EMC^2: Efficient MCMC Negative Sampling for Contrastive Learning with Global Convergence

Chung-Yiu Yau [1] **Hoi-To Wai** [1] **Parameswaran Raman** [2] **Soumajyoti Sarkar** [2] **Mingyi Hong** [3]

Abstract

A key challenge in contrastive learning is to generate negative samples from a large sample set to contrast with positive samples, for learning better encoding of the data. These negative samples often follow a softmax distribution which are dynamically updated during the training process. However, sampling from this distribution is non-trivial due to the high computational costs in computing the partition function. In this paper, we propose an Efficient Markov Chain Monte Carlo negative sampling method for Contrastive learning (EMC^2). We follow the global contrastive learning loss as introduced in (Yuan et al., 2022), and propose EMC^2 which utilizes an adaptive Metropolis-Hastings subroutine to generate hardness-aware negative samples in an online fashion during the optimization. We prove that EMC^2 finds an $\mathcal{O}(1/\sqrt{T})$-stationary point of the global contrastive loss in T iterations. Compared to prior works, EMC^2 is the first algorithm that exhibits global convergence (to stationarity) regardless of the choice of batch size while exhibiting low computation and memory cost. Numerical experiments validate that EMC^2 is effective with small batch training and achieves comparable or better performance than baseline algorithms. We report the results for pre-training image encoders on `STL-10` and `Imagenet-100`.

[1]Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong SAR of China. The work of C.-Y. Yau was done while interning at Amazon Web Services. [2]Amazon Web Services, USA. [3]Department of Electrical and Computer Engineering, University of Minnesota, USA. M. Hong holds concurrent appointments as an Amazon Scholar and as a faculty at University of Minnesota. This paper describes his work performed at Amazon. Correspondence to: Chung-Yiu Yau <cyyau@se.cuhk.edu.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Contrastive representation learning has been instrumental in self-supervised learning for large-scale pre-training of foundation models (Radford et al., 2021; Cherti et al., 2023) as well as in the fine-tuning stage on downstream tasks (Xiong et al., 2020; Lindgren et al., 2021). It helps encode real-world data into low-dimensional feature vectors that abstract the important attributes about the data, and generalize well outside of the training distribution. More recently, contrastive learning with multi-modal data has helped embed different data modalities into the same feature space (Li et al., 2023), such as the studies with visual-language models (Radford et al., 2021; Alayrac et al., 2022; Cherti et al., 2023) and document understanding (Xu et al., 2020; Lee et al., 2023).

Contrastive learning uses pairwise comparison of representations in the training objective, with the goal of learning representations of data where positive pairs are drawn closer while negative pairs move apart in the representation space. It is well known that generating a large dataset of pairwise samples such as image-text pairs of the same semantics costs much lower than manual labeling, e.g., the WebImageText dataset used for training CLIP originates from Wikipedia articles (Radford et al., 2021). While there have been many studies about training objectives for contrastive learning, optimizing the contrastive loss efficiently remains an open-problem as current optimization methods critically rely on a large batch size to maintain the quality of the negative sample distribution.

Formally, we aim to train models $\phi : \mathcal{X} \to \mathbb{R}^d$ and $\psi : \mathcal{Y} \to \mathbb{R}^d$ that are parameterized by $\theta \in \mathbb{R}^p$. These models encode data from the input space $\mathcal{X}, \mathcal{Y}$ of potentially two modalities into a vector space. Given $\mathbf{D}_X \subseteq \mathcal{X}$, $\mathbf{D}_Y \subseteq \mathcal{Y}$, we define a distribution of positive pairs $\mathcal{D}_{\text{pos}}$ such that $\text{supp}(\mathcal{D}_{\text{pos}}) \subseteq \mathbf{D}_X \times \mathbf{D}_Y$ and a negative set $\mathbf{D}_{\text{neg}}(x) \subseteq \mathbf{D}_Y$ for every data point $x \in \mathbf{D}_X$. We denote the dataset constructed in this manner by the tuple $(\mathcal{D}_{\text{pos}}, \mathbf{D}_{\text{neg}}(\cdot))$.

With the dataset $(\mathcal{D}_{\text{pos}}, \mathbf{D}_{\text{neg}}(\cdot))$, let $\beta > 0$ be the inverse temperature, this paper aims to minimize the following global contrastive loss proposed in (Lindgren et al., 2021;

Yuan et al., 2022):

$$\min_{\theta\in\mathbb{R}^p} \mathcal{L}(\theta) \tag{1}$$
$$:= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\rm pos}} \left[-\log \frac{\exp(\beta\phi(x;\theta)^\top\psi(y;\theta))}{\sum_{z\in\mathbf{D}_{\rm neg}(x)} \exp(\beta\phi(x;\theta)^\top\psi(z;\theta))} \right]$$

also see (Mikolov et al., 2013) that proposed a similar loss to (1). Assume $m := |\mathrm{supp}(\mathcal{D}_X)| < \infty$ where $\mathcal{D}_X$ is the marginal distribution of x on $\mathcal{D}_{\rm pos}$. Moreover, we let $m_{\rm neg} := |\mathbf{D}_{\rm neg}(x)| < \infty$ for all $x \in \mathbf{D}_X$[1].

The global contrastive loss (1) finds the feature encoders $\phi^\star, \psi^\star$ that map data into $\mathbb{R}^d$ with maximized similarity $\phi^\star(x)^\top\psi^\star(y)$ between positive data pair $(x, y) \sim \mathcal{D}_{\rm pos}$ and minimized similarity $\phi^\star(x)^\top\psi^\star(z)$ between negative data pair (x, z), $z \in \mathbf{D}_{\rm neg}(x)$. In other words, the loss maximizes the probability of observing a positive sample y under the context x among the other negative samples.

We notice that a highly related loss function design to (1) is the InfoNCE loss (Logeswaran & Lee, 2018; Oord et al., 2018; Chen et al., 2020), which is defined by:

$$\mathcal{L}_{\rm NCE}(\theta; B) \tag{2}$$
$$= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\rm pos}} \mathop{\mathbb{E}}_{\mathbf{Z}\sim\mathcal{D}_{\rm neg}(x;B)} \left[-\log \frac{\exp(\beta\phi(x;\theta)^\top\psi(y;\theta))}{\sum_{z\in\mathbf{Z}} \exp(\beta\phi(x;\theta)^\top\psi(z;\theta))} \right]$$

such that $\mathcal{D}_{\rm neg}(x; B)$ is a distribution of all subsets $\mathbf{Z} \subseteq \mathbf{D}_{\rm neg}(x)$ with $|\mathbf{Z}| = B$. In fact, the global contrastive loss (1) is an upper bound of the InfoNCE loss (2) since

$$\mathop{\mathbb{E}}_{\mathbf{Z}\sim\mathcal{D}_{\rm neg}(x;B)} \left[\log \sum_{z\in\mathbf{Z}} \exp(\beta\,\phi(x;\theta)^\top\psi(z;\theta)) \right] \tag{3}$$
$$\le \log \sum_{z\in\mathbf{D}_{\rm neg}(x)} \exp(\beta\,\phi(x;\theta)^\top\psi(z;\theta)),$$

for any $x \in \mathrm{supp}(\mathcal{D}_{\rm pos}), \theta \in \mathbb{R}^p$. As the equality holds when $B = m_{\rm neg}$, we have $\mathcal{L}_{\rm NCE}(\theta; m_{\rm neg}) = \mathcal{L}(\theta)$.

It was demonstrated that minimizing (2) with $B \gg 1$ produces encodings with interpretable geometric and semantic properties (Wang & Isola, 2020; Robinson et al., 2020; Zimmermann et al., 2021). At the same time, the convergence of pre-training foundation models on large scale dataset happens only when using a large enough batch size B (Radford et al., 2021). Such evidence points towards the success of models that are trained to minimize $\mathcal{L}_{\rm NCE}(\theta; B)$ of large negative batch size B. We believe that minimizing $\mathcal{L}(\theta)$, which yields the limiting upper bound to (2), can lead to a better performance for contrastive learning.

[1]The results of this paper can be extended to the case where the negative sample size $|\mathbf{D}_{\rm neg}(x)|$ is uneven.

1.1. Challenges in Optimizing (1)

We notice that the contrastive loss gradient is given by:

$$\nabla\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\rm pos}} \Big[-\beta\,\nabla_\theta(\phi(x;\theta)^\top\psi(y;\theta)) \Big] \tag{4}$$
$$+ \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\rm pos}} \Big[\beta \sum_{z\in\mathbf{D}_{\rm neg}(x)} p_{x,\theta}(z)\nabla_\theta(\phi(x;\theta)^\top\psi(z;\theta)) \Big]$$
$$\equiv \nabla\mathcal{L}_{\rm pos}(\theta) + \nabla\mathcal{L}_{\rm neg}(\theta),$$

with the softmax distribution:

$$p_{x,\theta}(z) = \frac{\exp(\beta\,\phi(x;\theta)^\top\psi(z;\theta))}{\sum_{z'\in\mathbf{D}_{\rm neg}(x)} \exp(\beta\,\phi(x;\theta)^\top\psi(z';\theta))}. \tag{5}$$

The challenge of optimizing $\mathcal{L}(\theta)$ lies in the overwhelming complexity to compute or approximate $\nabla\mathcal{L}_{\rm neg}(\theta)$ since $\mathbf{D}_{\rm neg}(x)$ often spans a large dataset, e.g., $m_{\rm neg} = 8.8 \times 10^6$ for the `MS MARCO` dataset (Bajaj et al., 2016).

To this end, SimCLR (Chen et al., 2020) proposed to replace $\mathbf{D}_{\rm neg}(x)$ in (4) by a randomly selected *negative batch* $\mathbf{Z}$ of B randomly augmented images, i.e., using the gradient of InfoNCE loss (2). While achieving reasonable performance in certain scenarios, SimCLR requires a large batch size B on large-scale dataset training. For example, it requires up to $B = 32768$ negative samples per iteration in training CLIP (Radford et al., 2021), making it impossible to train such models in non-commercial data center (Cherti et al., 2023). We note that there exists a number of contrastive learning tricks to avoid the problem of estimating $\nabla\mathcal{L}_{\rm neg}(\theta)$. For example, (Robinson et al., 2020) imposes a hardness-aware distribution on $\mathcal{D}_{\rm neg}(x; B)$ of the InfoNCE loss. (He et al., 2020) utilizes a momentum mechanism to prevent feature collapse. (Grill et al., 2020; Zbontar et al., 2021; Bardes et al., 2021) use alternative loss functions for contrastive learning, such as redundancy reduction loss and covariance regularization. See (Balestriero et al., 2023) for a comprehensive overview on different contrastive learning methods.

An alternative approach is to consider the estimation of $\nabla\mathcal{L}_{\rm neg}(\theta)$ through *negative sampling*. This approach is motivated from the observation that (5) is a probability mass function of a softmax distribution and thus the summation $\sum_{z\in\mathbf{D}_{\rm neg}(x)} p_{x,\theta}(z)\nabla_\theta(\phi(x;\theta)^\top\psi(z;\theta))$ in $\nabla\mathcal{L}_{\rm neg}(\theta)$ is equivalent to an expectation taken w.r.t. the softmax distribution. Subsequently, it can be approximated through *sampling*. In particular, if one draws $(x, y) \sim \mathcal{D}_{\rm pos}$, $z \sim \mathcal{P}_{x,\theta}$ where

$$\mathcal{P}_{x,\theta} \equiv (p_{x,\theta}(z))_{z\in\mathbf{D}_{\rm neg}(x)}, \tag{6}$$

then $\beta\nabla_\theta(\phi(x;\theta)^\top\psi(z;\theta))$ is an unbiased estimate of $\nabla\mathcal{L}_{\rm neg}(\theta)$.

However, sampling from $\mathcal{P}_{x,\theta}$ remains a highly non-trivial task due to the complexity in computing the partition function (i.e., denominator) in the softmax distribution (5). Prior

Algorithms	Convergence Error	Memory	Computation
SimCLR (Chen et al., 2020)	$\mathcal{O}\left(\frac{\beta^2}{\sqrt{T}} + \frac{m_{\text{neg}}^2 \exp(4\beta)\sigma^2}{B}\right)^\star$	$\mathcal{O}(BM_\phi + Bd)$	$\mathcal{O}(BC_\phi + B^2 d)$
Negative Cache (Lindgren et al., 2021)	$\mathcal{O}\left(\frac{\beta}{\sqrt{T}} + \frac{\beta^6}{\rho^2 T}\right)^\S$	$\mathcal{O}(M_\phi + m_{\text{neg}} d + d)$	$\mathcal{O}(C_\phi + m_{\text{neg}} d + \rho m C_\phi)$
SogCLR (Yuan et al., 2022)	$\mathcal{O}\left(\frac{\exp(6\beta)}{\sqrt{BT}} + \frac{m_{\text{neg}}^2 \exp(4\beta)\sqrt{m}\sigma^2}{B\sqrt{T}} + m_{\text{neg}}^2 \exp(4\beta) \upsilon^2\right)^\dagger$	$\mathcal{O}(BM_\phi + m + Bd)$	$\mathcal{O}(BC_\phi + B^2 d)$
EMC^2 (Ours, $R = 2$)	$\mathcal{O}\left(\frac{\beta}{\sqrt{T}} + \frac{m^2 m_{\text{neg}}^2 \exp(6c^2\beta)\beta^3\sigma^2}{B\sqrt{T}}\right)$	$\mathcal{O}(BM_\phi + m + Bd)$	$\mathcal{O}(BC_\phi + B^2 d)$
EMC^2 (Ours)	$\mathcal{O}\left(\frac{\beta}{\sqrt{T}} + \frac{2^R m^2 m_{\text{neg}}^2 \exp(6c^2\beta)\beta^3\sigma^2}{BR^2\sqrt{T}}\right)$	$\mathcal{O}(BM_\phi + m + Bd)$	$\mathcal{O}(BC_\phi + B^2 d + BR)$

Table 1. The second column shows the upper bound on convergence error $T^{-1}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla_\theta \mathcal{L}(\theta_t)\|_2^2\right]$. The last two columns show the memory/computation requirement per iteration of the algorithms. C_ϕ (resp. M_ϕ) denotes the computational (resp. memory) cost to compute the feature vector $\phi(x;\theta)$ or $\psi(x;\theta)$. *From (Theorem 1, Yuan et al. 2022). §The analysis shown in (Lindgren et al., 2021) only consider the case when batch size $B = 1$. †SogCLR only converge to an $\mathcal{O}(\upsilon^2)$-stationary solution, where υ is the worst-case feature heterogeneity error satisfying $\sup_{\theta\in\mathbb{R}^p}\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{pos}}}\mathbb{E}_{z\sim\text{Uniform}(\mathbf{D}_{\text{neg}}(x))}\left[|\phi(x;\theta)^\top\psi(z;\theta) - \phi(y;\theta)^\top\psi(z;\theta)|\right] \le \upsilon^2$.

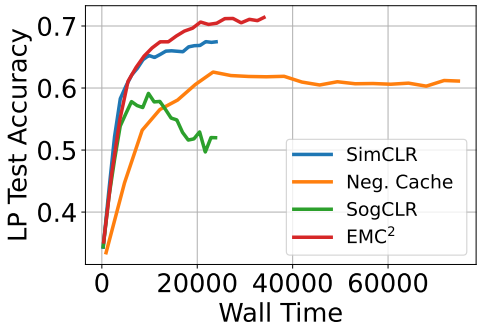

Figure 1. Training 100 epochs on `STL-10` with ResNet-18 using batch size $b = 32$. Horizontal axis is relative to the wall-clock training time in seconds.

works have proposed remedies to the sampling problem. The *negative cache* algorithm (Lindgren et al., 2021) stores features $\{\psi(z;\theta_\tau)\}_{z\in\mathbf{D}_{\text{neg}}(x)}$ from previous iterations and apply the Gumbel-max trick for sampling from an approximate $\mathcal{P}_{x,\theta}$. Though it is proven to converge to a stationary point of (1), the algorithm suffers from high computation and memory complexity. Under a similar purpose, SogCLR (Yuan et al., 2022) proposed a running mean estimator for the normalization sum in $\mathcal{P}_{x,\theta}$. While SogCLR corrects the gradient bias without a significant sampling nor computation overhead, the algorithm is only guaranteed to converge to a neighborhood of stationary points of (1) with a non-vanishing error. A comparison of the above works is in Table 1.

1.2. Our Contributions & Related Works

We propose EMC^2, an Efficient Markov Chain Monte Carlo negative sampling method for Contrastive Learning. Our method departs from the existing approaches since EMC^2 directly tracks $\mathcal{P}_{x,\theta}$ to generate negative samples for $\nabla\mathcal{L}_{\text{neg}}(\theta)$. More specifically,

- The EMC^2 utilizes a Metropolis-Hasting (M-H) algorithm specialized for negative sampling from (5). Moreover, the samples from our M-H algorithm *dynamically adjust* to the evolution of the stochastic gradient (SGD) iterates. This results in a state-dependent SGD scheme that enjoys low memory and computation complexity per iteration.
- We characterize the finite-time convergence rate of EMC^2. We show that it converges in expectation to a stationary point satisfying $\mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] = \mathcal{O}(1/\sqrt{T})$ for some $t \in \{1, ..., T\}$, where T is the number of iterations. Moreover, the latter is neither affected by small batch size, nor the burn-in period with the M-H algorithm; see Table 1.
- Our convergence analysis involves a non-trivial adaptation of the generic result for biased stochastic approximation scheme in (Karimi et al., 2019). Specifically, we prove that the θ dependent kernel which induces the MCMC's Markov chain is ergodic, and is Lipschitz w.r.t. θ.

Figure 1 previews the performance of EMC^2 for pre-training the image encoder on the `STL-10` dataset. Observe that when training with a small batch size, EMC^2 is around 2x faster than SimCLR and SogCLR, and 3x faster than Negative Caching. The rest of this paper is organized as follows. Section 2 develops the EMC^2 algorithm by showing how to combine M-H sampling in an online manner with SGD updates. Section 3 presents the main convergence analysis results. Finally, Section 4 shows the numerical experiments to corroborate our claims on the efficacy of EMC^2.

Related Works. We remark that existing works have considered using Markovian samples that is not i.i.d. in SGD optimization. General convergence analysis results have appeared in (Sun et al., 2018; Doan, 2022) for convex and non-convex optimization with a homogeneous Markov chain, and (Atchadé et al., 2017; Karimi et al., 2019)

with controlled Markov chain. Example applications include policy evaluation in reinforcement learning (Baxter & Bartlett, 2001; Bhandari et al., 2018; Srikant & Ying, 2019), Bayesian optimization for maximum likelihood (De Bortoli et al., 2021), expectation maximization with stochastic samples (Kuhn & Lavielle, 2004).

For tasks related to contrastive learning, prior works have proposed using tree indexed structure to accelerate sampling (Monath et al., 2023), and using the graph structure for graph representation learning (Yang et al., 2020). However, they only proved that the bias of the gradient estimator is bounded and lack convergence analysis for the overall learning algorithm. We remark that there are earlier works on estimating the cross-entropy loss over a large set of classes (Vembu et al., 2012), e.g., by sub-sampling the classes based on kernel methods (Blanc & Rendle, 2018) and random Fourier features (Rawat et al., 2019). In comparison, our work introduces the technique to contrastive learning, and provide a comprehensive theoretical and empirical analysis on its efficacy and convergence properties.

2. Our Proposed Method: EMC^2

This section develops EMC^2 for optimizing the global contrastive loss $\mathcal{L}(\theta)$ in (1). To simplify notation, we define

$$H(x, y; \theta) := \nabla_\theta \left[\phi(x; \theta)^\top \psi(y; \theta)\right] \tag{7}$$

as the gradient of the sample pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Recall from (4) that the population gradient $\nabla\mathcal{L}(\theta)$ is composed of two terms: (i) $\nabla\mathcal{L}_{\rm pos}(\theta)$ maximizes the correlation between positive sample pairs, (ii) $\nabla\mathcal{L}_{\rm neg}(\theta)$ minimizes the correlation between *negative* sample pairs.

To apply SGD on (1), with a positive sample pair drawn uniformly as $(x, y) \sim \mathcal{D}_{\rm pos}$, the vector $-\beta H(x, y; \theta)$ yields an *unbiased* estimate for $\nabla\mathcal{L}_{\rm pos}(\theta)$. Our challenge lies in obtaining an *unbiased* estimate for the negative sample gradient $\nabla\mathcal{L}_{\rm neg}(\theta)$. Observe that

$$\nabla\mathcal{L}_{\rm neg}(\theta) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\rm pos},\ z\sim\mathcal{P}_{x,\theta}} \left[\beta H(x, z; \theta)\right] \tag{8}$$

for $\mathcal{P}_{x,\theta}$ defined in (6). Compared to the case of $\nabla\mathcal{L}_{\rm pos}(\theta)$, the distribution for the tuple (x, z) in the expectation above depends on θ. It follows a softmax distribution (5) with a large summation in the denominator that is difficult, if not impossible, to evaluate when the number of negative samples is large. Furthermore, the distribution also depends on θ and has to be updated *dynamically* as we optimize θ.

We refer to the task of estimating (8) as the *negative sampling* problem. To improve memory consumption and computation cost, our idea is to develop an MCMC negative sampling method that generates the desired samples in an online manner. The resultant SGD method for (1) is then treated as a stochastic approximation scheme with state-dependent samples coming from a *controlled Markov chain*.

2.1. Negative Sampling via MCMC

The MCMC scheme (Robert & Casella, 1999) is a classical yet powerful method for generating samples from an arbitrary distribution π. We focus on the Metropolis-Hastings (M-H) algorithm (Chib & Greenberg, 1995) due to its simplicity. In a nutshell, the algorithm generates new candidate samples from a uniform distribution and adjusts the frequency of samples by a reject/accept mechanism to match the target distribution π. It induces a Markov chain whose stationary distribution is the target distribution.

For the negative sampling problem in (8), with $(x, y) \sim \mathcal{D}_{\rm pos}$, we generate samples $z \sim \mathcal{P}_{x,\theta}$ by running the M-H algorithm. In particular, we assign a state $Z_i \in [m_{\rm neg}]$ for each $i \in [m]$ that corresponds to a positive sample $x_i \in \mathbf{D}_X$. We generate a candidate sample $Z_i' \sim \mathrm{Uniform}([m_{\rm neg}])$, to be accepted as the new sample Z_i^+ with probability:

$$Q_{x_i,\theta}(Z_i', Z_i) = \frac{p_{x_i,\theta}(Z_i')}{p_{x_i,\theta}(Z_i)} = \frac{\exp(\beta\ \phi(x_i; \theta)^\top \psi(Z_i'; \theta))}{\exp(\beta\ \phi(x_i; \theta)^\top \psi(Z_i; \theta))}, \tag{9}$$

where Z_i is the old sample; if the sample Z_i' is rejected, then $Z_i^+ = Z_i$. The above procedure induces a Markov chain $\cdots \to Z_i \to Z_i^+ \to \cdots$ whose stationary distribution coincides with $\mathcal{P}_{x_i,\theta}$. Together with $(x_i, y_i) \equiv (x, y) \sim \mathcal{D}_{\rm pos}$, this allows us to construct an unbiased estimate for $\nabla\mathcal{L}_{\rm neg}(\theta)$ as $\beta H(x_i, Z_i^\infty; \theta)$, where Z_i^∞ is obtained after running the M-H algorithm for a certain number of steps.

We note that evaluating (9) does not require us to compute the partition function in the denominator of (5). As such, the computation complexity of evaluating (9) is only $2C_\phi$. On the other hand, we require storing the state Z_i for each positive sample $x_i \in \mathbf{D}_X$, which results in a memory cost of m integers. The latter needs not be updated for every iteration. Compared to the negative caching algorithm (Lindgren et al., 2021) which requires a cache memory consumption of $m_{\rm neg}d$ real numbers for storing the feature vectors $\psi(z; \theta)$ for all $z \in \mathbf{D}_{\rm neg}(x)$, and a computation cost of $\rho m_{\rm neg} C_\phi$ for updating a ρ-fraction of cache, the MCMC scheme enjoys a lower memory cost and computation complexity.

Single State MCMC vs Multi-State MCMC. Alternatively, one can apply the M-H algorithm to generate samples from the *joint distribution* of (x, z) in (8) using a single state MCMC. However, the corresponding reject/accept ratio [cf. (9)] involves the normalization constant $G(x, \theta) = \sum_{z\in\mathbf{D}_{\rm neg}(x)} \phi(x; \theta)^\top \psi(z; \theta)$ from (5), for which $G(x, \theta)$ and $G(x', \theta)$ have to be re-evaluated when transitioning from $(x, \cdot)$ to $(x', \cdot)$. Implementing such a scheme will require a $2m_{\rm neg}C_\phi$ computation complexity, or require approximating the normalization constant as a constant that is

Algorithm 1 Efficient MCMC Negative Sampling Method for Contrastive Learning (EMC^2)

input: Iteration number T, batch size B, negative batch size R, burn-in period $P < R$, Markov chain state initialization $\{Z_j\}_{j=1}^m$ and step size γ.
for $t = 0, ..., T-1$ **do**
  Draw a mini-batch $[(x_{i_1^{(t)}}, y_{i_1^{(t)}}), ..., (x_{i_B^{(t)}}, y_{i_B^{(t)}})] \sim \mathcal{D}_{\text{pos}}$, with indices $\{i_1^{(t)}, ..., i_B^{(t)}\} \subseteq [m]$.
  for $k = 1, ..., B$; $r = 0, ..., R-1$ **do**
    Draw negative sample $Z'_{i_k^{(t)}} \sim \text{Unif}(\mathbf{D}_{\text{neg}}(x_{i_k^{(t)}}))$.
    Update the Markov chain state $Z_{i_k^{(t)}}$ by

$$Z_{i_k^{(t)}} \leftarrow \begin{cases} Z'_{i_k^{(t)}} & \text{w.p. } \min\left\{1,\ Q_{x_{i_k^{(t)}},\theta_t}(Z'_{i_k^{(t)}}, Z_{i_k^{(t)}})\right\}, \\ Z_{i_k^{(t)}} & \text{otherwise,} \end{cases}$$

    where $Q_{x,\theta}(Z', Z)$ is defined in (9).
    If $r \geq P$, store the sample $\tilde{Z}_{i_k^{(t)}}^{(r)} = Z_{i_k^{(t)}}$.
  end for
  Update the model θ using (10).
end for

independent of x, both of which are encouraging the use of a multi-state Markov chain as illustrated in (9).

2.2. State-dependent SGD Algorithm

As the target distribution in (8) depends on θ, applying SGD with the classical MCMC scheme or M-H algorithm requires first *freezing* θ and then simulating the Markov chain for a considerable amount of time *prior to* forming the stochastic gradient. The latter is known as the *burn-in* period for MCMC and incurs additional complexity for every SGD iteration.

We propose to adjust the Markov chains in an online manner as θ is updated, i.e., the method shall not maintain a long burn-in period. In particular, at each SGD iteration, the M-H updates are executed with initialization given by the state stored previously. Meanwhile, the Markov chains with the reject/accept ratio (9) are controlled by the current θ.

Let t denotes the SGD iteration index and θ_t is the corresponding model. Suppose that a mini-batch of B positive samples are drawn $\{x_{i_k^{(t)}}, y_{i_k^{(t)}}\}_{k=1}^B$ and $\{\{\tilde{Z}_{i_k^{(t)}}^{(r)}\}_{r=0}^{R-1}\}_{k=1}^B$ are the sequences of samples generated by R steps of the M-H algorithm, where the latter is initialized by the states $Z_{i_k^{(t)}}$ from the previous SGD iteration. Our idea is to update the model θ by the following recursion:

$$\theta_{t+1} = \theta_t - \gamma \mathcal{H}\left(\boldsymbol{\xi}_{t+1}; \theta_t\right), \tag{10}$$

where $\boldsymbol{\xi}_{t+1} := \{x_{i_k^{(t)}}, y_{i_k^{(t)}}, \{\tilde{Z}_{i_k^{(t)}}^{(r)}\}_{r=P}^{R-1}\}_{k=1}^B$ collects the samples used, $\gamma > 0$ is the SGD step size, and

$$\mathcal{H}\left(\boldsymbol{\xi}_{t+1}; \theta_t\right) := -\frac{\beta}{B}\sum_{k=1}^{B} H(x_{i_k^{(t)}}, y_{i_k^{(t)}}; \theta_t) + \frac{\beta}{B(R-P)}\sum_{k=1}^{B}\sum_{r=P}^{R-1} H(x_{i_k^{(t)}}, \tilde{Z}_{i_k^{(t)}}^{(r)}; \theta_t), \tag{11}$$

where P is an adjustable burn-in parameter. The overall algorithm is summarized in Algorithm 1.

We observe that (10) is different from a standard SGD algorithm as $\mathcal{H}(\boldsymbol{\xi}_{t+1}; \theta_t)$ corresponds to a *biased* estimate for the gradient $\nabla\mathcal{L}(\theta_t)$ of the contrastive learning loss (1) in general. Instead, the algorithm belongs to the more general class of *biased stochastic approximation* scheme (Karimi et al., 2019) whose stochastic updates are driven by the Markov chain $\cdots \to \boldsymbol{\xi}_t \to \boldsymbol{\xi}_{t+1} \to \cdots$. Moreover, the Markov transition kernel is controlled by the current model θ_t that is updated simultaneously.

To reduce bias, a traditional approach is to consider $P \to \infty$, $R = P + 1$ which will ensure that the M-H algorithm can generate its samples from the stationary distribution. However, as we shall demonstrate in Section 3, under suitable and verifiable conditions, the θ-dependent Markov transition kernel is Lipschitz continuous and uniform geometrically ergodic. Subsequently, it ensures the convergence of (10) towards a stationary point of (1) for any P, R.

3. Convergence Analysis

This section shows that EMC^2 converges to a stationary point of (1). Our analysis is organized as follows: first we characterize the mixing rate of the Markov chain $\cdots \to \boldsymbol{\xi}_t \to \boldsymbol{\xi}_{t+1} \to \cdots$ in (10) and its smoothness property, then we analyze the convergence of the algorithm through studying the latter as a special case of the biased stochastic approximation scheme in (Karimi et al., 2019). Finally, we present our main results in Theorem 3.7 and Corollary 3.8.

3.1. Analysis of MCMC Negative Sampling

To enable the convergence of (10) utilizing the Markov chain $\{\boldsymbol{\xi}_t\}_{t\geq 0}$, intuitively it requires the Markov chain to (i) have a fast mixing time, and (ii) satisfy certain smoothness property as θ_t is gradually updated. To fix notation, we denote the state-dependent Markov transition kernel $\mathsf{P}_\theta : \Xi \times \Xi \to \mathbb{R}_+$ such that $\boldsymbol{\xi}_{t+1} \sim \mathsf{P}_{\theta_t}(\xi_t, \cdot)$. For simplicity, we analyze the latter transition probability of $\boldsymbol{\xi}_{t+1}$ when $P = R - 1$, i.e., the state dependent SGD update only takes the last sample in the M-H algorithm. Moreover, with a slight abuse of notation, we consider the collection of all hidden states $\boldsymbol{\xi}_{t+1} = \{\tilde{Z}_i^{(R-1)}\}_{i=1}^m \in \Xi$ which does not include x, y. As the latter is drawn i.i.d., it does not affect the convergence analysis for (10) in Section 3.2.

For our first task in showing the fast mixing of $\{\xi_t\}_{t\geq 0}$, it suffices to show that the Markov chain induced by $\mathtt{P}_\theta$ is *geometrically ergodic* for any fixed θ. To this end, we note that as the target distribution $\mathcal{P}_{x,\theta}$ belongs to the exponential family, applying (Theorem 2.1, Roberts & Tweedie 1996) implies ergodicity. To formally state the result, we impose the following assumptions:

Assumption 3.1. There exists $c > 0$ such that

$$\max\{\|\phi(x;\theta)\|, \|\psi(x;\theta)\|\} \leq c, \ \forall\, \theta \in \mathbb{R}^p, \ x \in \mathcal{X}. \quad (12)$$

Assumption 3.1 is common with $c \leq 1$ for contrastive learning problems (Chen et al., 2020; Radford et al., 2021). This effectively controls the behavior of $Q_{x,\theta}(z, Z)$ in the M-H algorithm (9) and thus the transition kernel $\mathtt{P}_\theta$. The following lemma shows the geometric ergodicity property for the Markov chain:

Lemma 3.2. *Under Assumption 3.1. For any $\theta \in \mathbb{R}^p$ and any initialization $\tilde{\boldsymbol{\xi}}_0 \in \Xi$, the Markov chain $\tilde{\boldsymbol{\xi}}_0 \to \tilde{\boldsymbol{\xi}}_1 \to \cdots$ induced by the transition kernel $\mathtt{P}_\theta$ converges geometrically to the stationary distribution $\pi_{\boldsymbol{x},\theta}(\boldsymbol{z}) = \prod_{i=1}^m p_{x_i,\theta}(z_i)$, where $\boldsymbol{z} = (z_i)_{i=1}^m$. In particular, it holds*

$$\big|\mathbb{P}(\tilde{\boldsymbol{\xi}}_\tau = \boldsymbol{z}) - \pi_{\boldsymbol{x},\theta}(\boldsymbol{z})\big| \leq \left(1 - \frac{BR}{2mm_{\mathrm{neg}}\exp(2c^2\beta)}\right)^\tau, \quad (13)$$

for any $\boldsymbol{z}$ and any $\tau \geq 0$.

See Appendix A for the proof, which relies on (Theorem 1.3, Mengersen & Tweedie 1996). Moreover, from (13), we observe that through mini-batch sampling among $\mathcal{D}_{\mathrm{pos}}$ and $\mathcal{P}_{x,\theta}$, the mixing rate improves by BR when consuming BR samples in EMC^2.

Our second task consists of showing that the Markov transition kernel is smooth w.r.t. θ. We require the following condition the models ϕ, ψ:

Assumption 3.3. There exists $L_P \geq 0$ such that for any $\theta, \theta' \in \mathbb{R}^p, (x, y) \in \mathrm{supp}(\mathcal{D}_{\mathrm{pos}}), z \in \mathbf{D}_{\mathrm{neg}}(x)$,

$$\|\phi(x;\theta)^\top\psi(z;\theta) - \phi(x;\theta')^\top\psi(z;\theta')\| \leq L_P\|\theta - \theta'\|. \quad (14)$$

Assumption 3.3 describes the Lipschitz condition of the similarity function $\phi(x;\theta)^\top\psi(z;\theta)$, which is important in establishing the Lipschitz condition of the transition kernel w.r.t. the model parameter θ. For example, it can be satisfied as $L_P = Lc$ for L-Lipschitz functions ϕ, ψ w.r.t. θ with bounded norm as in Assumption 3.1.

The following lemma shows that the transition kernel $\mathtt{P}_\theta$ is Lipschitz continuous with respect to θ, as follows:

Lemma 3.4. *Under Assumption 3.1, 3.3. It holds:*

$$|\mathtt{P}_\theta(\boldsymbol{\xi}, \boldsymbol{\xi}') - \mathtt{P}_{\theta'}(\boldsymbol{\xi}, \boldsymbol{\xi}')| \leq 2^{R+1}BL_P\exp(2c^2\beta)\beta\|\theta - \theta'\|, \quad (15)$$

for any $\boldsymbol{\xi}, \boldsymbol{\xi}' \in \Xi$ and any $\theta, \theta' \in \mathbb{R}^p$.

The proof can be found in Appendix B. We remark that the dependence on $2^{R+1}B$ in the Lipschitz constant is conservative. We anticipate the effective Lipschitz constant for the transition kernel to be much smaller than in (15). For example, (43) has applied a loose bound of 1 to control the transition probability.

3.2. Convergence to Stationary Point

Equipped with Lemma 3.2 & 3.4, we are ready to derive the convergence rate of Algorithm 1 towards a stationary point of (1). Our idea is to treat (10) as a biased stochastic approximation scheme and analyze the latter using the general convergence theories given in (Karimi et al., 2019).

We require the following conditions:

Assumption 3.5. There exists $L_H \geq 0$ such that for any $\theta, \theta' \in \mathbb{R}^p, x \in \mathbf{D}_X, y \in \mathbf{D}_Y$,

$$\|H(x, y;\theta) - H(x, y;\theta')\| \leq L_H\|\theta - \theta'\|. \quad (16)$$

Assumption 3.6. There exists $\sigma \geq 0$ such that for any $\theta \in \mathbb{R}^p, (x, y) \in \mathrm{supp}(\mathcal{D}_{\mathrm{pos}}), z \in \mathbf{D}_{\mathrm{neg}}(x)$,

$$\|H(x, y;\theta) - \mathbb{E}_{\mathcal{D}_{\mathrm{pos}}}[H(x, y;\theta)]\| \leq \sigma, \quad (17)$$

$$\|H(x, z;\theta) - \mathbb{E}_{\mathcal{D}_{\mathrm{pos}},\mathcal{P}_{x,\theta}}[H(x, z;\theta)]\| \leq \sigma. \quad (18)$$

Assumptions 3.5, 3.6 impose a uniform bound on the smoothness condition of the similarity function and the variance of stochastic gradient. These assumptions are standard in the literature, e.g., they are also used in (Lindgren et al., 2021; Yuan et al., 2022).

We present the main convergence result for EMC^2:

Theorem 3.7. *Under Assumptions 3.1, 3.3, 3.5, 3.6. For any $T \geq 1$, there exists a sufficiently small step size $\gamma > 0$ such that the iterate θ_t generated by EMC^2 satisfies*

$$\begin{aligned} &T^{-1}\textstyle\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] \quad (19)\\ &\leq \frac{2\mathcal{L}_{0,T}}{\gamma T} + \frac{24\beta\sigma mm_{\mathrm{neg}}\exp(2c^2\beta)}{\gamma BRT} + \gamma\beta^3\sigma^2 \\ &\times \mathcal{O}\Big(\frac{L_H mm_{\mathrm{neg}}\exp(2c^2\beta)}{BR} + \frac{L_P 2^R m^2 m_{\mathrm{neg}}^2\exp(6c^2\beta)}{BR^2}\Big) \end{aligned}$$

where the above expectation is taken with respect to the randomness in the algorithm, and $\mathcal{L}_{0,T} = \mathbb{E}[\mathcal{L}(\theta_0) - \mathcal{L}(\theta_T)] \leq 4\beta$.

See Appendix C for the proof. Theorem 3.7 implies that in expectation, EMC^2 finds an $\mathcal{O}(\frac{1}{\gamma T} + \gamma)$ stationary solution to (1) in T iterations, for any $T \geq 1$.

For a sufficiently large T, setting $\gamma = 1/\sqrt{T}$ yields a global convergence rate of $\mathcal{O}(1/\sqrt{T})$ regardless of the batch size B nor the burn-in period controlled by P, R, as observed in the following corollary:

Corollary 3.8. *Under Assumptions 3.1, 3.3, 3.5, 3.6. For a sufficiently large T, choosing $\gamma = \frac{1}{\sqrt{T}}$, $R = 2$, guarantees that the iterates generated by EMC^2 satisfy:*

$$T^{-1}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] \tag{20}$$
$$\leq \frac{8\beta}{\sqrt{T}} + \frac{12\beta\sigma m m_{\text{neg}}\exp(2c^2\beta)}{B^2\sqrt{T}} + \frac{\beta^3\sigma^2}{\sqrt{T}}$$
$$\times\,\mathcal{O}\Big(\frac{L_H m m_{\text{neg}}\exp(2c^2\beta)}{B} + \frac{L_P m^2 m_{\text{neg}}^2\exp(6c^2\beta)}{B}\Big).$$

We have presented a simplified version of the above result in Table 1.

4. Numerical Experiments: Unimodal Pre-Training of Image Encoder

This section examines the performance of EMC^2 for the contrastive learning task on training image encoders. We remark that although the analysis is only shown for the SGD optimizer with EMC^2, the stochastic gradient approximation in (11) can be applied to other popular first-order optimizers such as Adam.

Our experiments consider the task of pre-training self-supervised image encoder similar to (Chen et al., 2020). This task is described by an instance of (1) with unimodal encoders, i.e., $\phi(\cdot;\theta) \equiv \psi(\cdot;\theta)$ for any $\theta \in \mathbb{R}^p$. Moreover, the self-supervised dataset is specified with a set of augmented images. Let $\mathcal{A}$ be a set of image augmentation operators $g : \mathcal{X} \to \mathcal{X}$, we specify $(\mathcal{D}_{\text{pos}}, \mathbf{D}_{\text{neg}}(\cdot))$ as:

$$\begin{cases}\mathcal{D}_{\text{pos}} = \text{Uniform}(\{(g(x), h(x)) : x \in \mathbf{D};\ g, h \in \mathcal{A}\}),\\ \mathbf{D}_{\text{neg}}(h(x)) = \{g(y) : y \neq x, y \in \mathbf{D}; g, h \in \mathcal{A}\}.\end{cases} \tag{21}$$

For the dataset $\mathbf{D} \subseteq \mathcal{X}$, the positive pairs are set of the random augmentations of the same image, while for $x \in \mathbf{D}$, its negative pairs are set to be the whole dataset with augmentation except that of x.

4.1. Implementation of EMC^2

Image Augmentation. We follow previous studies on applying input augmentations for contrastive learning on image data (Xie et al., 2020) which would lead to samples from a distribution $\mathcal{D}_{\text{pos}}$ with infinite support. Despite the fact that EMC^2 is designed to track the Markov chain state for each image in $x_1, ..., x_m$, in practice we approximate Algorithm 1 by maintaining a Markov chain state for each set of $\{g(x_1) : g \in \mathcal{A}\}, ..., \{g(x_m) : g \in \mathcal{A}\}$[2]. This approximate algorithm remains practical and performant as verified empirically in Section 4.3.

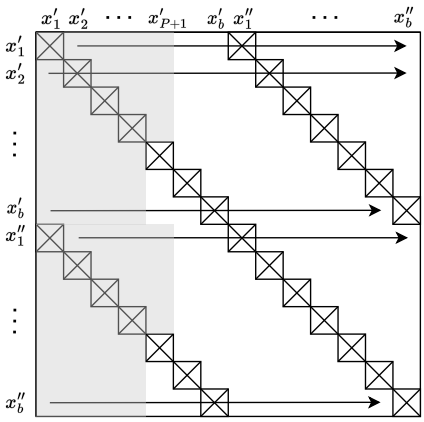

Figure 2. Illustration of mini-batch MCMC sampling with 2 augmentations (x'_i, x''_i) of each image x_i. Each horizontal arrow represents a distribution tracked by one Markov chain and the direction of M-H reject/accept step. Shaded area represents the samples used for burn-in with burn-in period $P < 2b-2$. Crossed-out diagonals are not regarded as negative samples.

Mini-batch Sampling. As proposed by (Chen et al., 2020), it is efficient to draw a mini-batch and use the in-batch samples as negative samples. We can implement M-H reject/accept steps on top of the same mini-batch sampling scheme with $R = B - 2$. In Figure 2, we illustrate a similarity matrix formed by the matrix product $\mathbf{V}^\top\mathbf{V}$ such that $\mathbf{V} = [\phi(x'_1), ..., \phi(x'_b), \phi(x''_1), ..., \phi(x''_b)] \in \mathbb{R}^{d\times 2b}$ contains the feature vectors of a mini-batch of size $B = 2b$ with two random augmentations (x'_i, x''_i) of the same image x_i, $i \in [b]$. EMC^2 can regard the in-batch samples $\{x'_1, ..., x'_b, x''_1, ..., x''_b\}\backslash\{x'_j, x''_j\}$ as negative samples from the prior distribution $\text{Uniform}(\mathbf{D}_{\text{neg}}(x_j))$ and perform M-H acceptance/rejection steps over this negative set. As illustrated in Figure 2, the M-H algorithm can run in parallel across the multiple Markov chains. For the experiments, we consume burn-in samples from the same mini-batch without requiring extra samples.

4.2. Dataset and Metrics

We concentrate on two common datasets under this setup – `STL-10` and `Imagenet-100`. We apply the Adam optimizer for training with `STL-10` and the LARS optimizer with `Imagenet-100`. Details of the datasets and hyperparameters used can be found in Appendix D.

[2]When the state Z_i is referred outside the iteration it was sampled, it points to the image x_i to avoid storing the augmented image $g(x_i)$ in memory.

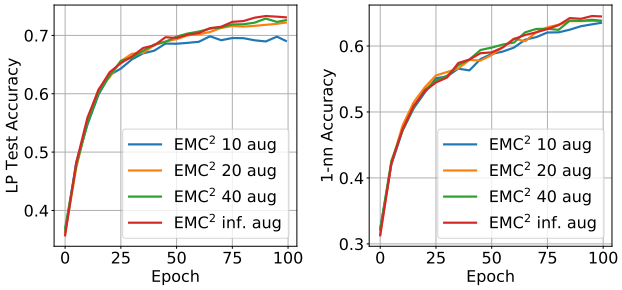

Figure 3. Comparison between different sizes of pre-augmented STL-10 with ResNet-18 and batch size $b = 256$. Horizontal axis is relative to the number of samples accessed.

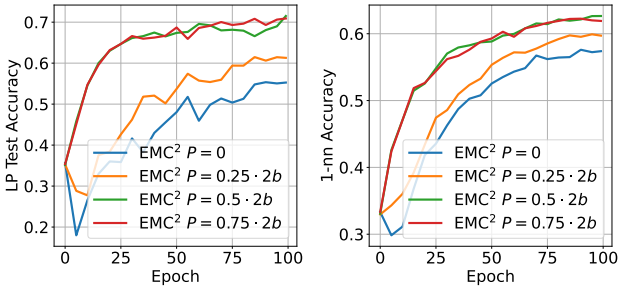

Figure 4. Comparison between different numbers of burn-in negative samples P for each Markov chain state Z_i.

To benchmark the performance of EMC^2 and other algorithms, we report linear probe (LP) accuracy and 1-nearest-neighbor (1-NN) accuracy of the embeddings produced by the encoder, which are standard metrics for evaluating image encoders (Chen et al., 2020; Kukleva et al., 2023). They correspond to the linear separability of features on a unit hypersphere (Wang & Isola, 2020) and the local geometry of the learnt features.

Since the negative cache algorithm (Lindgren et al., 2021) requires storing feature vectors of negatives, we are restricted to implementing on an alternative loss function with $\mathbf{D}_{\text{neg}}(x) = \{y : y \in \mathbf{D} \backslash \{x\}\}$, i.e., images without augmentation. Also, note that negative cache algorithm uses B extra samples per iteration as the Gumbel-max negative sampling requires out-of-batch samples.

4.3. Effects of Image Augmentations

We first examine the effect of the number of pre-computed image augmentations, $|\mathcal{A}|$, on the performance of EMC^2. Note that as suggested in (Chen et al., 2020), increasing $|\mathcal{A}|$ can lead to improved performance. We also compare a heuristic extension of EMC^2 that effectively deploys an infinite number of augmentations: the augmented images are generated on-the-fly at every iteration and a single Markov chain state Z_i is maintained for all augmentations of the same image x_i.

From Figure 3, as the number of pre-computed augmentations increases, the performance approaches to that of the infinite augmentation algorithm. This illustrates that EMC^2 adapts to random augmentations. In later experiments with image augmentations, we implement EMC^2 with the infinite augmentation setting for the best performance.

4.4. Effects of Burn-in Period

For every positive pair (x_i, y_i) in a mini-batch, EMC^2 uses P burn-in negative samples to warm-up the Markov chain state Z_i of sample x_i. We study the effect of P by comparing the performance for $P \in \{0, \frac{b}{2}, b, \frac{3}{2b}\}$. From Figure 4, we observe that the performance of EMC^2 improves by increasing P, and the improvement stops at a certain threshold of around $P = b$. This amounts to the mechanism of Markov chain convergence where the negative samples after the burn-in period are more accurate to the true sample distribution in (5).

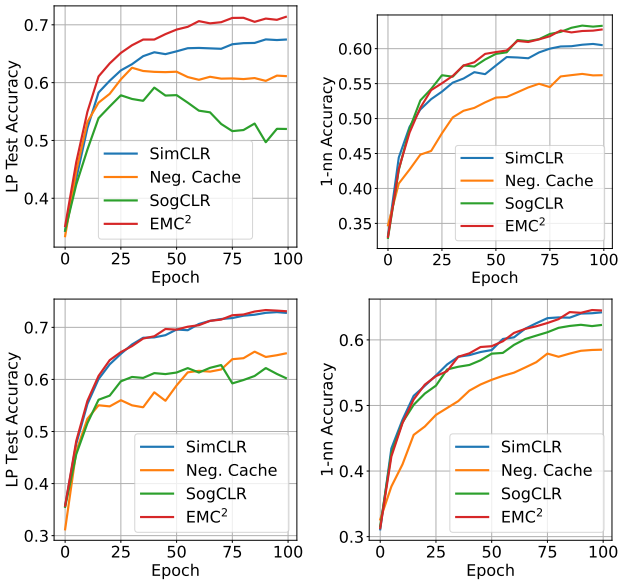

Figure 5. Comparison on STL-10 with ResNet-18 using batch size (top) $b = 32$, (bottom) $b = 256$. Horizontal axis is relative to the number of samples accessed.

4.5. Comparison to Baselines

We compare EMC^2 to baseline algorithms: Negative Cache (Lindgren et al., 2021), SimCLR (Chen et al., 2020), SogCLR (Yuan et al., 2022). We first examine the effects of batch sizes $b \in \{32, 256\}$ for the experiments on STL-10, where a standard setting for this dataset is $b = 256$, yet we note that a small batch size is often preferred for training large models due to limitations on GPU memory. Observe from Figure 5 that EMC^2 consistently outperforms the other

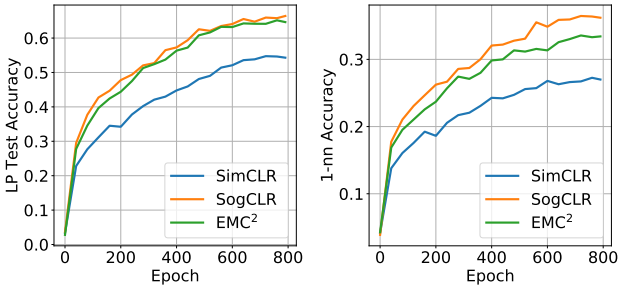

Figure 6. Comparison on `Imagenet-100` with ResNet-50 using batch size $b = 256$. Horizontal axis is relative to the number of samples accessed.

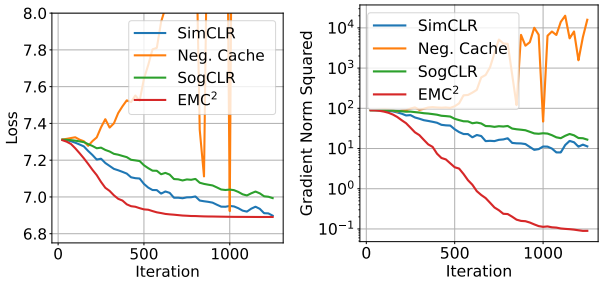

Figure 7. Comparison on a subset of `STL-10` using the first 500 images and pre-computed two augmentations for each image. Trained using SGD with batch size $b = 4$.

algorithms across all metrics regardless the choice of batch size. This is in line with the intuition that SimCLR is bottlenecked by a poor loss distribution on small batch size. For SogCLR, we suspect the performance drop on linear probe accuracy is due to the violation of feature homogeneity assumption on the setup of `STL-10` with Resnet-18. For negative cache, its performance on using one negative with stale cache error does not keep up with in-batch negatives approaches.

Figure 6 shows a similar experiment but on the more difficult dataset `Imagenet-100`. Note that the negative caching algorithm is not run due to the excess computation complexity in cache refreshing and Gumbel-max sampling. Observe that EMC^2 shows performance gain over SimCLR when trained on the standard batch size $b = 256$, while the performance is on par with SogCLR.

From the above experiments, we observe that EMC^2 delivers a consistent performance over different datasets, batch size, that is on par with the best compared baseline for the respective tasks. In Appendix D.2, additional time-complexity comparison on `STL-10` is presented to demonstrate the practical benefits of EMC^2.

Convergence to Stationary Solution. To examine the accuracy of EMC^2, we construct a simple dataset $(\mathcal{D}_{\text{pos}}, \mathbf{D}_{\text{neg}})$ by taking the first 500 images from `STL-10` and using two pre-computed augmentations for each image. Under this setting, the exact value of $\mathcal{L}(\theta)$ in (1) and $\|\nabla\mathcal{L}(\theta)\|^2$ in (4) can be evaluated as shown in Figure 7. We observe that the solution found by EMC^2 is more accurate than the other baselines by 2 orders of magnitude in terms of squared gradient norm.

The code used in the experiments are available at https://github.com/amazon-science/contrastive_emc2.

5. Conclusion

This paper proposed a novel method EMC^2 for optimizing the global contrastive loss (1). The algorithm combines an adaptively adjusted MCMC scheme for generating negative samples with a standard SGD update. We prove that EMC^2 enjoys low memory and computation complexities, and admits a global convergence rate of $\mathcal{O}(1/\sqrt{T})$ towards a stationary solution for (1). Numerical experiments show that EMC^2 enables small batch training for contrastive learning which is due to its global convergence property. We hope this work inspires future research into advanced sampling methods for contrastive learning, such as the Langevin dynamics, and the convergence property of optimization algorithms that rely on them.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Atchadé, Y. F., Fort, G., and Moulines, E. On perturbed proximal gradient algorithms. *The Journal of Machine Learning Research*, 18(1):310–342, 2017.

Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar,

S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

Bardes, A., Ponce, J., and LeCun, Y. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *journal of artificial intelligence research*, 15: 319–350, 2001.

Bhandari, J., Russo, D., and Singal, R. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on learning theory*, pp. 1691–1692. PMLR, 2018.

Blanc, G. and Rendle, S. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning*, pp. 590–599. PMLR, 2018.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2818–2829, 2023.

Chib, S. and Greenberg, E. Understanding the metropolis-hastings algorithm. *American statistician*, pp. 327–335, 1995.

De Bortoli, V., Durmus, A., Pereyra, M., and Vidal, A. F. Efficient stochastic optimisation by unadjusted langevin monte carlo: Application to maximum marginal likelihood and empirical bayesian estimation. *Statistics and Computing*, 31:1–18, 2021.

Doan, T. T. Finite-time analysis of markov gradient descent. *IEEE Transactions on Automatic Control*, 68(4):2140–2153, 2022.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Karimi, B., Miasojedow, B., Moulines, E., and Wai, H.-T. Non-asymptotic analysis of biased stochastic approximation scheme. In *Conference on Learning Theory*, pp. 1944–1974. PMLR, 2019.

Kuhn, E. and Lavielle, M. Coupling a stochastic approximation version of em with an mcmc procedure. *ESAIM: Probability and Statistics*, 8:115–131, 2004.

Kukleva, A., Böhle, M., Schiele, B., Kuehne, H., and Rupprecht, C. Temperature schedules for self-supervised contrastive methods on long-tail data. *arXiv preprint arXiv:2303.13664*, 2023.

Lee, C.-Y., Li, C.-L., Zhang, H., Dozat, T., Perot, V., Su, G., Zhang, X., Sohn, K., Glushnev, N., Wang, R., et al. Formnetv2: Multimodal graph contrastive learning for form document information extraction. *arXiv preprint arXiv:2305.02549*, 2023.

Li, C., Gan, Z., Yang, Z., Yang, J., Li, L., Wang, L., and Gao, J. Multimodal foundation models: From specialists to general-purpose assistants. *arXiv preprint arXiv:2309.10020*, 1(2):2, 2023.

Lindgren, E., Reddi, S., Guo, R., and Kumar, S. Efficient training of retrieval models using negative cache. *Advances in Neural Information Processing Systems*, 34: 4134–4146, 2021.

Logeswaran, L. and Lee, H. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*, 2018.

Mengersen, K. L. and Tweedie, R. L. Rates of convergence of the hastings and metropolis algorithms. *The annals of Statistics*, 24(1):101–121, 1996.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Monath, N., Zaheer, M., Allen, K., and McCallum, A. Improving dual-encoder training through dynamic indexes for negative mining. In *International Conference on Artificial Intelligence and Statistics*, pp. 9308–9330. PMLR, 2023.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Rawat, A. S., Chen, J., Yu, F. X. X., Suresh, A. T., and Kumar, S. Sampled softmax with random fourier features. *Advances in Neural Information Processing Systems*, 32, 2019.

Robert, C. P. and Casella, G. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

Roberts, G. O. and Tweedie, R. L. Geometric convergence and central limit theorems for multidimensional hastings and metropolis algorithms. *Biometrika*, 83(1):95–110, 1996.

Robinson, J., Chuang, C.-Y., Sra, S., and Jegelka, S. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

Srikant, R. and Ying, L. Finite-time error bounds for linear stochastic approximation andtd learning. In *Conference on Learning Theory*, pp. 2803–2830. PMLR, 2019.

Sun, T., Sun, Y., and Yin, W. On markov chain gradient descent. *Advances in neural information processing systems*, 31, 2018.

Vembu, S., Gartner, T., and Boley, M. Probabilistic structured predictors. *arXiv preprint arXiv:1205.2610*, 2012.

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pp. 9929–9939. PMLR, 2020.

Xie, Q., Dai, Z., Hovy, E., Luong, T., and Le, Q. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, 33: 6256–6268, 2020.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florencio, D., Zhang, C., Che, W., et al. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*, 2020.

Yang, Z., Ding, M., Zhou, C., Yang, H., Zhou, J., and Tang, J. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1666–1676, 2020.

Yuan, Z., Wu, Y., Qiu, Z.-H., Du, X., Zhang, L., Zhou, D., and Yang, T. Provable stochastic optimization for global contrastive learning: Small batch does not harm performance. In *International Conference on Machine Learning*, pp. 25760–25782. PMLR, 2022.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021.

Zimmermann, R. S., Sharma, Y., Schneider, S., Bethge, M., and Brendel, W. Contrastive learning inverts the data generating process. In *International Conference on Machine Learning*, pp. 12979–12990. PMLR, 2021.

A. Proof of Lemma 3.2

Notice that our target softmax distribution $\mathcal{P}_{x,\theta}$ has a finite support with each state having non-zero mass. For any $\theta \in \mathbb{R}^p$,

$$\min_{x,z,Z} Q_{x,\theta}(z, Z) = \min_{x,z,Z} \frac{\exp(\beta\, \phi(x;\theta)^\top \psi(z;\theta))}{\exp(\beta\, \phi(x;\theta)^\top \psi(Z;\theta))} \overset{\text{(Assm. 3.1)}}{\geq} \exp(-2c^2\beta) \tag{22}$$

Now recall that $m_{\text{neg}} = |\mathbf{D}_{\text{neg}}(x)|$ for any $x \in \mathbf{D}_X$, i.e., the state space consists of m_{neg} states on each Markov chain Z_i. Then the smallest probability to transition to state z from any initial state is lower bounded by $m_{\text{neg}}^{-1} \exp(-2c^2\beta)$, where the factor m_{neg}^{-1} comes from using the uniform prior $\text{Uniform}(\mathbf{D}_{\text{neg}}(x))$ in Algorithm 1.

By Theorem 1.3 of (Mengersen & Tweedie, 1996), the transition kernel of each Markov chain of Z_i converges to its target distribution $\mathcal{P}_{x,\theta}$ at a geometric rate of $1 - m_{\text{neg}}^{-1} \exp(-2c^2\beta)$. Therefore, the joint Markov chain of states $(Z_1, ..., Z_m)$ in Algorithm 1 has a mixing rate of $1 - m^{-1} m_{\text{neg}}^{-1} \exp(-2c^2\beta)$ when the batch size $B = 1$. As we perform mini-batch sampling of batch size B on the expectation over $(x, y) \sim \mathcal{D}_{\text{pos}}$ and R steps of M-H algorithm over $z \sim \mathcal{P}_{x,\theta}$ *in parallel*, the mixing rate can be improved to $[1 - Bm^{-1} m_{\text{neg}}^{-1} \exp(-2c^2\beta)]^R$. We can simplify the rate by the following upper bound, as

$$\left[1 - \frac{B}{m m_{\text{neg}} \exp(2c^2\beta)}\right]^R \leq \exp\left(-\frac{BR}{m m_{\text{neg}} \exp(2c^2\beta)}\right) \leq 1 - \frac{BR}{2m m_{\text{neg}} \exp(2c^2\beta)} \tag{23}$$

where the first inequality uses $1 - x \leq \exp(-x)\ \forall x \in \mathbb{R}$, the second inequality uses $\exp(-x) \leq 1 - x/2$ for $x \in [0, 1]$ and the assumption that $BR \leq m m_{\text{neg}} \exp(2c^2\beta)$. □

B. Proof of Lemma 3.4

For each Markov chain corresponding to Z_i in the M-H algorithm, the acceptance probability is given by $\min\{1, Q_{x_i,\theta}(Z, Z')\}$. To establish Lemma 3.4, we first observe that for any $Z, Z' \in \mathbf{D}_{\text{neg}}(x_i)$ and $\theta, \theta' \in \mathbb{R}^p$, it holds

$$\begin{aligned}
&|Q_{x,\theta}(Z, Z') - Q_{x,\theta'}(Z, Z')| && (24)\\
&= |\exp\left(\beta\phi(x;\theta)^\top \psi(Z;\theta) - \beta\phi(x;\theta)^\top \psi(Z';\theta)\right) - \exp\left(\beta\phi(x;\theta')^\top \psi(Z;\theta') - \beta\phi(x;\theta')^\top \psi(Z';\theta')\right)| && (25)\\
&\leq \exp(2c^2\beta) \cdot \beta \cdot |\phi(x;\theta)^\top \psi(Z;\theta) - \phi(x;\theta)^\top \psi(Z';\theta) - (\phi(x;\theta')^\top \psi(Z;\theta') - \phi(x;\theta')^\top \psi(Z';\theta'))| && (26)\\
&\overset{\text{(Assm. 3.3)}}{\leq} 2L_P \cdot \exp(2c^2\beta) \cdot \beta \cdot \|\theta - \theta'\| && (27)
\end{aligned}$$

where (26) uses the fact that $\exp(\cdot)$ is $\exp(2c^2\beta)$-Lipschitz when restricted on the domain $[-2c^2\beta, 2c^2\beta]$ and the restriction is due to Assumption 3.1.

Now denote $\mathbf{K}_{i,\theta} \in \mathbb{R}^{m_{\text{neg}} \times m_{\text{neg}}}$ as the transition matrix of Z_i and

$$[\mathbf{K}_{i,\theta}]_{z',z} = \begin{cases} \frac{1}{m_{\text{neg}}} \cdot \min\{1, Q_{x_i,\theta}(z, z')\} & \text{if } z \neq z', \\ \frac{1}{m_{\text{neg}}} + \frac{1}{m_{\text{neg}}} \sum_{j \in [m_{\text{neg}}] \setminus \{z\}} (1 - \min\{1, Q_{x_i,\theta}(z, j)\}) & \text{if } z = z'. \end{cases} \tag{28}$$

Then for $z \neq z'$,

$$\begin{aligned}
|[\mathbf{K}_{i,\theta}]_{z',z} - [\mathbf{K}_{i,\theta'}]_{z',z}| &= \frac{1}{m_{\text{neg}}} |\min\{1, Q_{x_i,\theta}(z, z')\} - \min\{1, Q_{x_i,\theta'}(z, z')\}| && (29)\\
&\leq \frac{1}{m_{\text{neg}}} |Q_{x_i,\theta}(z, z') - Q_{x_i,\theta'}(z, z')| \overset{(27)}{\leq} \frac{1}{m_{\text{neg}}} \cdot 2L_P \cdot \exp(2c^2\beta) \cdot \beta \cdot \|\theta - \theta'\| && (30)
\end{aligned}$$

and for $z = z'$,

$$|[\mathbf{K}_{i,\theta}]_{z',z} - [\mathbf{K}_{i,\theta'}]_{z',z}| = \left| \frac{1}{m_{\text{neg}}} \sum_{j \in [m_{\text{neg}}] \setminus \{z\}} (-\min\{1, Q_{x_i,\theta}(z, j)\} + \min\{1, Q_{x_i,\theta'}(z, j)\}) \right| \tag{31}$$

$$\leq \frac{1}{m_{\text{neg}}} \sum_{j \in [m_{\text{neg}}] \setminus \{z\}} |Q_{x_i,\theta}(z,j) - Q_{x_i,\theta'}(z,j)| \overset{(27)}{\leq} 2L_P \cdot \exp(2c^2\beta) \cdot \beta \cdot \|\theta - \theta'\| \tag{32}$$

To quantify the smoothness of the transition kernel with R Metropolis-Hastings steps, we observe that by the notation $[\mathbf{K}]_{z,z'} = \mathbf{e}_z^\top \mathbf{K} \mathbf{e}_{z'}$ for some basis vectors $\mathbf{e}_z, \mathbf{e}_{z'}$,

$$\max_{z,z'} |\mathtt{P}_\theta(Z_i^{(0)} = z; Z_i^{(R)} = z') - \mathtt{P}_{\theta'}(Z_i^{(0)} = z; Z_i^{(R)} = z')| \tag{33}$$

$$= \max_{z,z'} |[\mathbf{K}_{i,\theta}^R]_{z',z} - [\mathbf{K}_{i,\theta'}^R]_{z',z}| \tag{34}$$

$$= \max_{z,z'} |\mathbf{e}_{z'}^\top (\mathbf{K}_{i,\theta}^R - \mathbf{K}_{i,\theta'} \mathbf{K}_{i,\theta}^{R-1} + \mathbf{K}_{i,\theta'} \mathbf{K}_{i,\theta}^{R-1} - \mathbf{K}_{i,\theta'}^R) \mathbf{e}_z| \tag{35}$$

$$\leq \max_{z,z'} |\mathbf{e}_{z'}^\top (\mathbf{K}_{i,\theta} - \mathbf{K}_{i,\theta'}) \mathbf{K}_{i,\theta}^{R-1} \mathbf{e}_z| + |\mathbf{e}_{z'}^\top \mathbf{K}_{i,\theta'} (\mathbf{K}_{i,\theta}^{R-1} - \mathbf{K}_{i,\theta'}^{R-1}) \mathbf{e}_z| \tag{36}$$

$$\overset{(i)}{\leq} \max_{z,z'} \|\mathbf{e}_{z'}^\top (\mathbf{K}_{i,\theta} - \mathbf{K}_{i,\theta'})\|_\infty \cdot \|\mathbf{K}_{i,\theta}^{R-1} \mathbf{e}_z\|_1 + \|\mathbf{e}_{z'}^\top \mathbf{K}_{i,\theta'}\|_1 \cdot \|(\mathbf{K}_{i,\theta}^{R-1} - \mathbf{K}_{i,\theta'}^{R-1}) \mathbf{e}_z\|_\infty \tag{37}$$

$$\overset{(ii)}{\leq} \max_{z,z'} \|\mathbf{e}_{z'}^\top (\mathbf{K}_{i,\theta} - \mathbf{K}_{i,\theta'})\|_\infty \overset{(iii)}{+} 2 \cdot |[\mathbf{K}_{i,\theta}^{R-1}]_{z',z} - [\mathbf{K}_{i,\theta'}^{R-1}]_{z',z}| \tag{38}$$

$$\overset{(27)}{\leq} 2L_P \cdot \exp(2c^2\beta) \cdot \beta \cdot \|\theta - \theta'\| + 2 \cdot |[\mathbf{K}_{i,\theta}^{R-1}]_{z',z} - [\mathbf{K}_{i,\theta'}^{R-1}]_{z',z}| \tag{39}$$

$$\leq (1 + 2 + ... + 2^{R-1}) \cdot 2L_P \cdot \exp(2c^2\beta) \cdot \beta \cdot \|\theta - \theta'\| \tag{40}$$

$$= (2^R - 1) \cdot 2L_P \cdot \exp(2c^2\beta) \cdot \beta \cdot \|\theta - \theta'\| \tag{41}$$

where (i) uses Hölder's inequality for the norm pair $(\|\cdot\|_1, \|\cdot\|_\infty)$, (ii) uses the fact that $\mathbf{K}_{i,\theta}^{R-1}$ is a column stochastic matrix and (iii) uses the inequality $\|\mathbf{e}_{z'}^\top \mathbf{K}_{i,\theta'}\|_1 \leq 2$ because

$$\|\mathbf{e}_{z'}^\top \mathbf{K}_{i,\theta'}\|_1 = \sum_{z \in [m_{\text{neg}}]} [\mathbf{K}_{i,\theta}]_{z',z} \tag{42}$$

$$\overset{(28)}{=} \sum_{z \in [m_{\text{neg}}] \setminus \{z'\}} \frac{1}{m_{\text{neg}}} \min\{1, Q_{x_i,\theta}(z,z')\} \tag{43}$$

$$+ \frac{1}{m_{\text{neg}}} + \frac{1}{m_{\text{neg}}} \sum_{j \in [m_{\text{neg}}] \setminus \{z'\}} (1 - \min\{1, Q_{x_i,\theta}(z',j)\}) \tag{44}$$

$$\leq \frac{m_{\text{neg}} - 1}{m_{\text{neg}}} + \frac{1}{m_{\text{neg}}} + \frac{m_{\text{neg}} - 1}{m_{\text{neg}}} \leq 2 \tag{45}$$

Finally, in the Markov chain $\tilde{\boldsymbol{\xi}}_0 \to \tilde{\boldsymbol{\xi}}_1 \to \cdots$, since only B state variables $\{Z_{i_k^{(t)}}\}_{k=1}^B$ are active or updated in each transition, we have

$$\mathtt{P}_\theta(\boldsymbol{\xi}; \boldsymbol{\xi}') = \begin{cases} 0 & \text{if } \Delta_\xi = \|\xi - \xi'\|_0 > B, \\ \binom{m}{B}^{-1} \cdot \binom{m - \Delta_\xi}{B - \Delta_\xi} \cdot \prod_{k=1}^B \mathtt{P}_\theta(Z_{i_k^{(t)}}^{(0)}; Z_{i_k^{(t)}}^{(R)}) & \text{otherwise.} \end{cases} \tag{46}$$

where $\binom{m}{B}^{-1} \binom{m - \Delta_\xi}{B - \Delta_\xi}$ is the probability that $j \in \{i_k^{(t)}\}_{k=1}^B$ for j s.t. $\xi(j) \neq \xi'(j)$. To simplify notations, suppose $Z_{i_k^{(t)}}^{(0)} = z_{t,k}$ and $Z_{i_k^{(t)}}^{(R)} = z'_{t,k}$, then

$$|\mathtt{P}_\theta(\xi; \xi') - \mathtt{P}_{\theta'}(\xi; \xi')| = \binom{m}{B}^{-1} \binom{m - \Delta_\xi}{B - \Delta_\xi} \left| \prod_{k=1}^B \mathtt{P}_\theta(Z_{i_k^{(t)}}^{(0)}; Z_{i_k^{(t)}}^{(R)}) - \prod_{k=1}^B \mathtt{P}_{\theta'}(Z_{i_k^{(t)}}^{(0)}; Z_{i_k^{(t)}}^{(R)}) \right| \tag{47}$$

$$= \binom{m}{B}^{-1} \binom{m - \Delta_\xi}{B - \Delta_\xi} \left| \prod_{k=1}^B [\mathbf{K}_{i,\theta}^R]_{z_{t,k}, z'_{t,k}} - \prod_{k=1}^B [\mathbf{K}_{i,\theta'}^R]_{z_{t,k}, z'_{t,k}} \right| \tag{48}$$

$$= \binom{m}{B}^{-1} \binom{m-\Delta_\xi}{B-\Delta_\xi} \Big| \prod_{k=1}^{B} [\mathbf{K}^R_{i,\theta}]_{z_{t,k},z'_{t,k}} - [\mathbf{K}^R_{i,\theta'}]_{z_{t,1},z'_{t,1}} \prod_{k=2}^{B} [\mathbf{K}^R_{i,\theta}]_{z_{t,k},z'_{t,k}} \tag{49}$$

$$+ [\mathbf{K}^R_{i,\theta'}]_{z_{t,1},z'_{t,1}} \prod_{k=2}^{B} [\mathbf{K}^R_{i,\theta}]_{z_{t,k},z'_{t,k}} - \prod_{k=1}^{B} [\mathbf{K}^R_{i,\theta'}]_{z_{t,k},z'_{t,k}} \Big| \tag{50}$$

$$\overset{(41)}{\leq} \binom{m}{B}^{-1} \binom{m-\Delta_\xi}{B-\Delta_\xi} B \cdot 2^{R+1} \cdot L_P \cdot \exp(2c^2\beta) \cdot \beta \cdot \|\theta - \theta'\| \tag{51}$$

The proof is concluded by taking maximum over $0 \leq \Delta_\xi \leq B$. □

C. Proof of Theorem 3.7

Our idea is to apply Theorem 2 of (Karimi et al., 2019) which shows the convergence of biased stochastic approximation scheme with Markovian noise such as (10). In particular, we shall take the Lyapunov function therein as our global contrastive loss, i.e., $V(\theta) = \mathcal{L}(\theta)$. We proceed by verifying the required assumptions.

Verifying A1-A2 of (Karimi et al., 2019) Using Lemma 3.2, we observe that the mean field of the stochastic update in (10), i.e., the expected value of $\mathcal{H}(\xi;\theta)$ when ξ is drawn from the stationary distribution of the Markov chain induced by $\mathtt{P}_\theta$, coincides with the gradient of $\mathcal{L}(\theta)$, i.e., $h(\theta) = \nabla\mathcal{L}(\theta)$. As such, A1-A2 of (Karimi et al., 2019) are satisfied with $c_0 = d_0 = 0$, $c_1 = d_1 = 1$.

Verifying A3 of (Karimi et al., 2019) For this, we need to show that $\nabla\mathcal{L}(\theta)$ is Lipschitz continuous. We observe that a stronger condition holds as the stochastic gradient map is Lipschitz w.r.t. θ. For any sample x, y, z and any $\theta, \theta' \in \mathbb{R}^p$, we have

$$\begin{aligned} &\| -\beta\nabla_\theta(\phi(x;\theta)^\top\psi(y;\theta)) + \beta\nabla_\theta(\phi(x;\theta)^\top\psi(z;\theta)) - [-\beta\nabla_\theta(\phi(x;\theta')^\top\psi(y;\theta')) + \beta\nabla_\theta(\phi(x;\theta')^\top\psi(z;\theta'))]\| \\ &\leq \beta\|\nabla_\theta(\phi(x;\theta)^\top\psi(y;\theta)) - \nabla_\theta(\phi(x;\theta')^\top\psi(y;\theta'))\| + \beta\|\nabla_\theta(\phi(x;\theta)^\top\psi(z;\theta)) - \nabla_\theta(\phi(x;\theta')^\top\psi(z;\theta'))\| \\ &\overset{(\text{Assm. } 3.5)}{\leq} 2\beta L_H\|\theta - \theta'\| \end{aligned} \tag{52}$$

Verifying A5-6 of (Karimi et al., 2019) For these assumptions, we observe that A12-A14 of (Karimi et al., 2019) can be satisfied with the constants

$$\bar{L}_P = 2^{R+1}BL_P \cdot \exp(2c^2\beta) \cdot \beta, \quad \bar{L}_H = 2\beta L_H, \quad \bar{\rho} = 1 - \frac{BR}{2mm_{\text{neg}}\exp(2c^2\beta)}, \quad K_P = 1, \quad \bar{\sigma} \overset{(56)}{=} 2\beta\sigma \tag{53}$$

Applying Lemma 7 of (Karimi et al., 2019) shows that A5-A6 can be satisfied with

$$L^{(0)}_{PH} = \frac{2\beta\sigma\bar{\rho}}{BRm^{-1}m^{-1}_{\text{neg}}\exp(-2c^2\beta)/2}, \quad L^{(1)}_{PH} = \frac{6\cdot 2^{R+1}B\exp(2c^2\beta)\beta^2\sigma L_P}{(BRm^{-1}m^{-1}_{\text{neg}}\exp(-2c^2\beta)/2)^2} + \frac{2\beta L_H}{BRm^{-1}m^{-1}_{\text{neg}}\exp(-2c^2\beta)/2} \tag{54}$$

Verifying A7 of (Karimi et al., 2019) The stochastic gradient used in (10) has a uniformly bounded error from its mean field as

$$\begin{aligned} &\|\mathcal{H}(\xi_{t+1};\theta_t) - \nabla\mathcal{L}(\theta_t)\| \\ &\leq \beta\left\|\frac{1}{B}\sum_{\ell=1}^{B} H(x^{(t)}_{i_\ell}, y^{(t)}_{i_\ell};\theta_t) - \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{pos}}}[\nabla_\theta(\phi(x;\theta)^\top\psi(y;\theta))]\right\| \\ &\quad + \beta\left\|\frac{1}{B(B-P)}\sum_{k=1}^{B}\sum_{r=P}^{R-1} H(x_{i^{(t)}_k}, \tilde{Z}^{(r)}_{i^{(t)}_k};\theta_t) - \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{pos}}}\mathbb{E}_{z\sim\mathcal{P}_{x,\theta}}[\nabla_\theta(\phi(x;\theta)^\top\psi(z;\theta))]\right\| \\ &\leq \frac{\beta}{B}\sum_{\ell=1}^{B}\|H(x^{(t)}_{i_\ell}, y^{(t)}_{i_\ell};\theta_t) - \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{pos}}}[\nabla_\theta(\phi(x;\theta)^\top\psi(y;\theta))]\| \end{aligned} \tag{55}$$

$$+ \frac{\beta}{B(B-P)} \sum_{k=1}^{B} \sum_{r=P}^{B-1} \|H(x_{i_k^{(t)}}, \tilde{Z}_{i_k^{(t)}}^{(r)}; \theta_t) - \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{pos}}} \mathbb{E}_{z\sim\mathcal{P}_{x,\theta}}[\nabla_\theta(\phi(x;\theta)^\top \psi(z;\theta))]\|$$

$$\overset{\text{(Assm. 3.6)}}{\leq} 2\beta\sigma \tag{56}$$

Convergence of (10) Upon verifying A1-A3, A5-A7 of (Karimi et al., 2019), we can apply Theorem 2 therein to analyze the convergence of (10). In particular, we choose a constant step size γ satisfying $\gamma \leq \frac{1}{2(2\beta L_H + C_h)}$. For any $T \geq 1$ and the randomly drawn $t \sim \text{Uniform}([0, ..., T-1])$, we have

$$\mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] = \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] \tag{57}$$

$$\leq \frac{\mathbb{E}[\mathcal{L}(\theta_0) - \mathcal{L}(\theta_T)] + 3L_{PH}^{(0)} + [8\beta^3 L_H \sigma^2 + 2L_{PH}^{(1)}\beta\sigma + 2\beta L_H L_{PH}^{(0)}(1+2\beta\sigma)](\gamma^2 T)}{\gamma T/2} \tag{58}$$

where $C_h = L_{PH}^{(1)}(1+\sigma) + L_{PH}^{(0)}(\beta L_H + 1)$. Simplifying and rearranging terms leads to the conclusion.

Estimating $\mathbb{E}[\mathcal{L}(\theta_0) - \mathcal{L}(\theta_T)]$ We observe that by Assumption 3.1, for any $\theta \in \mathbb{R}^p$,

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\text{pos}}} \left[-\beta\, \phi(x;\theta)^\top \psi(y;\theta)\right] + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\text{pos}}} \left[\log \sum_{z\in\mathbf{D}_{\text{neg}}(x)} \exp(\beta\, \phi(x;\theta)^\top \psi(z;\theta))\right] \tag{59}$$

$$\leq \beta + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\text{pos}}} \left[\log \sum_{z\in\mathbf{D}_{\text{neg}}(x)} \exp(\beta)\right]$$

By a similar argument,

$$\mathcal{L}(\theta) \geq -\beta + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\text{pos}}} \left[\log \sum_{z\in\mathbf{D}_{\text{neg}}(x)} \exp(-\beta)\right] \tag{60}$$

Therefore,

$$\mathbb{E}[\mathcal{L}(\theta_0) - \mathcal{L}(\theta_T)] \leq 2\beta + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\text{pos}}} \left[\log \sum_{z\in\mathbf{D}_{\text{neg}}(x)} \exp(\beta) - \log \sum_{z\in\mathbf{D}_{\text{neg}}(x)} \exp(-\beta)\right] \tag{61}$$

$$= 2\beta + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\text{pos}}} \left[\log\left(\frac{\sum_{z\in\mathbf{D}_{\text{neg}}(x)} \exp(\beta)}{\sum_{z\in\mathbf{D}_{\text{neg}}(x)} \exp(-\beta)}\right)\right]$$

$$= 2\beta + \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}_{\text{pos}}} [\log\exp(2\beta)] = 4\beta.$$

□

D. Experiment Details

D.1. Details of Dataset and Hyperparameters

In Table 2, we list the important attributes of datasets we used in the experiment section.

In Table 3, we list the hyperparameter values adopted in our experiments.

D.2. Run-Time Complexity

Since the computational complexity differs among algorithms, we provide a performance-time plot in Figure 8 to compare the time complexity. Note that in this setup, negative cache algorithm uses four Tesla T4 GPUs for training and refreshing the negative cache while the other algorithms run on one Tesla T4 GPU.

Dataset	# of Pos. Pairs m	# of Neg. Samples m_{neg}	Image Crop Size
STL-10	100,000 (inf. aug)	100,000 (inf. aug)	96×96
Imagenet-100	124,689 (inf. aug)	124,689 (inf. aug)	224×224

Table 2. Datasets attributes.

Dataset	Model	Inverse Temp. β	Batch Size b	Learning Rate γ	Feature Dim. d	Weight Decay	Cache Refresh ρ (Negative Cache)	Burn-in Steps P (EMC^2)
STL-10	Resnet-18	14.28	32	10^{-4}	512	10^{-4}	0.01024	31
STL-10	Resnet-18	5	256	10^{-3}	512	10^{-4}	0.1	255
Imagenet-100	Resnet-50	14.28	256	1.2	128	10^{-6}	-	255
STL-10 subset	Resnet-18	5	4	10^{-3}	512	10^{-4}	0.1	3

Table 3. Hyperparameter values of the experiments.

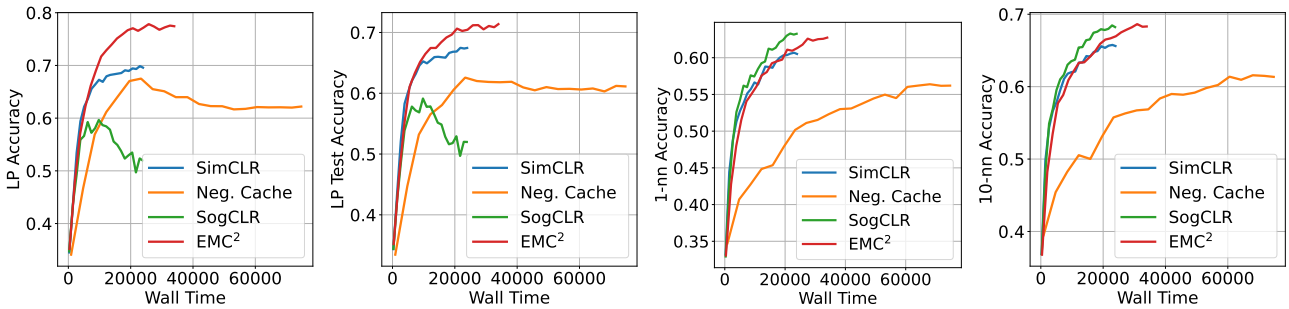

Figure 8. Training 100 epochs on STL-10 with ResNet-18 using batch size $b = 32$. Horizontal axis is relative to the wall-clock training time in seconds.