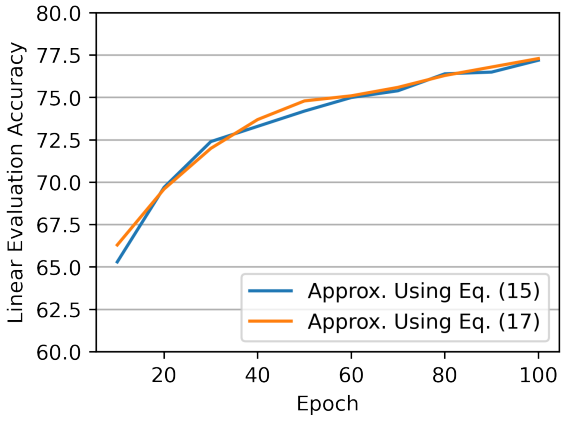


Figure 5: Comparison of different approximations of the generative term, demonstrated on CIFAR10.

E.4 Evaluation with KNN Classifiers

We also reproduce Tables 1, 2, and 3 using K-nearest neighbour (KNN) classifiers. We followed the evaluation protocol of Caron et al. [39] and used a weighted KNN classifier with $K = 20$. The results are given below. In Table 7, except for the case of BYOL on MNIST, we see that EBCLR beats all baselines again by a non-trivial margin. EBCLR is also shown to be sample efficient. In Table 8, we see EBCLR outperforms all baselines on the task of transfer learning. In Table 9, we see EBCLR is robust to small batch sizes. This leads to the lower efficiency of SimCLR compared to EBCLR. On CIFAR10 and CIFAR100, relative efficiency does not decrease below 0.05 since we saved checkpoints every 5 epochs. We can reasonably expect SimCLR to have lower relative efficiency at smaller batch sizes if we had saved checkpoints every epoch.

Dataset	**MNIST**		**FMNIST**		**CIFAR10**		**CIFAR100**	
Statistic	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.	Accuracy	Rel. Eff.
SimSiam	94.5	0.05	80.4	0.01	59.6	0.1	23.9	0.05
BYOL	**98.8**	–	84.4	0.2	62.3	0.15	31.3	0.1
SimCLR	97.4	0.15	84.1	0.15	57.2	0.05	29.5	0.1
MoCo v2	94.5	0.05	83.1	0.1	54.1	0.05	26.0	0.05
EBCLR	98.1	–	**86.6**	–	**71.4**	–	**39.4**	–

Table 7: KNN accuracy and efficiency relative to EBCLR. Efficiency of a method relative to EBCLR is calculated by the following formula: (number of epochs used by EBCLR to reach the final accuracy of the method) / (total number of training epochs).

Direction	**M** $\rightarrow$ **FM**	**FM** $\rightarrow$ **M**	**C10** $\rightarrow$ **C100**	**C100** $\rightarrow$ **C10**
SimSiam	80.3	85.1	26.1	52.6
BYOL	82.6	92.4	29.3	63.7
SimCLR	80.7	87.6	25.4	56.0
MoCo v2	78.7	89.7	23.8	52.2
EBCLR	**83.4**	**95.5**	**35.7**	**65.4**

Table 8: Comparison of transfer learning results in the KNN evaluation setting. Left side of the arrow is the dataset than the encoder was pre-trained on, and right side of the arrow is the dataset that KNN evaluation was performed on. We use the following abbreviations. **M** : MNIST, **FM** : FMNIST, **C10** : CIFAR10, **C100** : CIFAR100.

Dataset	**MNIST**			**FMNIST**			**CIFAR10**			**CIFAR100**		
Batch Size	16	64	128	16	64	128	16	64	128	16	64	128
SimCLR	97.1	97.7	97.4	82.3	83.2	83.3	52.8	56.0	56.8	22.1	22.7	28.8
EBCLR	**98.5**	**98.3**	**98.1**	**85.0**	**85.9**	**86.6**	**72.2**	**72.6**	**71.4**	**40.6**	**41.4**	**39.4**
Rel. Eff.	0.05	0.05	0.15	0.05	0.15	0.1	0.05	0.05	0.05	0.05	0.05	0.05

Table 9: KNN accuracies and efficiencies relative to EBCLR with various batch sizes. Efficiency of SimCLR relative to EBCLR is calculated by the following formula: (number of epochs used by EBCLR with the same batch size to reach the final accuracy of SimCLR) / (total number of training epochs).

E.5 Generative Performance of EBCLR

In this section, we explore the effect of the discriminative loss on the generative performance of EBCLR. To this end, we solved

$$\max_{\theta} \gamma \mathbb{E}_p[\log q_\theta(v' \mid v)] + \mathbb{E}_p[\log q_\theta(v)] \tag{23}$$

for various values of γ on FMNIST. Recall that in the original EBCLR objective, weight λ comes in front of the generative term. We then measured the Fréchet Inception Distance (FID, lower is better) between samples from the true marginal $p(v)$ and SGLD samples from the EBM marginal $q_\theta(v)$ approximated by (19) / (17). The results are shown in Table 10.

Interestingly, we found using an appropriate $\gamma > 0$ led to improvements in FID over $\gamma = 0$. In other words, contrastive learning did help improve the generative performance of the model. However, using an excessively large γ led to a deterioration of the performance. This is analogous to the result of Section 4.3 where we observed using an appropriate weight on the generative term led to improvements over using only the discriminative term (contrastive loss). The difference is that to achieve optimal generative performance, the generative term weight needs to be larger (i.e., $\gamma < 1$) whereas to achieve optimal discriminative performance, the discriminative term weight needs to be larger (i.e., $\lambda < 1$ in (11)). We speculate that using a larger model could mitigate this trade-off, but this topic is beyond the scope of this work.

γ	0	0.01	0.1	1	10
FID ↓	40.78	9.68	7.68	17.03	139.19

Table 10: Generative performance of EBCLR for various values of γ.

F A Discussion on Making EBCLR Scalable

We could consider three alternative losses / methods for training EBCLR: noise contrastive estimation (NCE) [40], VERA [41], and score matching [42]. Although the NCE loss does not depend on MCMC, it requires specification of a noise distribution whose normalized density can be easily evaluated. In most cases, such noise distribution samples are easily distinguishable from natural image samples and lead to the density chasm problem [43], making optimization difficult. Telescoping density-ratio estimation [43] attempts to mitigate this problem, but it does not seem to scale well to large data. Flow contrastive estimation [44] uses a flow model to define the noise distribution and trains a flow model along with the EBM. Though flow contrastive estimation requires an auxiliary flow model, we believe this is the most promising alternative to MCMC. In a similar sense, VERA, which uses a generator to sample from the EBM, seems promising as well. Yet, flow contrastive estimation and VERA also did not demonstrate scalability to large-scale datasets such as ImageNet. Score matching matches the gradient of model log-density to the gradient of data log-density, so it does not rely on MCMC. However, it requires calculation of the Hessian, so in the deep learning setting, this approach is also quite expensive.