A PROBABILISTIC APPROACH TO SELF-SUPERVISED LEARNING USING CYCLICAL STOCHASTIC GRADIENT MCMC

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we aim to enhance self-supervised learning by leveraging Bayesian techniques to capture the full posterior distribution over representations instead of relying on maximum a posteriori (MAP) estimates. Our primary objective is to demonstrate how a rich posterior distribution can improve performance, calibration, and robustness in downstream tasks. We introduce a practical Bayesian self-supervised learning method using Cyclical Stochastic Gradient Hamiltonian Monte Carlo (cSGHMC). By placing a prior over the parameters of the self-supervised model and employing cSGHMC, we approximate the high-dimensional, multimodal posterior distribution over the embeddings. This exploration of the posterior distribution yields interpretable and diverse representations. By marginalizing over these representations in downstream tasks, we gain significant improvements in predictive performance, calibration and out-of-distribution detection. We validate our method across various datasets, demonstrating the practical benefits of capturing the full posterior in Bayesian self-supervised learning.

1 INTRODUCTION

Self-supervised learning is a learning strategy where the data themselves provide the labels (Jing & Tian, 2020). The aim of self-supervised learning is to learn useful representations of the input data without relying on human annotations (Zbontar et al., 2021). Since they do not rely on annotated data, they have been used as an essential step in many areas such as natural language processing, computer vision and biomedicine (Jospin et al., 2022), where the data annotation is time-consuming and expensive. Despite the notable advancements made in recent years, self-supervised models are often trained using stochastic optimization methods which estimate the *distribution* over parameters as a *point mass*, ignoring the inherent uncertainty present in the parameter space. Remarkably, if the regularizer imposed on the model parameters is viewed as the log of a prior on the distribution of the parameters, optimizing the cost function may be viewed as a *maximum a-posteriori* (MAP) estimate of model parameters (Li et al., 2016b). Bayesian methods provide principled alternatives that model the whole posterior over the parameters and effectively account for the inherent uncertainty in the parameter space (Zhang et al., 2020). While the benefits of Bayesian methods and modeling uncertainty have been extensively explored in supervised learning (Li et al., 2016a; Maddox et al., 2019; Wilson & Izmailov, 2020), their potential advantages in self-supervised learning remain largely unexplored.

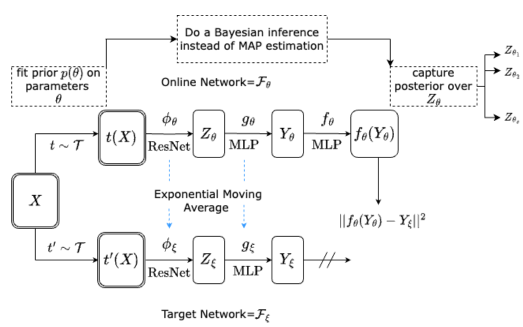


Figure 1: Illustration of our probabilistic self-supervised learning approach. We fit a prior over the parameters of the online network $\mathcal{F}_\theta$, perform Bayesian optimization instead of MAP estimation, capture the posterior over embeddings, and marginalize over embeddings in downstream tasks.

Indeed the posterior distribution over the parameters of a self-supervised learning model may be multimodal and thus insufficiently represented by a single point estimate. Each mode in the posterior can provide a meaningful different representation of data. By exploring the posterior distribution over the parameters instead of relying on point mass, our aim is to enhance performance and generalizability in downstream tasks. Additionally, it enables the estimation of uncertainties associated with predictions in downstream task, which holds significant value in numerous critical decision-making systems. In this paper, we address these challenges by introducing a novel approach that explores the posterior distribution over representations, offering a more robust framework for self-supervised learning. Our contributions are as follows:

- We propose a novel Bayesian formulation for self-supervised learning that surpasses the limitations of MAP estimation by approximating the full posterior distribution over representations.
- Our probabilistic approach uses cSGHMC, a family of Markov Chain Monte Carlo (MCMC) methods, to effectively capture multimodality in the posterior, enabling exploration of a diverse representation space and avoiding convergence to narrow, indistinguishable samples.
- We provide a rigorous empirical analysis to validate the effectiveness of our sampling-based approach. Our results demonstrate the potential of Bayesian learning, improving predictive performance, generalizability, and calibration in various downstream tasks. Specifically, we demonstrate the advantage of our method over deterministic approaches in tasks such as semi-supervised learning, transfer learning, and out-of-distribution detection.

2 Related Works

This work closely aligns with two lines of research: Bayesian inference and self-supervised learning.

Bayesian Inference Bayesian Deep Learning, emerging from Bayesian Neural Networks (Denker & LeCun, 1990; Neal, 1996), offers an alternative to point estimation by capturing model uncertainty. Sampling the posterior distribution presents challenges, leading to approximation methods. MCMC algorithms are popular for accurate posterior sampling, while variational inference (VI) learns an approximate posterior. Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC) methods (Welling & Teh, 2011; Chen et al., 2014; Ma et al., 2015)) combine MCMC with mini-batching, enabling scalable inference. Cyclical Stochastic Gradient MCMC (cSG-MCMC) (Zhang et al., 2020) specifically addresses the exploration of highly multimodal parameter spaces within realistic computational budgets.

Self Supervised Learning Self-supervised learning is key to extracting representations from vast unlabeled data, enhancing downstream task performance (Von Kügelgen et al., 2021). Among the promising approaches in self-supervised learning, contrastive methods (Chen et al., 2020) stand out. They learn representations by maximizing the similarity between embeddings of distorted images (Zbontar et al., 2021). A challenge in similarity learning is feature collapse, where features converge to a single point. Techniques like negative sampling in SimCLR (Chen et al., 2020) and stop gradients in BYOL (Grill et al., 2020) help prevent this collapse.

Pre-Trained Models as Bayesian Priors Previous works have modeled pretrained representations as Bayesian priors optimal for downstream tasks. Gao et al. (2022) use reference priors to compute uninformative priors via mutual information. Shwartz-Ziv et al. (2022) propose a variational approach to construct an informative prior. Our approach differs by sampling the full posterior rather than relying on a variational approximation, which risks overlay representation. We also employ a simple representation for the posterior, yielding effective results across tasks while improving uncertainty estimation.

3 Problem Statement

Given a dataset $\mathcal{D}$, a self-supervised learning model $\mathcal{F}_\theta$ parameterized by θ, aims to produce a representation Z_θ by solving a predefined proxy task. In this paper, we wish to learn a distribution over the embeddings Z_θ by placing a prior over the parameters θ and adopting Bayesian learning instead of relying on MAP estimation. Our method is illustrated in Fig. 1. To learn the representations, we use BYOL, a state-of-the-art contrastive learning method that eliminates the need for negative samples and demonstrates robustness to changes in batch size and data augmentations. However, this choice does not limit our probabilistic approach, which can extend seamlessly to other contrastive learning methods. In order to capture the distribution over the embeddings, we utilize cSGHMC. In the following sections, we first provide a description of the self-supervised learning model employed for representation learning. Then, we describe cSGHMC and demonstrate how it enables obtaining a distribution over the embeddings.

3.1 Self Supervised Learning

Contrastive learning aims to learn representations by contrasting two augmented views of an image. BYOL achieves this by minimizing a contrastive loss between an online network $\mathcal{F}_\theta$ (parameterized by θ) and target network $\mathcal{F}_\xi$ (parameterized by ξ). The online network consists of three components, an encoder $\phi(.)$ (e.g., Resnet-18), a projection head $g(.)$ (e.g., a Multi-Layer Perceptron (MLP)) and a prediction head $f(.)$ (e.g., an MLP). The target network is similar but lacks the prediction head. These two networks interact and learn from each other. The online network is trained to predict the representation of the target network, which is extracted from the same image under a different augmented view. The target network is updated using a slow-moving average of the online network, acting as a regularization mechanism and eliminating the need for negative samples, thereby preventing collapsed representations (Von Kügelgen et al., 2021). Moreover, this enhances BYOL's robustness to image augmentations and batch size changes compared to methods like SimCLR (Grill et al., 2020).

Formally, for a given mini-batch $X = \{x_i\}_{i=1}^N$ sampled from a dataset $\mathcal{D}$, BYOL produces two distorted views, $t(X)$ and $t'(X)$, via a distribution of data augmentations $\mathcal{T}$. These two batches of distorted views are then fed into the online network and the target network, respectively, resulting in batches of embeddings, Z_θ and Z_ξ. The embeddings are subsequently transformed into Y_θ and Y_ξ using the projection heads g_θ and g_ξ. The online network then outputs a prediction $f_\theta(Y_\theta)$ of Y_ξ employing the prediction head f_θ. Finally, the mean squared error between the normalized predictions $\overline{f_\theta}(Y_\theta)$ and target projections $\overline{Y_\xi}$ is defined as: $\mathcal{L}_{\theta,\xi} = \|\overline{f_\theta}(Y_\theta) - \overline{Y_\xi}\|^2$. $\tilde{\mathcal{L}}_{\theta,\xi}$ is computed by separately feeding $t'(X)$ to the online network $\mathcal{F}_\theta$ and $t(X)$ to the target network $\mathcal{F}_\xi$. It is worth noting that $\mathcal{L}_{\theta,\xi}$ and $\tilde{\mathcal{L}}_{\theta,\xi}$ are the same, the only distinction lies in the views fed to the target and online networks, which are swapped. Then, at each training step, a stochastic optimization step is performed to minimize $\mathcal{L}^{\text{BYOL}}_{\theta,\xi} = \mathcal{L}_{\theta,\xi} + \tilde{\mathcal{L}}_{\theta,\xi}$, where the gradient is taken only with respect to θ and *not* ξ[1]. So, during training *only* the parameters θ of the online network $\mathcal{F}_\theta$ are updated as follows:

$$\theta \leftarrow \text{optimizer}(\theta, \nabla_\theta \mathcal{L}^{\text{BYOL}}_{\theta,\xi}). \tag{1}$$

The weights ξ are an exponential moving average of the online network's parameters θ with a target decay rate $\tau \in [0, 1]$: $\xi \leftarrow \tau\xi + (1 - \tau)\theta$. At the end of training, the projection head $g_\theta(.)$ and the prediction head $f_\theta(.)$ are dropped and the encoder $\phi_\theta(.)$ is used for the downstream task.

3.2 Posterior Sampling using cSGHMC

In the Bayesian paradigm, for a given dataset $\mathcal{D} = \{x_i\}_{i=1}^n$ and a θ-parameterized model, the *posterior distribution* over θ is computed using Bayes' rule as: $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$, where $p(\theta)$ is a *prior* assigned to the parameters θ and $p(\mathcal{D}|\theta)$ is the likelihood.

In MAP optimization, the prior has the role of a regularizer and the likelihood has the role of a cost function. An optimizer is optimized to find the MAP solution which is amenable to the parameter update:

$$\Delta\theta = -\frac{\ell}{2}\left(\frac{n}{N}\sum_{i=1}^{N} \nabla_\theta \log p(x_i|\theta) + \nabla_\theta \log p(\theta)\right), \tag{2}$$

for a given randomly sampled mini-batch $X = \{x_i\}_{i=1}^N \subset \mathcal{D}$ and learning rate ℓ.

[1]It was depicted by stop-gradient in Fig.1

In contrast to MAP optimization, in the Bayesian paradigm the model explores the distribution over the model parameters. Welling & Teh (2011) showed that this distribution can be approximated using Stochastic Gradient Langevin Dynamics (SGLD) by injecting Gaussian noise to the parameter updates of SGD so that they do not collapse to just the MAP solution. This leads to the following parameter update:

$$\Delta\theta = -\frac{\ell}{2}\left(\frac{n}{N}\sum_{i=1}^{N}\nabla_\theta \log p(x_i|\theta) + \nabla_\theta \log p(\theta)\right) + \sqrt{\ell}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{3}$$

Note that when $\mathcal{D}$ is too large, it is too expensive to evaluate the log posterior $U(\theta) := \log p(\mathcal{D}|\theta) + \log p(\theta)$, for all the data points at each iteration. Hence, SG-MCMC methods use a mini-batch gradient to approximate $\nabla_\theta U(\theta)$ with an unbiased estimate $\nabla_\theta U(\theta) \approx n\nabla_\theta \tilde{U}(\theta)$, where $\nabla_\theta \tilde{U}(\theta) := \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log p(x_i|\theta) + \frac{1}{n}\nabla_\theta \log p(\theta)$. In particular, the log prior *scales* with the *dataset size* at each iteration. SGHMC (Chen et al., 2014) is an improved counterpart of SGLD which introduces a momentum variable m. The posterior sampling is done using the following update rule:

$$\begin{aligned} m &= \beta m - \frac{\ell}{2} n\nabla_\theta \tilde{U}(\theta) + \sqrt{(1-\beta)\ell}\epsilon; \quad \epsilon \sim \mathcal{N}(0, I) \\ \theta &= \theta + m, \end{aligned} \tag{4}$$

where β is the momentum term. The convergence to the true posterior is ensured by equation 3 and equation 4, given that learning rate ℓ follows the Robbins-Monro conditions and decays towards zero (Welling & Teh, 2011). Recently, cSG-MCMC was proposed, which adopts a cyclical learning rate schedule defined by cycles of iterations with a high-to-low learning rate and ensures the effective capture of *multimodal* posterior distribution. The method consists of two stages: The exploration stage, where a large learning rate at the beginning of each cycle encourages the sampler to take large steps, enabling it to escape local modes via stochastic gradients, and the sampling stage, where a small learning rate at the end of each cycle allows the sampler to explore individual local modes. In this paper we apply cSGHMC to take samples from the posterior distribution.

4 Posterior over Representations

To infer a posterior over the embeddings, we place a *prior* $p(\theta)$ over the parameters θ of the online network $\mathcal{F}_\theta$ as depicted in Fig. 1. By placing a distribution over θ, we induce a distribution over an infinite space of online networks $\mathcal{F}_\theta$. This, in turn, results in a distribution over embeddings Z_θ. Sampling from this distribution corresponds to sampling from the following conditional posterior:

$$p(\theta|X) \propto p(X|\theta)p(\theta), \tag{5}$$

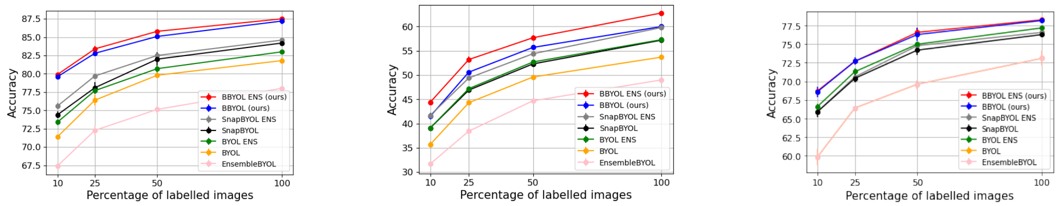
where X is a mini-batch. Equation 5 can be interpreted intuitively as follows: We sample weights θ from the prior $p(\theta)$. By conditioning on this sample of weights, we construct a specific online network $\mathcal{F}_\theta$. This network is then utilized to generate an embedding Z_θ for the mini-batch X. In the following, we describe how we use cSGHMC to sample from the posterior $p(\theta|X)$ over the representations.

Prior and Likelihood. To perform Bayesian approximation over embeddings, we place an isotropic Gaussian prior $p(\theta) = \mathcal{N}(0, I)$ on the parameters θ of the online network $\mathcal{F}_\theta$, which is implemented through weight decay. Then for a mini-batch $X = \{x_i\}_{i=1}^N$ we compute the mini-batch average gradient of $\tilde{U}(\theta)$, expressed as $\nabla_\theta \tilde{U}(\theta) := \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log p(x_i|\theta) + \frac{1}{n}\nabla_\theta \log p(\theta)$, where the likelihood $p(x_i|\theta)$ is the loss function similar to that used in BYOL, defined as:

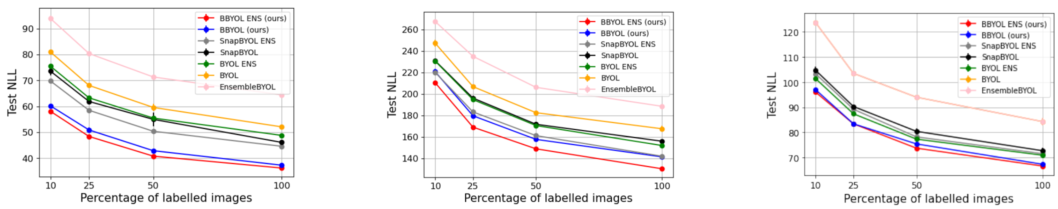
$$l_i(\theta) = \left\|\overline{f_\theta}(y_{i1}) - \overline{y}'_{i1}\right\|^2 + \left\|\overline{f_\theta}(y_{i2}) - \overline{y}'_{i2}\right\|^2, \tag{6}$$

with y_{ik}, y'_{ik} representing the online and target projections for i-th input sample x_i, respectively. Specifically, we define: $y_{i1} = g_\theta(\phi_\theta(t(x_i)))$, $y_{i2} = g_\theta(\phi_\theta(t'(x_i)))$, $y'_{i1} = g_\xi(\phi_\xi(t'(x_i)))$, $y'_{i2} = g_\xi(\phi_\xi(t(x_i)))$ where $t, t' \sim \mathcal{T}$. Then, we compute the following regularized loss function over a mini-batch X:

$$\mathcal{J}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l_i(\theta) + \frac{1}{2n}\|\theta\|^2, \tag{7}$$



(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet-10



(d) CIFAR-10 (e) CIFAR-100 (f) ImageNet-10

Figure 2: Performance comparison. First row indicates improvement in accuracy. The second row indicates improvement in calibration. Bayesian approaches outperform all other baselines in terms of both accuracy and calibration.

and update the parameters θ of the online network according to the rule outlined in equation 4. A precise description of our proposed method for sampling from the posterior distribution over embeddings Z_θ is outlined in Algorithm 1 in Appendix A. The algorithm generates samples from the posterior over the parameters θ of the online network $\mathcal{F}_\theta$. This yields a distribution over embeddings Z_θ, as we compute the gradients of the loss with respect to the sampled parameters θ.

Contrastive Learning and Cross Entropy Zimmermann et al. (2021) analyzed the link between contrastive learning and identifiability, showing that contrastive learning inverts the data-generating process. They demonstrated that the InfoNCE loss family corresponds to the cross-entropy between the ground-truth and inferred latent distributions. This suggests that the BYOL loss in equation 6 can also be interpreted as a cross-entropy loss, providing a valid likelihood. By incorporating a prior over the online network and using cSGHMC with the update rule from equation 4, we can efficiently sample from the posterior distribution over embeddings. To ensure valid samples, we only draw posterior samples after the model stabilizes, minimizing parameter updates. At this stage, the online network's parameters remain stationary, ensuring the samples accurately reflect the converged posterior.

Practical Considerations Wenzel et al. (2020) showed that tempering the posterior: $p(\theta|\mathcal{D}) \propto \exp(-U(\theta)/T)$, where $T < 1$ is the temperature, improves performance for Bayesian inference. In our work, we also adopt a cold posterior approach, selecting T via tuning on validation set (see Appendix C.1 for details). Following Zhang et al. (2020), we carefully chose the epoch at which Gaussian noise is injected to update parameters (referred to as *epoch-noise* in Algorithm 1), such that it maintains a balance between the exploration and sampling stages. Additionally, we address computational constraints, including sampling cost, efficiency, and memory overhead, discussed in Appendix B.

Our proposed probabilistic approach extends the principles of MAP optimization into a Bayesian framework, offering the added benefit of uncertainty estimation. In fact, by performing MAP optimization using SGD instead of posterior sampling, one approximates the entire *posterior* distribution over θ with a single *point estimate*, thus disregarding the richness of the full posterior.

Marginalizing over Representations: After completing the pre-training phase, we proceed to marginalize the posterior distribution over θ for downstream tasks. To compute the predictive distribution for a new instance x we use a model average over all collected samples with respect to the posterior over θ: $p(y|x,\mathcal{D}) = \int p(y|x,\theta)p(\theta|\mathcal{D})d\theta$. Solving this integral is intractable. Instead, we approximate it using Monte Carlo approximation, given by: $p(y|x,\mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y|x,\theta_s)$, where θ_s, $s = 1, \ldots, S$, is sampled from $p(\theta|\mathcal{D})$. We observe that this model average significantly enhances performance, calibration, and out-of-distribution detection in downstream tasks. In addition, by obtaining samples from the posterior, the uncertainty for a new instance x in a downstream task can be computed. In a multi-class classification setting with C classes, this is given by: $\mathcal{H}(y|x, \mathcal{D}) = -\sum_{c \in C} p(y = c|x, \mathcal{D}) \log p(y = c|x, \mathcal{D})$.

5 Experiments

In this section, we present the experimental results, evaluating the performance and efficiency of the proposed method across several tasks, including semi-supervised learning and out-of-distribution detection. We implemented our code in PyTorch (Paszke et al., 2017), and the code is available at: `https://github.com/Mjavan/PSelf-Supervised`.

5.1 Experimental Setup

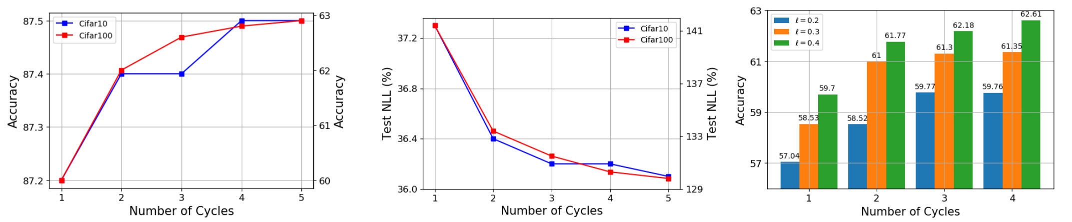
Datasets For pre-training phase, we pre-train all models on two image datasets STL-10 (Coates et al., 2011), using its 100,000 unlabeled samples, and Tiny-ImageNet (Le & Yang, 2015), using its 100,000-sample training set. For downstream tasks, we conduct our experiments on four image classification datasets: CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), STL-10 and ImageNet-10 (Chang et al., 2017). Pre-trained models are fine-tuned on different subsets of the training set from these datasets and evaluated on the test set. Only, for ImageNet-10, we use the validation set for evaluation due to the absence of ground-truth labels in the test set.

Implementation Details We adopt ResNet-18 (He et al., 2016) as an encoder for the self-supervised learning model. Following the original setting of BYOL, we use 2-layer MLPs as the projection and prediction heads. We apply the standard ResNet without modification on the input images of original sizes for all datasets which produces a feature vector of size 512 for each sample. We refer this feature vector as *representation* or *embedding*. We use the same set of data augmentations as described in Grill et al. (2020) on both datasets for pre-training, consisting of random cropping, resizing with a random horizontal flip, followed by a color distortion and a grayscale conversion.

Evaluation Metrics Two widely-used metrics including Accuracy (ACC), and Negative Log Likelihood (NLL) are used to evaluate our method. Higher value of ACC indicates better performance of the model and lower value of NLL indicates better calibration. NLL is a proper scoring rule and a popular metric for evaluating predictive uncertainty (Lakshminarayanan et al., 2017).

Baselines In order to demonstrate the effectiveness of our proposed probabilistic approach, referred to as BBYOL, we conduct a comparative analysis with several methods including: (i) BYOL: MAP estimation trained with SGD; (ii) BYOL ENS: stochastic optimization ensemble method trained with SGD, where network parameters are collected every 200 epochs; (iii) SnapBYOL: MAP estimation trained with SGD and cyclical stepsize schedule; (iv) SnapBYOL ENS: a stochastic optimization ensemble method with a cyclical stepsize schedule, where network parameters are collected at the end of ecah cycle and (v) EnsembleBYOL: an ensemble of BYOL trained with SGD from scratch for different random initialization. In the methods mentioned above, when we use only the last embedding in a downstream task we refer to the model as BYOL, SnapBYOL and BBYOL. Instead, when we perform marginalization over embeddings, we adopt BYOL ENS, SnapBYOL ENS and BBYOL ENS. EnsembleBYOL also signifies marginalizing over embeddings.

All models are trained from scratch for 1000 epochs. In BBYOL and SnapBYOL, we collect 1 sample at the end of each cycle for the last 4 cycles resulting in a total of 4 samples. In BYOL, we take 4 samples on last 200 epochs, maintaining a regular interval of 50 epochs between each sample. To ensure consistency in the training budget across all methods, we trained EnsembleBYOL by employing the SGD optimizer with a fixed learning rate for 250 epochs, using four different random seeds. Other training and baseline hyperparameters are provided in Appendix C.1. The experiments are carried out on Nvidia A40 48 GB and it takes about 21 gpu-hours on STL-10, and 24 gpu-hours on Tiny-ImageNet. We repeat experiments for 3 random seeds and report average NLL and ACC over 3 runs with the standard error from the mean predictor.



(a) Testing Accuracy (b) Testing NLL (c) Restart Learning Rate

Figure 3: Left (a-b): The effect of ensemble size on the performance of CIFAR-10 and CIFAR-100 in BBYOL ENS as a function of the number of cycles. Adding more embeddings improves accuracy and calibration. Right (c): The effect of restart learning rate on CIFAR-100. Larger learning rate improves accuracy.

5.2 Image Classification

5.2.1 Semi-Supervised Learning

In this section we present the evaluation results of proposed method on a semi-supervised image classification task. In this task, the quality of learned representations is assessed by fine-tuning a pre-trained model on subsets of original training datasets with labels. We evaluate over a variety of downstream training set sizes and analyze the obtained gains in performance and calibration. We follow the semi-supervised protocol in Grill et al. (2020) and provide a detailed description of hyperparameters in Appendix C.2.

In Fig. 2, we compare the methods outlined above across various dataset sizes in terms of accuracy and calibration. We observe the followings: (i) BBYOL consistently outperforms BYOL, SnapBYOL and Ensemble BYOL by a large margin in both metrics across all datasets. (ii) Marginalizing over representations in BBYOL ENS improves performance and calibration compared to BBYOL. Marginalizing is more effective when the downstream task is more difficult for example in CIFAR-100. (iii) Marginalizing over representations in BYOL ENS and SnapBYOL ENS also improves performance. It is due to the nature of contrastive loss which induces diversity in the parameter space. Whenever the loss is not too high, marginalizing over these representations contributes to enhanced performance. However, even with this improvement, BBYOL ENS still achieves sizable gains in both performance and calibration over the baselines.

Among above observations, Point (i) is particularly interesting, even if we do not want to use model averaging over representations due to a higher test-time cost, the last representation in BBYOL trained using a Bayesian approach has significant better performance in accuracy and calibration compared to a MAP estimation. In Appendix D.1, we provide additional evaluations with models pre-trained on Tiny-ImageNet. To further assess the efficacy of our proposed probabilistic approach, we also conduct experiments using SimCLR (Chen et al., 2020), with results presented in Appendix D.3. Consistent with our previous findings, the proposed probabilistic self-supervised methods outperform their deterministic counterparts across all metrics.

Ensemble Size In some applications, it may be beneficial to vary the size of the ensemble dynamically at test time depending on available resources. Fig. 3 (a-b) displays the performance of BBYOL ENS on CIFAR-10 and CIFAR-100 as the effective ensemble size, is varied. Although ensembling more models generally leads to better performance, in most cases we observe substantial improvements in accuracy and drops in NLL when the second and third models are introduced to the ensemble. This suggests that only a small number of embeddings are necessary to yield further performance gains in Bayesian model averaging. This implies that Bayesian marginalization provides particularly compelling results if one is willing to expend a little additional computation.

Restart Learning Rate We then investigate the impact of restart learning rate at the beginning of each cycle in Fig. 3 (c). The results confirm that ensembles with higher restart learning rates exhibit superior performance, likely attributed to the substantial perturbation introduced between cycles, thus enhancing representation diversity.

5.3 Out-of-Distribution Detection

To further investigate the efficacy of the proposed probabilistic approach compared to MAP estimation, we explore the out-of-distribution (OOD) detection task (Zhang et al., 2020). This task involves evaluating a model trained on known data with unseen data, where we expect a better model to exhibit low probability and maximum entropy, resulting in the mode of the predictive entropy histogram being focused at higher values. We consider two datasets CIFAR-10 and SVHN (Netzer et al., 2011) as OOD datasets. A pre-trained model on STL-10, is fine-tuned on CIFAR-100 and evaluated on SVHN and CIFAR-10. Fig. 4 presents the histogram of the predictive entropy for SVHN. The histogram for CIFAR-10 had the same distribution, so we just included SVHN. Additionally, we assess the quality of the predictive uncertainty using two quantitative metrics, NLL and the area under the receiver operating characteristic curve (AUROC) (Deng et al., 2009), a higher value of AUROC indicates a better detector. We see that the uncertainty estimates from BBYOL and BBYOL ENS are better than the other methods, as the mode of histogram focuses at higher values. BYOL ENS and SnapBYOL ENS also improve uncertainty estimate on unseen data compared to BYOL and SnapBYOL, respectively, but they still exhibit lower entropy than BBYOL ENS. Moreover, the predictive uncertainty improves on unseen data, as the ensemble size increases reaching to the highest value in BBYOL ENS (12), where we take 12 samples from last 4 cycles (3 samples per cycle). It indicates that the embeddings generated by sampling from the posterior in BBYOL suggest diverse modes, offering varied characterizations of the training data. When assessing unseen data, each mode yields distinct predictions, resulting in maximum disagreement and increased entropy. The quantitative results for NLL and AUROC are summarized in Table 1.

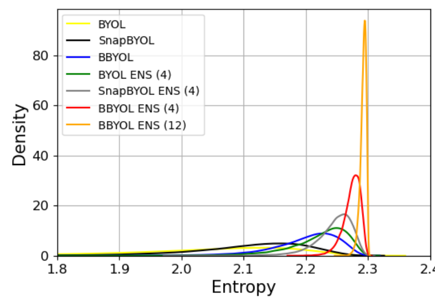


Figure 4: Histogram of the predictive entropy on test examples from unknown data (SVHN), as we vary the ensemble size.

Table 1: OOD detection. BBYOL ENS outperforms baselines across ensemble sizes (in parentheses). Our results are underlined; with the best in bold. Standard errors for NLL (rounded to two decimals) were zero

OOD	Method	NLL ↓	AUROC (%) ↑	OOD	Method	NLL ↓	AUROC (%) ↑
	BYOL	2.61	84.1 ± 1.7		BYOL	2.62	84.3 ± 1.4
SVHN	SnapBYOL	2.51	91.6 ± 0.7	CIFAR-10	SnapBYOL	2.52	90.8 ± 0.9
	BBYOL	2.40	95.3 ± 0.5		BBYOL	2.41	94.8 ± 0.6
	BYOL ENS (4)	2.38	93.6 ± 0.4		BYOL ENS (4)	2.38	93.6 ± 0.3
	SnapBYOL ENS (4)	2.35	96.8 ± 0.1		SnapBYOL ENS (4)	2.35	96.5 ± 0.0
	BBYOL ENS (4)	2.33	98.4 ± 0.0		BBYOL ENS (4)	2.33	98.2 ± 0.0
	BBYOL ENS (12)	**2.31**	**99.0**		BBYOL ENS (12)	**2.31**	**99.0**

6 Conclusion

In this paper, we introduce a Bayesian approach to representation learning that challenges the limitations of traditional MAP-based solutions. Rather than relying on point estimates, we explore the full posterior distribution over representations using a powerful SG-MCMC method tailored to capture multimodal structure. This probabilistic perspective enables richer and more diverse representations. Our extensive experiments reveal that sampling from the posterior leads to significant improvements in accuracy, calibration, and uncertainty estimation across various downstream tasks. By embracing the complexity of the posterior, our method offers deeper insights into data and enhances model robustness and reliability in real-world scenarios. While Bayesian marginalization introduces negligible computational overhead during inference, it still provides a clear performance boost.

7 Acknowledgments

We gratefully acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 459422098.

References

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pp. 5879–5887, 2017.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pp. 1683–1691. PMLR, 2014.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 215–223. PMLR, 2011.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

John S. Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, pp. 853–859, 1990.

Yansong Gao, Rahul Ramesh, and Pratik Chaudhari. Deep reference priors: What is the best way to pretrain a model? In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 7036–7051. PMLR, 2022.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. In *IEEE transactions on pattern analysis and machine intelligence*, volume 43, pp. 4037–4058, 2020.

Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. In *IEEE Computational Intelligence Magazine*, volume 17, pp. 29–48, 2022.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 30, pp. 6405–6416, 2017.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. In *CS 231N*, volume 7, pp. 3, 2015.

Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016a.

Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, and Lawrence Carin. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5666–5675, 2016b.

Yi-An Ma, Tianqi Chen, and Emily B. Fox. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, volume 28, pp. 2917–2925, 2015.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in neural information processing systems*, volume 32, 2019.

Radford M. Neal. Bayesian learning for neural networks. Lecture Notes in Statistics. Springer, New York, NY, 1996.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

Ravid Shwartz-Ziv, Micah Goldblum, Hossein Souri, Sanyam Kapoor, Chen Zhu, Yann LeCun, and Andrew G Wilson. Pre-train your loss: Easy bayesian transfer learning with informative priors. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27706–27715, 2022.

Julius Von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. In *Advances in neural information processing systems*, volume 34, pp. 16451–16467, 2021.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML, pp. 681–688, 2011.

Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub undefinedwiatkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*, ICML, 2020.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS, 2020.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. In *International Conference on Learning Representations*, 2020.

Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In *International Conference on Machine Learning*, 2021.

A Psedocodes

Algorithm 1 Probabilistic Self-Supervised Learning

Input: ℓ_0 initial learning rate; $\beta \in [0, 1)$ momentum term; $p(\theta) = \mathcal{N}(0, I)$ prior over θ; $\tau \in [0, 1]$ target decay rate; $T \geq 0$ temperature; K number of iterations in one cycle
Output: sequence $\theta_1, \theta_2, ...$

for $k = 1, 2, \ldots$ **do**
  $X_k = \{x_i\}_{i=1}^N$ // sample a batch of N images
  for $x_i \in X_k$ **do**
    $t \sim \mathcal{T}, t' \sim \mathcal{T}$ // sample image augmentations
    $y_{i1} = g_\theta(\phi_\theta(t(x_i)))$ and $y_{i2} = g_\theta(\phi_\theta(t'(x_i)))$
    $y'_{i1} = g_\xi(\phi_\xi(t'(x_i)))$ and $y'_{i2} = g_\xi(\phi_\xi(t(x_i)))$
    $l_i(\theta) = \|\overline{f_\theta}(y_{i1}) - \overline{y}'_{i1}\|^2 + \|\overline{f_\theta}(y_{i2}) - \overline{y}'_{i2}\|^2$
  end for
  $\ell_k \leftarrow C(k)\ell_0$ // update learning rate using cyclic modulation
  if *epoch-noise* **then**
    $\epsilon_k \sim \mathcal{N}(0, I)$ // sample noise
    $m_k \leftarrow \beta m_{k-1} - \frac{\ell_k}{2} \frac{n}{N} \sum_{i=1}^N \nabla_\theta \log l_i(\theta) - \frac{\ell_k}{2} \nabla_\theta \log p(\theta) + \sqrt{T(1-\beta)\ell_k} \epsilon_k$
  else
    $m_k \leftarrow \beta m_{k-1} - \frac{\ell_k}{2} \frac{n}{N} \sum_{i=1}^N \nabla_\theta \log l_i(\theta) - \frac{\ell_k}{2} \nabla_\theta \log p(\theta)$
  end if
  $\theta_k \leftarrow \theta_{k-1} + m_k$ // update θ_k using Equation equation 4
  $\xi_k \leftarrow \tau \xi_{k-1} + (1 - \tau)\theta_k$
  if $k \mod K = 0$ **then**
    yield θ_k // sample θ_k at the end of cycle
  end if
end for

B Computational Considerations

In the pre-training phase, both BBYOL and BBYOL ENS have the same computation time as BYOL trained with SGD, since samples are drawn during the training of a single network. In terms of memory overhead, BBYOL introduces only a minimal additional requirement compared to traditional MAP estimation. As only one sample is taken per cycle, the memory overhead remains negligible. We store only the posterior sample at the end of each cycle, which requires insignificant extra memory. Hence, Bayesian optimization does not incur any significant computational cost during pre-training, apart from this small and manageable memory overhead.

The primary computational overhead occurs during the downstream phase, where both fine-tuning and prediction costs scale linearly with the number of samples drawn from the posterior. For example, if 4 samples are drawn, Bayesian inference incurs approximately 4 times the computational cost of the traditional MAP method. However, our experiments indicate that even a modest sample size (3-4) significantly boosts performance and generalization across all metrics compared to the baselines, making this approach computationally efficient. Notably, a single posterior sample (without ensembling) in BBYOL consistently outperforms traditional MAP methods at the same computational cost. Moreover, BBYOL ENS significantly outperforms deep ensembles as well as BYOL ENS and SnapBYOL ENS, which have the same train and test-time costs.

C Implementation Details

C.1 Pre-Training

BBYOL We follow the steps outlined in Algorithm 1 for training BBYOL. We use cSGHMC (Zhang et al., 2020) with cyclic learning rate schedule of length 50 epochs. In accordance with Zhang et al. (2020), we adopt normal prior $\mathcal{N}(0, I)$ for the parameters of the online network. As stated in the main paper, scaling the prior with the dataset size is necessary for training cSGHMC, given that we evaluate it on a minibatch of data. For injecting Gaussian noise, we swapped over epochs $\{35, 40, 45\}$ and select epoch 40. For the initial learning rate we swapped over $\{0.1, 0.2, 0.3\}$

and set the initial learning rate to 0.2. The batch size is set to 256 and we use momentum term of 0.9. As described in (Zhang et al., 2020), tempering helps improve performance for Bayesian inference with neural networks. Therefore, we swapped the values $\{0.1, 0.01\}$ and set the temperature to 0.1.

BYOL For training BYOL we use SGD optimizer with a fixed learning rate schedule and momentum term 0.9. For the initial learning rate we swapped over $\{0.0003, 0.003\}$ and set the learning rate to 0.003. We found that using weight decay destroys representations so no weight decay is used. Other parameters are as the same as BBYOL.

SnapBYOL To train SanpBYOL, we employ the SGD optimizer with a cyclic learning rate schedule of length 50 epochs. For the initial learning rate, for STL-10 we swapped over $\{0.1, 0.09, 0.07, 0.03, 0.02, 0.01, 0.009\}$ and set the initial learning rate to 0.01. For Tiny-ImageNet we swapped over $\{0.03, 0.02, 0.01, 0.009, 0.008, 0.007, 0.006\}$ and set the initial learning rate to 0.006. It is worth noting that for learning rates higher than this, the model does not converge when applied to Tiny-ImageNet. Similar to BYOL, we did not employ weight decay, and all other parameters remain consistent with those of BYOL.

EnsembleBYOL To train EnsembleBYOL, we utilized the same parameters as BYOL while incorporating a normal prior of $\mathcal{N}(0, I)$ on the parameters of the online network. We trained the model for four different random seeds, each for 250 epochs.

C.2 Fine-Tunning

For fine-tuning, we follow the protocol of Grill et al. (2020). To begin, we initialize the network with the parameters of the pre-trained representation and then fine-tune it using a subset of the original labeled datasets. We do not use any data augmentation during fine-tuning. We utilize cross-entropy as the loss function and employ SGD with Nesterov momentum as the optimizer. We set the batch size to 80 and the momentum to 0.9. We experiment with different combinations of learning rates including $\{2e-5, 1e-5, 1e-4, 2e-4, 3e-4, 4e-4, 5e-4\}$, weight decay options of $\{0, 5e-4\}$ and the number of epochs set to $\{50, 60\}$. We select the hyperparameters that give us the best performance on our local validation set and report performance on test set. Table 2 describes parameters for each dataset.[2]

D Additional Experiments

D.1 Experiments with Pre-Trained Models on Tiny-ImageNet

In this section we provide results obtained from pre-training on Tiny-ImageNet and fine-tuned on CIFAR-10 and CIFAR-100. We pre-train all models with parameters described in Appendix C and take 4 samples at last 200 epochs for BYOL, BBYOL and SnapBYOL. For EnsembleBYOL, we marginalise over representations obtained from four pre-trained models, each trained with different random seeds. The results in semi-supervised image classification on CIFAR-10 and CIFAR-100 are illustrated in Fig. 5. Consistent with our results from pre-training on STL-10, BBYOL and BBYOL ENS outperform their MAP estimation counterparts in both metrics and various dataset sizes.

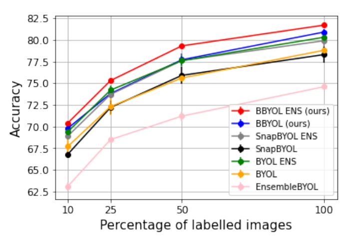
D.2 Experiments using ImageNet Pre-Trained Encoder

To disentangle the impact of parameters from the enhancements achieved by our proposed probabilistic method compared to the deterministic baselines, we perform an experiment in which we initialize the backbone parameters (ResNet-18), using pre-trained weights from ImageNet. In this experiment, we employ an ImageNet-trained ResNet-18 feature extractor as an encoder and train all methods described in Appendix C.1 for 200 epochs on STL-10, with an initial learning rate set to 0.1. As described in Appendix C.1, we use cyclic learning rate schedule of length 50 epochs for BBYOL and SnapBYOL and take one sample at the end of each cycle, yielding 4 embeddings in total. In BYOL, we take 4 samples with regular interval of 50 epochs between each sample. For BBYOL

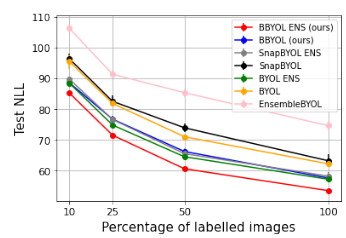
[2] We employed the same parameters for fine-tuning on ImageNet-10 using a pre-trained model on STL-10, which were originally used for fine-tuning of a model pre-trained on Tiny-ImageNet.

Table 2: Parameters used during the fine-tuning phase for each dataset

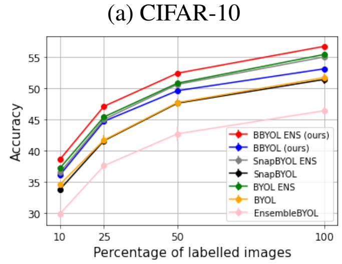
Pre-training	Fine-tuning	Split (%)	Learning rate	Weight decay
STL-10	CIFAR-10	100	$2e-4$	0
		50	$5e-4$	0
		25	$5e-4$	0
		10	$5e-4$	0
	CIFAR-100	100	$5e-4$	0
		50	$3e-4$	0
		25	$4e-4$	0
		10	$5e-4$	0
	STL-10	100	$5e-4$	0
		50	$5e-4$	0
		25	$5e-4$	0
		10	$5e-4$	0
Tiny-ImageNet	CIFAR-10	100	$2e-4$	0
		50	$5e-4$	0
		25	$4e-4$	0
		10	$4e-4$	0
	CIFAR-100	100	$5e-4$	0
		50	$5e-4$	0
		25	$5e-4$	0
		10	$5e-4$	0
	ImageNet-10	100	$2e-4$	0
		50	$5e-4$	0
		25	$3e-4$	0
		10	$5e-4$	0



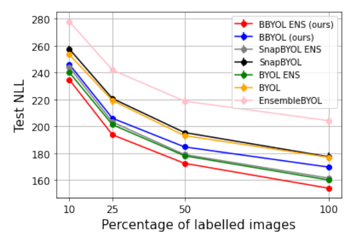
(a) CIFAR-10



(b) CIFAR-10



(c) CIFAR-100



(d) CIFAR-100

Figure 5: Performance comparison. Bayesian approach outperforms MAP estimation in terms of both accuracy and calibration.

we employ $\mathcal{N}(0, I)$ as a prior and introduce Gaussian noise starting from epoch 40. The rest of parameters remain consistent as described in Appendix C.1. In the downstream phase, we fine-tune on the training set of each classification task, following the protocol outlined in Appendix C.2 and using the parameters from Table 2. Table 3 indicates the accuracy and NLL on the test sets of each dataset[3]. Consistent with our previous results, BBYOL and BBYOL ENS outperform all baselines in both metrics.

Table 3: Downstream evaluation results on different datasets by fine-tuning on each task: bold indicates the best result. BBYOL and BBYOL ENS lead to better classification accuracy as well as better predictive uncertainty as evidenced by lower NLL.

	Method / Data	CIFAR10	CIFAR100	ImageNet10	STL10
ACC↑	BYOL	93.3	73.6	90.2	93.0
	SnapBYOL	93.8	74.4	91.9	93.6
	BBYOL	93.6	76.8	94.9	93.6
	BYOL ENS	94.1	77.0	91.6	93.5
	SnapBYOL ENS	94.2	77.5	92.6	93.9
	BBYOL ENS	**94.3**	**79.5**	**95.9**	**94.4**
	Method / Data	CIFAR10	CIFAR100	ImageNet10	STL10
NLL↓	BYOL	0.19	0.89	0.32	0.21
	SnapBYOL	0.18	0.87	0.25	0.19
	BBYOL	0.18	0.77	0.14	0.19
	BYOL ENS	0.16	0.76	0.26	0.19
	SnapBYOL ENS	**0.16**	0.75	0.22	0.18
	BBYOL ENS	**0.16**	**0.67**	**0.11**	**0.17**

D.3 Experiments using SimCLR

We also assess the efficacy of our proposed probabilistic approach with the SimCLR method (Chen et al., 2020). In this experiment, we employ an ImageNet-trained ResNet-18 feature extractor as the backbone. We train methods described in Appendix C.1 for 200 epochs on STL-10, using the NT-Xent loss (Chen et al., 2020) with a temperature of 0.1 and an initial learning rate set to 0.1. We adhere to the settings outlined in Appendix C.1 for each method, employing 4 embeddings with a 50-epoch interval between each sample for marginalization. Table 4 displays the results, with BSimCLR representing our proposed Bayesian method, while SimCLR and SnapSimCLR denote the deterministic methods trained using fixed and cyclic learning rate schedules, respectively. We observe that the results obtained by BSimCLR and BSimCLR ENS outperform all baselines with a significant gain in both metrics.

[3]Only for ImageNet-10, we employ the Validation set for evaluation due to the absence of ground-truth labels in the Test set.

Table 4: Downstream evaluation results on different datasets by fine-tuning on each task using representations from SimCLR: bold indicates the best result. BSimCLR and BSimCLR ENS lead to better accuracy and better predictive uncertainty.

	Method / Data	CIFAR10	CIFAR100	ImageNet10	STL10
ACC↑	SimCLR	88.9	67.5	84.8	86.7
	SnapSimCLR	89.6	69.4	85.7	86.7
	BSimCLR	<u>93.7</u>	<u>75.5</u>	<u>93.5</u>	<u>92.8</u>
	SimCLR ENS	89.3	69.1	85.5	86.5
	SnapSimCLR ENS	89.9	70.8	86.3	86.3
	BSimCLR ENS	**94.0**	**78.4**	**94.3**	**93.3**

	Method / Data	CIFAR10	CIFAR100	ImageNet10	STL10
NLL↓	SimCLR	0.31	1.09	0.50	0.39
	SnapSimCLR	0.28	1.04	0.46	0.38
	BSimCLR	<u>0.19</u>	<u>0.81</u>	<u>0.20</u>	<u>0.21</u>
	SimCLR ENS	0.29	1.03	0.47	0.39
	SnapSimCLR ENS	0.27	0.98	0.44	0.38
	BSimCLR ENS	**0.17**	**0.71**	**0.17**	**0.20**