Posterior Sampling on Simsiam: Rethinking Optimization in Siamese Self-Supervised Learning

Daniel de Mello
Dept. of Computer Science
Purdue University
West Lafayette, IN, 47906
`ddemello@purdue.edu`

Ruqi Zhang
Dept. of Computer Science
Purdue University
West Lafayette, IN, 47906
`ruqiz@purdue.edu`

Bruno Ribeiro
Dept. of Computer Science
Purdue University
West Lafayette, IN, 47906
`ribeirob@purdue.edu`

Abstract

Chen and He [2020] states that self-supervised pre-training can be performed without contrastive learning (CL) (i.e., using negative pairs). Rather, the proposed approach (SimSiam) merely maximizes the similarity between two transformations of the same example. Interestingly, even though a global optimum for this task is to collapse SimSiam into a constant function ignoring input, Chen and He [2020] argues that, in practice, the training converges to non-global optima yielding useful representations of the input. A key component is a stop-gradient (SG) operation which, if not used, causes SimSiam to quickly collapse to the global optimum. In this work, we investigate whether SG is genuinely indispensable or if satisfactory outcomes can be achieved by better exploring the loss landscape. Namely, we keep the loss landscape intact by not changing SimSiam's architecture, and explore it with SGHMC Chen et al. [2014], a sampling method known for efficiently covering distant regions of the posterior distribution. Our empirical finding is that the proposed samples of the posterior never reach collapsed points for properly chosen step-sizes of SGHMC, indicating a large room for future optimization methods other than SG that could avoid collapse. Although SGHMC turns out not as effective as SG for improving accuracy in the downstream task, we believe our results beg more investigation about the actual necessity of SG.

1 Introduction

In recent years, self-supervised representation (SSL) learning research has obtained breakthrough results with contrastive learning (CL) methods based on **instance discrimination** Chen et al. [2020], Wu et al. [2018], He et al. [2019], Jaiswal et al. [2020], enabling SOTA results in downstream evaluation tasks, like Imagenet classification. Said methods discover a feature mapping $\phi : \mathbb{R}^n \to \mathbb{R}^d$ for an instance $\mathbf{x}_i \in \mathbb{R}^n$, such that $\phi(T_a(\mathbf{x}_i))$ is similar to $\phi(T_b(\mathbf{x}_i))$, with $T_{a,b} : \mathbb{R}^n \to \mathbb{R}^n$ representing an *augmented view* of $\mathbf{x}_i$, and $T_{a,b} \sim \mathcal{T}(T)$, where $\mathcal{T}(T)$ defines a distribution over transformations, with prior assumption that $T_{a,b}$ preserve the overall semantics of $\mathbf{x}_i$. Such model is often referred to as **siamese**, as it learns over 2 parallel projections with the same encoder. One of the issues in obtaining ϕ is avoiding a **collapsed representation**, where $\phi(.)$ trivially learns a constant mapping for all $\mathbf{x}_i$, thus obtaining a useless representation. To avoid this, these methods employ **negative pairs** for learning $\phi(.)$: the loss pushes apart $\phi(T_a(\mathbf{x}_i))$ and $\phi(T_b(\mathbf{x}_j))$, for $i \neq j$.

Surprisingly, Grill et al. [2020], Chen and He [2020] employed the same kind of siamese encoder structure as CL methods like SimCLR, but using only positive samples for similarity. It was later found by Chen and He [2020] that the fundamental components of these methods were a "prediction head" network $p : \mathbb{R}^d \to \mathbb{R}^d$ mapping from the encoding obtained with one mapping (source) to the encoding in the parallel mapping (target), with the similarity measurement performed between the output of $p(.)$ and the target mapping. The other key aspect was using a stop-gradient (SG) operation in the target encoding, thus stopping the gradient flux for the encoding of that respective transformation. Precisely, for source and target encodings $\phi(T_a(\mathbf{x}_i)) = \mathbf{Z}_a$ and $\phi(T_b(\mathbf{x}_i)) = \mathbf{Z}_b$, respectively, the similarity loss $\mathcal{L}_{sim}$ for i-th instance is computed as $\mathcal{L}_{sim} = -p(\mathbf{Z}_a) \cdot \mathrm{sg}(\mathbf{Z}_b)$, where sg marks the SG operation on the target, and both $\mathbf{Z}_a$ and $\mathbf{Z}_b$ are l^2 normalized.

Some recent works Tian et al. [2021], Zhang et al. [2022], Wen and Li [2022] have tried to explain how collapse avoidance is possible in SimSiam. Specifically, Zhang et al. [2022] studied the decomposition of the gradient of SimSiam loss without predictor and why removing it leads to collapse. The negative of this gradient can be derived as $-\frac{\partial(-\mathbf{Z}_a \mathrm{sg}(\mathbf{Z}_b))}{\partial \mathbf{Z}_a} = \mathbf{Z}_b$. Then, this term can be decomposed into a center vector $\mathbf{o}_z$ and a residual vector $\mathbf{r}_b$, such that $\mathbf{o}_z := \mathbb{E}[\mathbf{Z}_a] = \mathbb{E}[\mathbf{Z}_b]$ (estimated by mean over minibatch) and $\mathbf{r}_b := \mathbf{Z}_b - \mathbf{o}_z$. In short, the authors conjectured (with empirical demonstration) that methods which introduce negative samples or, in the case of SimSiam, asymmetrical architecture, add an extra gradient component that contains $\mathbf{o}_z$ in a negative direction, thus decreasing the amount of $\mathbf{o}_z$ in $\mathbf{Z}_b$ (de-centering effect). Collapse happens precisely when the ratio of $\mathbf{o}_z$ in $\mathbf{Z}_b$, given by $||\mathbf{o}_z||/||\mathbf{Z}_b||$, approaches 1 (recall $\mathbf{Z}_b$ is l^2 normalized). So, a gradient that goes in the opposite direction of $\mathbf{o}_z$ would prevent it. Another even more recent work was Wen and Li [2022], which provided an alternative explanation via a statistical analysis of the feature learning dynamics of the prediction head. In short, their analysis identifies two effects that help to avoid collapse. 1) the substitution effect where the prediction head decreases the learning speed of a strong feature in other neurons after it is learned in a single neuron; 2) the acceleration effect where the strong features accelerate the learning of weaker features in substituted neurons. Our work contrasts with Tian et al. [2021], Zhang et al. [2022], Wen and Li [2022] in that we focus our study on the SG operation alone.

Our goal in this work is to isolate the effects of *optimization*, and for this we keep the predictor intact to preserve the original loss landscape in SimSiam. The SG operation determines which minimum SimSiam lands into, while introducing (removing) the predictor introduces (removes) particular minimum regions. By modifying only SG, we can investigate its role as an optimization component. Our method is to replace the regular optimization by posterior sampling, which can better characterize the full loss landscape and circumvent potential optimization biases governing the optimization in SimSiam without SG. We choose Stochastic Gradient Hamiltonian Monte-Carlo (SGHMC) Chen et al. [2014] as a sampling method, which is optimal for complex tasks in high-dimensions, where running standard HMC Brooks et al. [2011] is prohibitively expensive. We empirically reveal that SGHMC is sufficient to prevent collapse, although it's not effective for raising the accuracy in the downstream task. We do not claim that SGHMC should replace SG, but, instead, that posterior sampling being able to avoid collapse should encourage future investigation on whether SG is really an essential component of SimSiam and related SSL methods. Our contributions are the folowing:

- We perform posterior sampling on SimSiam with SGHMC, which to the best of our knowledge is the first attempt of posterior sampling over SSL methods similar to SimSiam, adapting the similarity loss to be modeled as a likelihood, necessary for posterior sampling.
- We perform experiments under several sampling scenarios of SimSiam without SG training on CIFAR-10, demonstrating that most proposed samples never reach collapsing regions.

2 Method

2.1 Stochastic Gradient Hamiltonian Monte-Carlo

HMC is a powerful MCMC sampling technique that models sampling from a target distribution $\mathcal{P}$ (typically a posterior) as a system governed by Hamiltonian dynamics. For a model parameterized by $\boldsymbol{W}$, points in $\mathcal{P}$ (the parameter space) are modeled with a potential energy $U(\boldsymbol{W})$, and "velocities" auxiliary variables $\boldsymbol{\Phi}$ are modeled as kinetic energy. This simulation enables proposals of distant states, with high acceptance probability after a Metropolis-Hastings correction. However, each HMC samples requires computing gradients for the potential energy over the entire dataset, which is prohibitively expensive for complex distributions. Chen et al. [2014] corrects convergence issues introduced by the noise in stochastic gradients for HMC, enabling a cheaper sampling cost. SGHMC follows the system of stochastic differential equations in Eq. 1:

$$\begin{cases} d\boldsymbol{W} = \boldsymbol{M}^{-1}\boldsymbol{\Phi} dt \\ d\boldsymbol{\Phi} = -\nabla U(\boldsymbol{W})dt - \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{\Phi} dt + \mathcal{N}(0, 2(\boldsymbol{C} - \hat{\boldsymbol{B}})dt) + \mathcal{N}(0, 2\boldsymbol{B}dt) \end{cases} \tag{1}$$

where $\boldsymbol{M}$ is the "mass" matrix (often identity) in standard HMC modeling the correlations directions of movement, $\boldsymbol{B}$ is the gradient noise, $\hat{\boldsymbol{B}}$ an arbitrary estimate for $\boldsymbol{B}$ (typically 0), $\boldsymbol{C}$ a friction term for correcting the noise. Eq. 1 can be discretized by the leapfrog numerical integrator, with $\boldsymbol{B}dt \rightarrow 0$ as step sizes $\epsilon \rightarrow 0$, and $\boldsymbol{C}$ dominating the updates. Our implementation of SGHMC uses the "kick-drift-kick" version of leapfrog, and is summarized in Algorithm 1 in the Appendix.

2.2 Adapting SimSiam to SGHMC

Our goal is to employ SGHMC to generate samples of the posterior defined over the parameters of SimSiam (parameters for both the encoder and predictor networks). In typical Bayesian inference methods, like supervised classification, the required likelihood is present in the loss via negative log-likelihood. However, this isn't the case here. Instead, SimSiam seeks to maximize the cosine similarity by minimizing a negative similarity loss. Therefore, we need a method to model our similarity loss as likelihood. Once modeled, we can then use it as the Potential energy $U(\mathbf{W})$ for SGHMC. Besides, we need to ensure that $U(\mathbf{W})$ is always positive, as the notion of energy relies on this assumption. Our solution is to enforce that the $\mathcal{L}_{SimSiam}$ behaves as a negative of a probability, that is, ranging from [-1,0] instead of [-1, 1], while still being computed as negative cosine similarity. SimSiam's symmetrized loss is defined as $\mathcal{L}_{SimSiam} = -1/2\left(\boldsymbol{P}_a \cdot \mathrm{sg}\left(\boldsymbol{Z}_b\right) + \boldsymbol{P}_b \cdot \mathrm{sg}\left(\boldsymbol{Z}_a\right)\right)$, And to enforce it always stays in [-1,0], we convert it to a slightly different version:

$$\mathcal{L}_{SimSiamAbs} = -\frac{1}{2}\left(\left|\boldsymbol{P}_a \cdot \mathrm{sg}\left(\boldsymbol{Z}_b\right)\right| + \left|\boldsymbol{P}_b \cdot \mathrm{sg}\left(\boldsymbol{Z}_a\right)\right|\right) \tag{2}$$

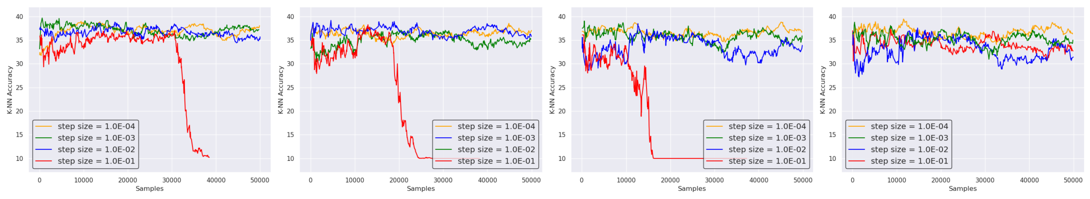
That is, we take the absolute value of each pair of cosine similarities, forcing each to lie in [0,1], and hence the negative to lie in [-1, 0]. We have empirically observed that the cosine similarity between two projections of SimSiam rarely stays negative (typically occurring in the initial stages of the training). Which means that, in practice, $\mathcal{L}_{SimSiamAbs}$ is almost the same as $\mathcal{L}_{SimSiam}$. Thus, with $\mathcal{L}_{SimSiamAbs}$ guaranteed to be in $[-1, 0]$, we treat it as the negative of a probability in a negative log-likelihood scenario. With this view, we can define our potential energy function as $U(\mathbf{W}) = -\log(-\mathcal{L}_{SimSiamAbs})$. And it is clear our model of $U(\mathbf{W})$ always stays positive. Intuitively, regions with small $U(\mathbf{W})$ behave much like in typical HMC sampling, where a small potential energy is associated with small loss. Specifically, in our case, a small $U(\mathbf{W})$ is associated with a small $\mathcal{L}_{SimSiamAbs}$, with the global minimum $U(\mathbf{W}) = 0$ occurring at $\mathcal{L}_{SimSiamAbs} = -1$. For a classification scenario on some data D, $U(\mathbf{W}) = 0$ would be result from $-P(D|\mathbf{W}) = -1$, which is analogous to our problem, as the classification loss minimizes the negative likelihood, while our loss minimizes the negative similarity.

3 Results

Our experimental setup for this section is described as follows. For dataset, we use the 50000 training examples of CIFAR-10 for training SimSiam's SSL task. The 10000 examples test set is used for measuring k-NN-accuracy (with k-NN being trained also with the 50000 training set). The transformations applied to input pairs are the same as described in Chen and He [2020]. We sample minibatch sizes of 512. For architecture, we use a vanilla CNN architecture for SimSiam's encoder, while the prediction head is modeled by an MLP with 1 hidden layer (see appendix for more details). For experiments without SG, we simply do not use $\mathrm{sg}(.)$ in Equation 2.

3.1 Main Results

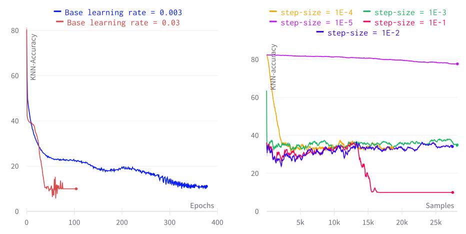
In Figure 1 we depict k-NN accuracy (k=200) for 16 experiments of SGHMC without the SG operation. Each experiment varies the number of leapfrogs m choosing from $\{1, 2, 3, 5\}$ and step-size ϵ choosing from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. For all experiments, we used an identity matrix for the "mass" covariance matrix $\boldsymbol{M}$, and a friction covariance matrix $\boldsymbol{C}$ such that $\boldsymbol{M}^{-1}\boldsymbol{C}\epsilon = 0.01\boldsymbol{I}$ (a setting adopted by the authors of SGHMC for classification with bayesian neural networks). We observe here that all runs avoid collapse, except the ones using $\epsilon = 0.1$ (an unrealistic high ϵ), whose collapse can be inferred by the k-NN accuracy reaching 10 % (no better than random). Even then, these few occurrences of collapse only happened after more than 10000 samples, indicating SGHMC was still able to explore extensive areas of the posterior without finding collapsed points.



(a) 1 leapfrog per sample (b) 2 leapfrogs per sample (c) 3 leapfrogs per sample (d) 5 leapfrogs per sample

Figure 1: k-NN Accuracy for each SimSiam parameter sample obtained with SGHMC, across 50 k sample proposals, using varying number of leapfrogs: 1 (a), 2 (b), 3 (c), 5 (d), with each subfigure depicting 4 plots for different step sizes.

3.2 SGHMC on pretrained SimSiam



(a) SimSiam w/o SG (b) SGHMC w/o SG

Figure 2: k-NN accuracy measurements considering an already pretrained SimSiam model for 800 epochs. In a) we train SimSiam for additional epochs, but removing SG, and for base learning rates $\eta \in \{3 \cdot 10^{-2}, 3 \cdot 10^{-3}\}$. In b), we run SGHMC (3 leapfrogs) with SG removed, considering step-size $\epsilon \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

In Figure 2, we consider an already pretrained SimSiam model following the standard setting (original hyperparameters and SG enabled), and measure how performing SGHMC with SG removed affects the accuracy, as well as how the standard SimSiam training after disabling SG would perform. In Figure 2 a) we can observe how the k-NN accuracy of SimSiam with SG removed quickly collapses from the optimal pretrained state at 80 % accuracy, remaining stationary around the usual initial state of 40 %, and then fully collapsing to the completely random state of 10%. The sharp decline from 80 % to 40 %, which took place in the very first epoch, strongly suggests how volatile the optimum loss region for high accuracy obtained by standard SimSiam (it should be noted that SimSiam uses warmup of 10 epochs, linearly increasing from 0 to the base learning rate, *i.e.*, for a base learning rate $\eta = 0.03$, the learning rate in the first epoch starts at 0.003). In Figure 2 b), our experiments with SGHMC show the proposed samples quickly decaying to the 40 % accuracy region, except when using exceptionally small step-size $\epsilon = 10^{-6}$. However, all step sizes ϵ were sufficient to resist total collapse for much longer than SimSiam with SG removed in Figure 2 a) under all learning rates η, with the only collapse for SGHMC occurring around 13k samples for the unusually high step-size $\epsilon = 10^{-2}$. For comparison, each epoch in SimSiam contains around 100 gradient updates, while each step in 3-leapfrogs SGHMC contains 5 gradient updates. So, SimSiam without SG reached total collapse at around 4000 gradient updates (40 epochs), while SGHMC with the highest step-size did it only after around 75000 gradient updates.

4 Conclusion

SGHMC is known for its capacity of making high likelihood proposals that are far apart from each other, thus making it one of the optimal sampling approaches for navigating extensive distances in the loss landscape. In this work we have demonstrated that posterior sampling over SimSiam, performed by SGHMC, shows a low propensity of collapse, under varying hyperparameters settings. Moreover, we've shown that SimSiam with SG removed collapses much faster than any situation where SGHMC is used. Our results provide insights about the possibility of moving around the loss landscapce by simultaneously avoiding collapse and removing SG.

References

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, editors. *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, may 2011. doi: 10.1201/b10905. URL `https://doi.org/10.1201%2Fb10905`.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691, Bejing, China, 22–24 Jun 2014. PMLR.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/chen20j.html`.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning, 2020. URL `https://arxiv.org/abs/2011.10566`.

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. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2019. URL `http://arxiv.org/abs/1911.05722`. cite arxiv:1911.05722Comment: CVPR 2020 camera-ready. Code: https://github.com/facebookresearch/moco.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10268–10278. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/tian21a.html`.

Zixin Wen and Yuanzhi Li. The mechanism of prediction head in non-contrastive self-supervised learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=d-kvI4YdNu`.

Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Chaoning Zhang, Kang Zhang, Chenshuang Zhang, Trung X. Pham, Chang D. Yoo, and In So Kweon. How does simsiam avoid collapse without negative samples? a unified understanding with self-supervised contrastive learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=bwq6O4Cwdl`.

A Supplementary Material

A.1 SGHMC algorithm

Our implementation of SGHMC algorithm uses the “kick-drift-kick” version of leapfrog integrator, as described by Algorithm 1.

Algorithm 1 Single Step Sampling of Stochastic Gradient Hamiltonian Monte-Carlo

```
Require: Previous sample $\boldsymbol{W}_t$, Size of Leapfrog Step $\epsilon$, Number of Leapfrog Steps $L$, “Mass” matrix
    $\boldsymbol{M}$, Friction $\boldsymbol{C}$
Ensure: New sample $\boldsymbol{W}_{t+1}$
  $\boldsymbol{\Phi}_0 \sim \mathcal{N}(0, \boldsymbol{M})$
  $\boldsymbol{X}_0 = \boldsymbol{W}_t$
  for $l = 0, \cdots, L-1$ do
    $\boldsymbol{\Phi}_{\left(l+\frac{1}{2}\right)\epsilon} = (1 - \epsilon \boldsymbol{C}\boldsymbol{M}^{-1})\boldsymbol{\Phi}_{l\epsilon} - \frac{\epsilon}{2} \frac{\partial U(\boldsymbol{W})}{\partial \boldsymbol{W}}\Big|_{\boldsymbol{W}=\boldsymbol{X}_{l\epsilon}} + \mathcal{N}(0, 2\boldsymbol{C}\epsilon)$
    $\boldsymbol{X}_{(l+1)\epsilon} = \boldsymbol{X}_{l\epsilon} + \epsilon \boldsymbol{M}^{-1}\boldsymbol{\Phi}_{\left(l+\frac{1}{2}\right)\epsilon}$
    $\boldsymbol{\Phi}_{(l+1)\epsilon} = (1 - \epsilon \boldsymbol{C}\boldsymbol{M}^{-1})\boldsymbol{\Phi}_{\left(l+\frac{1}{2}\right)\epsilon} - \frac{\epsilon}{2} \frac{\partial U(\boldsymbol{W})}{\partial \boldsymbol{W}}\Big|_{\boldsymbol{W}=\boldsymbol{X}_{(l+1)\epsilon}} + \mathcal{N}(0, 2\boldsymbol{C}\epsilon)$
  end for
  $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t$
```

A.2 Architecture details

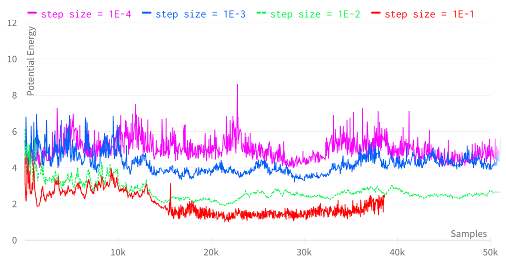
SimSiam’s **Encoder + projector**:

- **Conv1**: 3x32, 3x3 kernel, stride 1, padding 1, ReLU activation.
- **Conv2**: 32x64, 3x3 kernel, stride 1, padding 1, ReLU activation.
- **MaxPool1**: 2x2 kernel, stride 2, output: 64x16x16.
- **BatchNorm1**: 64 channels.
- **Conv3**: 64x128, 3x3 kernel, stride 1, padding 1, ReLU activation.
- **Conv4**: 128x128, 3x3 kernel, stride 1, padding 1, ReLU activation.
- **MaxPool2**: 2x2 kernel, stride 2, output: 128x8x8.
- **BatchNorm2**: 128 channels.
- **Conv5**: 128x256, 3x3 kernel, stride 1, padding 1, ReLU activation.
- **Conv6**: 256x256, 3x3 kernel, stride 1, padding 1, ReLU activation.
- **MaxPool3**: 2x2 kernel, stride 2, output: 256x4x4.
- **BatchNorm3**: 256 channels.
- **Fully Connected (Linear)**: 256x4x4 input, 128 output, ReLU activation.

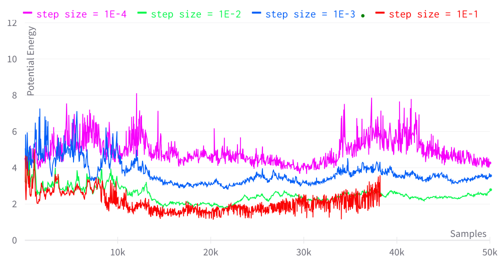
SimSiam’s **Predictor**:

- **Linear**: 128x64.
- **BatchNorm**: 64 channels + ReLU activation.
- **Linear**: 64x128.

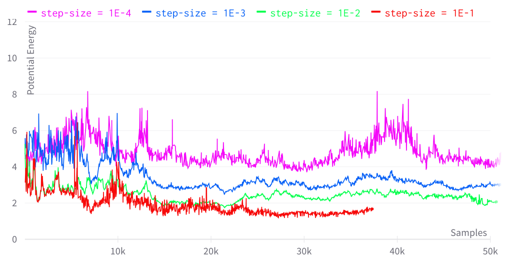
A.3 Potential Energy Measurements for experiments in Figure 1



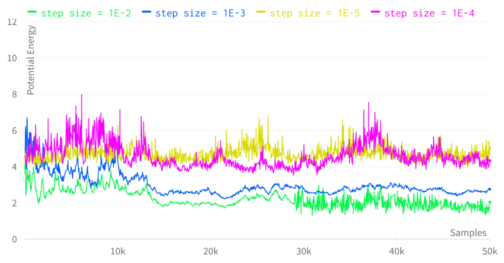
(a) 1 leapfrog per sample



(b) 2 leapfrogs per sample



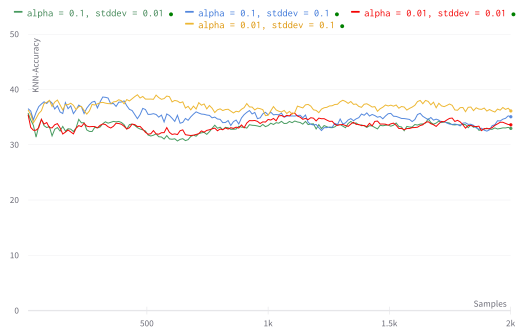
(c) 3 leapfrogs per sample



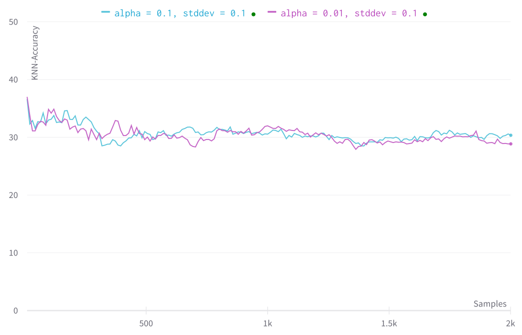
(d) 5 leapfrogs per sample

Figure 3: Potential Energy Loss measurements for each SGHMC parameter sample in each experiment of Figure 1: across 50 k sample proposals, using varying number of leapfrogs: 1 (a), 2 (b), 3 (c), 5 (d), with each subfigure depicting 4 plots for different step sizes.

A.4 SGHMC Experiments With Different Hyperparameters



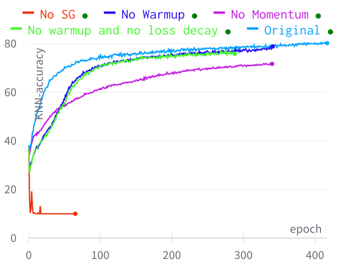
(a) $\epsilon = 0.001$



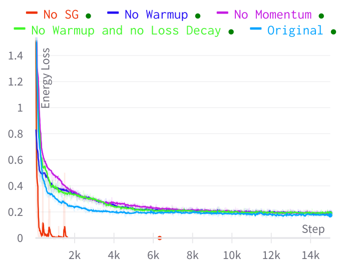
(b) $\epsilon = 0.01$

Figure 4: Additional SGHMC experiments varying the standard deviation of the velocities. In the main results shown previously in Figure 1, we fixed the standard deviation at 1.0. Here, we choose it from $\{0.1, 0.01\}$. Additionally, we try try different combinations of $\alpha = \boldsymbol{M}^{-1}\boldsymbol{C}\epsilon$, a hyperparameter modeling the relation of $\boldsymbol{M}, \boldsymbol{C}, \epsilon$. As mentioned in the main experiments, in Figure 4 we used $\alpha = \boldsymbol{M}^{-1}\boldsymbol{C}\epsilon = 0.01\boldsymbol{I}$. Here, we choose α from $\{0.1, 0.01\}$. We fix leapfrog steps to 3 for all experiments. a) uses learning rate $\epsilon = 0.001$ while b) uses $\epsilon = 0.01$. Results displayed for 2000 SGHMC samples. No evidence of collapse was found for any experiment, indicating that varying the strength of the velocities deviation isn't critical for the conclusions obtained in the paper.

A.5 Energy Loss on Original SimSiam Training



(a) k-NN accuracy per epoch



(b) Loss measurements

Figure 5: Training metrics for SimSiam adapted to our Energy loss, with variations in optimization hyperparameters, including SG removal.

In Figure 5 we depict the metrics for 400 epochs of SimSiam's training on CIFAR-10, with our proposed CNN architecture and modified energy loss given by $U(\mathbf{W}) = -\log(-\mathcal{L}_{SimSiamAbs})$. The results are for 5 optimization scenarios: 1) original training setting containing optimization tricks described in Chen and He [2020] (stop-gradient, 10 epochs warmup, cosine decaying loss after warmup, 0.9 momentum SGD); 2) without warmup epochs; 3) without momentum; 4) without warmup epochs and cosine decay; 5) without stop-gradient. Figure 5 a) depicts k-NN-accuracy for first 400 epochs and Figure 5 b) depicts the loss measurement over the first 14000 gradient updates. The reason we show all this scenarios is to understand the role of each optimization component in the performance (not just SG), in order to form a better comparisons with our SGHMC method, that lacks all these optimization particularities. We conclude that the energy loss doesn't affect the original SimSiam results, so our observed experiments with SGHMC aren't caused by this modification in the loss when computing the gradients. Besides, the particular optimization hyperparameters, especially momentum, of SimSiam play a important role in aiding SG to perform better in k-NN accuracy. So, the absence of these hyperparameters in SGHMC could potentially be hindering SGHMC from exploring regions of higher k-NN-accuracy in our experiments described in Figure 1.