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 and reduce the similarity between representations of negative pairs. Unfortunately, contrastive methods usually require large datasets with significant number of negative pairs per iteration to achieve reasonable performance on downstream tasks. To address this problem, here we explore Energy-Based Contrastive Learning (EBCLR) that combines contrastive learning with Energy-Based Models (EBMs) and can be theoretically interpreted as learning the joint distribution of positive pairs. EBCLR shows promising results on small and medium-scale datasets such as MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. Specifically, 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, 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 number of negative pairs.

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.

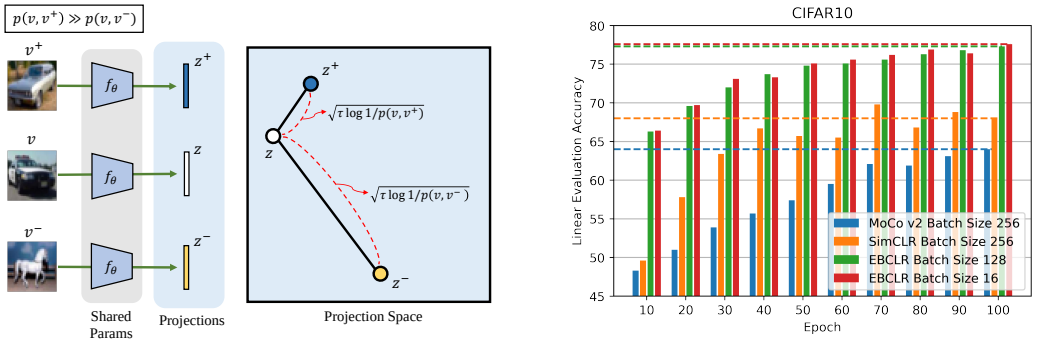


Figure 1: **Left:** An illustration of EBCLR. We use $p(v, v')$, the joint distribution of positive pairs, as a measure of semantic similarity of images. A DNN f_θ is trained such that the squared distance between its outputs projected onto the unit sphere approximates $\tau \log 1/p(v, v')$ up to a constant. **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 approach this issue by proposing Energy-Based Contrastive Learning (EBCLR) which combines 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 its reliance on Stochastic Gradient Langevin Dynamics (SGLD) [8], 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 number of negative pairs, and it outperforms SimCLR and MoCo v2 [9] in terms of sample efficiency and linear evaluation accuracy.

Our contributions can be summarized as follows:

- A novel contrastive learning method called EBCLR is proposed by learning the joint distribution of positive pairs. We show that EBCLR loss is equivalent to a combination of a contrastive term and a generative term. To the best of our knowledge, this is the first work to apply EBMs to contrastive learning of visual representations.
- To accelerate the training of EBCLR, we show that it is necessary to use an appropriate variance schedule in SGLD.
- By exploring the sensitivity of EBCLR to the generative term, we propose a further improvement by appropriately weighing the generative term.
- EBCLR is several times more sample-efficient than contrastive and non-contrastive methods. This leads to a large performance advantage for EBCLR given the same number of training epochs.
- Unlike SimCLR, EBCLR is shown to be robust to small number of negative pairs. Empirically, we observe little difference between linear evaluation accuracies for EBCLR with 254 negative pairs (batch size 128) and 30 negative pairs (batch size 16) per positive pair.

2 Related Works

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 [9] 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) [10] 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 learning generative classifiers [11, 12], generating images [13], and training regression models [14, 15]. However, to the best of our knowledge, this paper is the first work to combine EBMs with contrastive 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-Leiber (KL) divergence between q_θ and p that maximizes the expected log-likelihood of q_θ under p [16]:

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

Stochastic gradient ascent can be used to solve (2) [16]. 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.

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 [13, 17, 11, 12] have used Stochastic Gradient Langevin Dynamics (SGLD) [8] to generate samples from q_θ. Specifically, given a sample v_0 from some proposal distribution q_0, the iteration

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

guarantees that the sequence $\{v_t\}$ converges to a sample from q_θ assuming $\{\alpha_t\}$ decays at a polynomial rate [8].

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 [13, 17, 11, 12]. Moreover, Yang and Ji [12] 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.

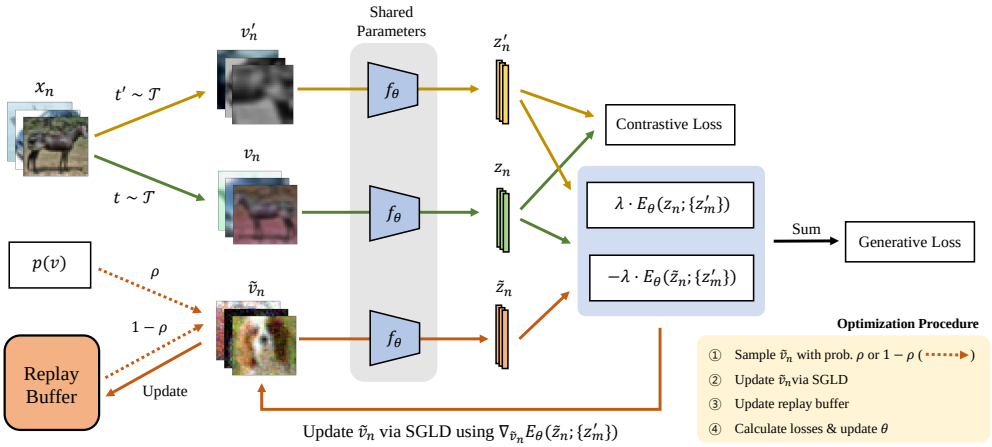


Figure 2: An illustration of the learning process of EBCLR.

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, τ 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.

Our key idea is then to use $p(v, v')$ as a measure of semantic similarity of v and v'. Specifically, $p(v, v')$ will be high when v and v' are semantically similar and low otherwise. Thus, if q_θ successfully approximates p, by the definition of q_θ in (6), the distance between z and z' will be controlled by the inverse of semantic similarity of v and v'. 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. Similar decomposition was used by Grathwohl et. al [11] 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 **??** in the Supplementary Material.

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 immediate 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 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 use (15) as for the empirical mean, but either choice showed identical performance (c.f. Appendix D.3). So, we have found (15) to be not worth the additional complexity, and have resorted to the simpler approximation (17) instead.

If we compare (19) with (12), we can see that this approximation of $q_\theta(v)$ yields the energy function (after ignoring the irrelevant constant $\log N$)

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

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 follows:

$$\tilde{v} \leftarrow \tilde{v} - \alpha \cdot \text{clamp}\{\nabla_{\tilde{v}} E_\theta(\tilde{v}; \{v'_m\}_{m=1}^N), \delta\} + \epsilon \tag{21}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and the update is repeated T times. 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 [13, 11, 12].

Second, the proposal distribution q_0 is set to be the data distribution $p(v)$. This choice differs from those of previous works [13, 11, 12] 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 $[\cdot]_+ \coloneqq \max\{0, \cdot\}$, $\sigma^2_{\max}$ 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 given in Algorithms 1 and 2, respectively, in the Supplementary Material, and the overall learning flow of EBCLR is described in Figure 2.

4 Experiments

We now describe the experiment settings. A complete description is deferred to Appendix C.

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, 9, 7, 6]. We use four datasets: MNIST [18], Fashion MNIST (FMNIST) [19], CIFAR10, and CIFAR100 [20].

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 [21] 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 [13]. We also replace ReLU activations with leaky ReLU to expedite the convergence of SGLD. For the baseline methods, we use settings proposed in the original works while keeping the backbone fixed to be ResNet-18.

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).

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 $[-4, -1]$.

4.1 Comparison with Baselines

We use batch size 128 for EBCLR and batch size 256 for the baseline methods following Wang et. al [22] 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** $\to$ **FM**	**FM** $\to$ **M**	**C10** $\to$ **C100**	**C100** $\to$ **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 led to similar conclusions (c.f. Supplementary Material Section D.1).

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

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

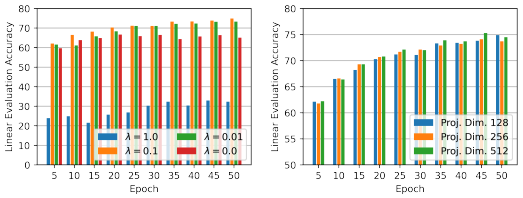
Table 3: Linear evaluation accuracies and efficiencies relative to EBCLR with various batch sizes. Efficiency of SimCLR relative to EBCLR is calculated by the following formula: (number of epochs used by EBCLR with the same batch size to reach the final accuracy of SimCLR) / (total number of training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

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 have a small number of GPUs. Repeating the experiments with longer training again led to similar conclusions (c.f. Supplementary Material Section D.2).

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.



(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 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.

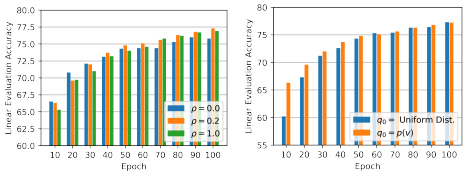
Initially, $\rho = 0.0$ shows the best performance, as SGLD quickly reaches samples of lower energy. However, learning then slows down because of the lack of diversity of samples in the replay buffer $\mathcal{B}$. This implies it is necessary to set $\rho > 0$ in order to learn good representations.

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.

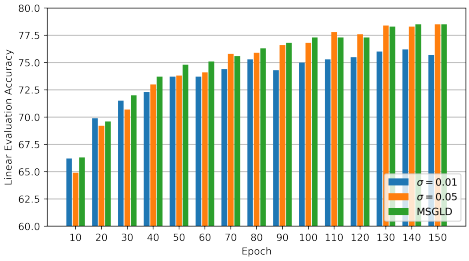
Given the above observations, it is clear why the intermediate value 0.2 is the best choice out of $\rho \in \{0.0, 0.2, 1.0\}$. $\rho = 0.2$ allows enough time for samples in the replay buffer to reach low energy while still maintaining diversity of samples in $\mathcal{B}$. Also, it is sample-efficient 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 high efficiency of EBCLR compared to the baseline methods in Tables 1 and 3.

We believe this acceleration effect can be explained through the work of Hinton [23]. 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 [23].



(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.

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.

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 are limitations of EBMs in general. Given the increasing efforts to overcome these limitations such as Proximal-YOPO-SGLD, 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 reduction of resource consumption.

6 Conclusion

In this work, we proposed EBCLR which combines contrastive learning with EBMs. This amalgamation of ideas have led to both theoretical and practical contributions. Theoretically, EBCLR associates distance on the projection space with the density of positive samples. Since the distribution of positive samples reflects 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 robust to small number of negative pairs. Hence, EBCLR is applicable even in the scenario 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] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

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

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

[11] 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.

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

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

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

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

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

[17] 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.

[18] 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.

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

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

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

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

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

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

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 B.
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 C.
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]