Function Space Bayesian Pseudocoreset for Bayesian Neural Networks

Balhae Kim [1] **Hyungi Lee** [1] **Juho Lee** [1]

Abstract

A Bayesian pseudocoreset is a compact synthetic dataset summarizing essential information of a large-scale dataset and thus can be used as a proxy dataset for scalable Bayesian inference. Typically, a Bayesian pseudocoreset is constructed by minimizing a divergence measure between the posterior conditioning on the pseudocoreset and the posterior conditioning on the full dataset. However, evaluating the divergence can be challenging, particularly for the models like deep neural networks having high-dimensional parameters. In this paper, we propose a novel Bayesian pseudocoreset construction method that operates on a function space. Unlike previous methods, which construct and match the coreset and full data posteriors in the space of model parameters (weights), our method constructs variational approximations to the coreset posterior on a function space and matches it to the full data posterior in the function space. By working directly on the function space, our method could bypass several challenges that may arise when working on a weight space, including limited scalability and multi-modality issue.

1. Introduction

Deep learning has achieved remarkable success in various domains, but accurately estimating and characterizing the uncertainties associated with its predictions remain challenging. Bayesian approaches offer a principled framework to address this issue by capturing uncertainty through probability distributions. However, applying Bayesian methods to large-scale deep learning problems poses significant computational difficulties due to the high dimensionality of the models and the massive amounts of data involved (Chen et al., 2014; Baker et al., 2017; Quiroz et al., 2018).

[1]Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. Correspondence to: Balhae Kim <balhaekim@kaist.ac.kr>, Juho Lee <juholee@kaist.ac.kr>.

Accepted to ICML workshop on Structured Probabilistic Inference Generative Modeling. Copyright 2023 by the author(s).

To address this issue, a potential solution is to employ a Bayesian coreset (Huggins et al., 2016). A Bayesian coreset is a small subset of the original dataset where the posterior conditioning on it closely approximates the original posterior conditioning on the full dataset. Once the Bayesian coreset is trained, it can be utilized as a lightweight proxy dataset for subsequent Bayesian inference.

A Bayesian coreset is constructed by selecting a subset from a large dataset and learning the corresponding weights. However, recent research suggests that this approach may not be effective, especially in high-dimensional settings (Manousakas et al., 2020). Instead, an alternative method of synthesizing a coreset, wherein the coreset is learned as trainable parameters, has been found to significantly enhance the quality of the approximation. This synthesized coreset is referred to as a Bayesian pseudocoreset. The process of learning a Bayesian pseudocoreset involves minimizing a divergence measure between the posterior of the full dataset and the posterior of the pseudocoreset. However, it is generally challenging due to the intractability of constructing the posterior distributions, as well as computing the divergence between them, which necessitates approximation. Consequently, existing works on Bayesian pseudocoresets have primarily focused on small-scale problems. (Manousakas et al., 2020; Chen et al., 2022; Manousakas et al., 2022; Naik et al., 2023). Recently, Kim et al. (2022) introduced a scalable method for constructing Bayesian pseudocoresets, however, it still demands substantial computational resources for high-dimensional models like deep neural networks.

In this paper, we present a novel approach to enhance the scalability of Bayesian pseudocoreset construction, particularly for Bayesian neural networks (BNNs) with a large number of parameters. Our proposed method operates in *function space*. When working with BNNs, it is common to define a prior distribution on the weight space and infer the corresponding weight posterior distribution, which also applies to Bayesian pseudocoreset construction. However, previous studies (Sun et al., 2019; Rudner et al., 2022a) have highlighted the challenge of interpreting weights in high-dimensional neural networks, making it difficult to elicit meaningful prior distributions. Additionally, in high-dimensional networks, the loss surfaces often exhibit a complex multimodal structure, which means that proximity in the weight space does not necessarily imply proximity in the desired prediction variable (Pan et al., 2020a; Rudner et al., 2022b). This same argument can be applied to Bayesian pseudocoreset construction, as matching the full data and pseudocoreset posteriors in the weight space may not result in an optimal pseudocoreset in terms of representation power and computational scalability.

To be more specific, our method constructs a variational approximation to the pseudocoreset posteriors in function space by linearization and variational approximation to the true posterior. Then we learn Bayesian pseudocoreset by minimizing a divergence measure between the full data posterior and the pseudocoreset posterior in the function space. Compared to the previous weight space approaches, our method readily scales to the large models for which the weight space approaches were not able to compute. Moreover, it has another advantage that the posteriors learned from the Bayesian pseudocoreset in function space have better out-of-distribution (OOD) robustness, similar to the previous reports showing the benefit of function space approaches in OOD robustness (Rudner et al., 2022a).

In summary, this paper presents a novel approach to creating a scalable and effective Bayesian pseudocoreset using function space variational inference. The resulting Bayesian pseudocoreset is capable of being generated in high-dimensional image and deep neural network settings and has better uncertainty quantification abilities compared to weight space variational inference. We demonstrate the efficiency of the function space Bayesian pseudocoreset through the various experiments.

2. Background

2.1. Bayesian pseudocoresets

In this paper, we focus on probabilistic models for supervised learning problem. Let $\theta \in \Theta$ be a parameter, and let $p(y \,|\, x, \theta)$ be a probabilistic model indexed by the parameter θ. Given a set of observations $\mathbf{x} := (x_i)_{i=1}^n$ and the set of labels $\mathbf{y} := (y_i)_{i=1}^n$ with each $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, we are interested in updating our prior belief $\pi_0(\theta)$ about the parameter to the posterior,

$$\pi_{\mathbf{x}}(\theta) := \frac{\pi_0(\theta)}{Z(\mathbf{y} \,|\, \mathbf{x})} \prod_{i=1}^n p(y_i \,|\, x_i, \theta), \tag{1}$$

$$Z(\mathbf{y} \,|\, \mathbf{x}) := \int_\Theta \prod_{i=1}^n p(y_i \,|\, x_i, \theta) \pi_0(\mathrm{d}\theta). \tag{2}$$

However, when the size of the dataset n is large, the computation of the posterior distribution can be computationally expensive and infeasible. To overcome this issue, Bayesian pseudocoresets are constructed as a synthetic dataset $\mathbf{u} = (u_j)_{j=1}^m$ with $m \ll n$ with the set of labels $\tilde{\mathbf{y}} := (\tilde{y}_j)_{j=1}^m$ where the posterior conditioning on it approximates the original posterior $\pi_{\mathbf{x}}(\theta)$.

$$\pi_{\mathbf{u}}(\theta) = \frac{\pi_0(\theta)}{Z(\tilde{\mathbf{y}} \,|\, \mathbf{u})} \prod_{j=1}^m p(\tilde{y}_j \,|\, u_j, \theta), \tag{3}$$

$$Z(\tilde{\mathbf{y}} \,|\, \mathbf{u}) := \int_\Theta \prod_{j=1}^m p(\tilde{y}_j \,|\, u_j, \theta) \pi_0(\mathrm{d}\theta). \tag{4}$$

This approximation is made possible by solving an optimization problem that minimizes a divergence measure D between the two posterior distributions [1]

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \; D(\pi_{\mathbf{x}}, \pi_{\mathbf{u}}). \tag{5}$$

2.2. Bayesian pseudocoresets in weight-space

Kim et al. (2022) advocates using forward KL divergence as the divergence measure when constructing Bayesian pseudocoresets, with the aim of achieving a more even exploration of the posterior distribution of the full dataset when performing uncertainty quantification with the learned pseudocoreset. The derivative of the forward KL divergence with respect to the pseudocoreset $\mathbf{u}$ is computed as

$$\begin{aligned} \nabla_{\mathbf{u}} D_{\mathrm{KL}}[\pi_{\mathbf{x}} \| \pi_{\mathbf{u}}] = \mathbb{E}_{\pi_{\mathbf{u}}}\Big[\nabla_{\mathbf{u}} \sum_{j=1}^m \log p(\tilde{y}_j \,|\, u_j, \theta)\Big] \\ - \nabla_{\mathbf{u}} \mathbb{E}_{\pi_{\mathbf{x}}}\Big[\sum_{j=1}^m \log p(\tilde{y}_j | u_j, \theta))\Big] \end{aligned} \tag{6}$$

For the gradient, we need the expected gradients of the log posteriors that require sampling from the posteriors $\pi_{\mathbf{x}}$ and $\pi_{\mathbf{u}}$. Most of the probabilistic models do not admit simple closed-form expressions for these posteriors, and it is not easy to simulate those posteriors for high-dimensional models. To address this, Kim et al. (2022) proposes to use a Gaussian variational distributions $q_{\mathbf{u}}(\theta)$ and $q_{\mathbf{x}}(\theta)$ to approximate $\pi_{\mathbf{x}}$ and $\pi_{\mathbf{u}}$ whose the means are set to the parameters obtained from the SGD trajectories,

$$q_{\mathbf{u}}(\theta) = \mathcal{N}(\theta; \mu_{\mathbf{u}}, \Sigma_{\mathbf{u}}), \quad q_{\mathbf{x}}(\theta) = \mathcal{N}(\theta; \mu_{\mathbf{x}}, \Sigma_{\mathbf{x}}), \tag{7}$$

where $\mu_{\mathbf{u}}$ and $\mu_{\mathbf{x}}$ are the maximum a posteriori (MAP) solutions computed for the dataset $\mathbf{u}$ and $\mathbf{x}$, respectively. The gradient, with the stop gradient applied to $\mu_{\mathbf{u}}$, is approximated as,

$$\begin{aligned} \frac{\nabla_{\mathbf{u}}}{S} \sum_{s=1}^S \Big(\sum_{j=1}^m \log p\Big(\tilde{y}_j \,|\, u_j, \mathrm{sg}(\mu_{\mathbf{u}}) + \Sigma_{\mathbf{u}}^{1/2} \varepsilon_{\mathbf{u}}^{(s)}\Big) \\ - \sum_{j=1}^m \log p\Big(\tilde{y}_j \,|\, u_j, \mu_{\mathbf{x}} + \Sigma_{\mathbf{x}}^{1/2} \varepsilon_{\mathbf{x}}^{(s)}\Big)\Big). \end{aligned} \tag{8}$$

[1] In principle, we should learn the (pseudo)labels $\tilde{\mathbf{y}}$ as well, but for classification problem, we can simply fix it as a constant set containing equal proportion of all possible classes. We assume this throughout the paper.

Here, $\varepsilon_{\mathbf{u}}^{(s)}$ and $\varepsilon_{\mathbf{x}}^{(s)}$ are i.i.d. standard Gaussian noises and S is the number of Monte-Carlo samples.

3. Function space Bayesian pseudocoreset

3.1. Function space Bayesian neural networks

We follow the framework presented in Rudner et al. (2020; 2022a) to define a Function-space Bayesian Neural Network (FBNN). Let $\pi_0(\theta)$ be a prior distribution on the parameter and $g_\theta : \mathcal{X} \to \mathbb{R}^d$ be a neural network index by θ. Let $h : \Theta \to (\mathcal{X} \to \mathbb{R}^d)$ be a deterministic mapping from a parameter θ to a neural network g_θ. Then a function-space prior is simply defined as a pushforward $\nu_0(f) = h_*\pi_0(f) = \pi_0(h^{-1}(f))$. The corresponding posterior is also defined as a pushforward $\nu_{\mathbf{x}}(f) = h_*\pi_{\mathbf{x}}(f)$ and so is the pseudocoreset posterior $\nu_{\mathbf{u}}(f) = h_*\pi_{\mathbf{u}}(f)$.

3.2. Learning function space Bayesian pseudocoresets

Given the function space priors and posteriors, a Function space Bayesian PseudoCoreset (FBPC) is obtained by minimizing a divergence measure between the function space posteriors. We follow Kim et al. (2022) suggesting to use the forward KL divergence, so our goal is to solve

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \; D_{\mathrm{KL}}[\nu_{\mathbf{x}} \| \nu_{\mathbf{u}}]. \tag{9}$$

The following proposition provides an expression for the gradient to minimize the divergence, whose proof is given Appendix A.

Proposition 3.1. *The gradient of the forward KL divergence with respect to the coreset* $\mathbf{u}$ *is*

$$\begin{aligned}&\nabla_{\mathbf{u}} D_{\mathrm{KL}}[\nu_{\mathbf{x}} \| \nu_{\mathbf{u}}]\\ &= -\nabla_{\mathbf{u}} \mathbb{E}_{[\nu_{\mathbf{x}}]_{\mathbf{u}}}[\log p(\tilde{\mathbf{y}} \mid \mathbf{f}_{\mathbf{u}})] + \mathbb{E}_{[\nu_{\mathbf{u}}]_{\mathbf{u}}}[\nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \mid \mathbf{f}_{\mathbf{u}})],\end{aligned} \tag{10}$$

where $[\nu_{\mathbf{x}}]_{\mathbf{u}}$ *and* $[\nu_{\mathbf{u}}]_{\mathbf{u}}$ *are finite-dimensional distributions of the stochastic processes* $\nu_{\mathbf{x}}$ *and* $\nu_{\mathbf{u}}$*, respectively,* $\mathbf{f}_{\mathbf{u}} := (f(u_j))_{j=1}^m$*, and* $p(\tilde{\mathbf{y}} \mid \mathbf{f}_{\mathbf{u}}) = \prod_{j=1}^m p(\tilde{y}_j \mid f(u_j))$.

To evaluate the gradient Equation (10), we should identify the finite-dimensional functional posterior distributions $[\nu_{\mathbf{x}}]_{\mathbf{u}}$ and $[\nu_{\mathbf{u}}]_{\mathbf{u}}$. While this is generally intractable, as proposed in Rudner et al. (2020; 2022a), we can instead consider a linearized approximation of the neural network g_θ,

$$\tilde{g}_\theta(\cdot) = g_{\mu_{\mathbf{x}}}(\cdot) + \mathcal{J}_{\mu_{\mathbf{x}}}(\cdot)(\theta - \mu_{\mathbf{x}}), \tag{11}$$

where $\mu_{\mathbf{x}} = \mathbb{E}_{\pi_{\mathbf{x}}}[\theta]$ and $\mathcal{J}_{\mu_{\mathbf{x}}}(\cdot)$ is the Jacobian of g_θ evaluated at $\mu_{\mathbf{x}}$. Then we approximate the function space posterior $\nu_{\mathbf{x}}$ with $\tilde{\nu}_{\mathbf{x}} := \tilde{h}_*\pi_{\mathbf{x}}$ where $\tilde{h}(\theta) = \tilde{g}_\theta$, and as shown in Rudner et al. (2020; 2022a), the finite dimensional distribution $[\tilde{\nu}_{\mathbf{x}}]_{\mathbf{u}}$ is a multivariate Gaussian distribution,

$$[\tilde{\nu}_{\mathbf{x}}]_{\mathbf{u}}(\mathbf{f}_{\mathbf{u}}) = \mathcal{N}\Big(\mathbf{f}_{\mathbf{u}} \mid g_{\mu_{\mathbf{x}}}(\mathbf{u}), \mathcal{J}_{\mu_{\mathbf{x}}}(\mathbf{u})\Sigma_{\mathbf{x}}\mathcal{J}_{\mu_{\mathbf{x}}}(\mathbf{u})^\top\Big), \tag{12}$$

with $\Sigma_{\mathbf{x}} = \mathrm{Cov}_{\pi_{\mathbf{x}}}(\theta)$. Similarly, we obtain $[\tilde{\nu}_{\mathbf{u}}]_{\mathbf{u}}(\mathbf{f}_{\mathbf{u}})$. Using these linearized finite-dimensional distribution, we can approximate

$$\begin{aligned}&\nabla_{\mathbf{u}} D_{\mathrm{KL}}[\nu_{\mathbf{x}} \| \nu_{\mathbf{u}}]\\ &= -\nabla_{\mathbf{u}} \mathbb{E}_{[\tilde{\nu}_{\mathbf{x}}]_{\mathbf{u}}}[\log p(\tilde{\mathbf{y}} \mid \mathbf{f}_{\mathbf{u}})] + \mathbb{E}_{[\tilde{\nu}_{\mathbf{u}}]_{\mathbf{u}}}[\nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \mid \mathbf{f}_{\mathbf{u}})],\end{aligned} \tag{13}$$

3.3. Tractable approximation to the gradient

Even with the linearization, evaluating Equation (13) is still challenging because it requires obtaining $\mu_{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$ which are the statistics of the weight-space posterior $\pi_{\mathbf{x}}$. Rudner et al. (2022a) proposes to learn a variational approximation $q_{\mathbf{x}}(\theta)$ in the *weight-space*, and use the linearized pushforward of the variational distribution $\tilde{h}_* q_{\mathbf{x}}$ as a proxy to the function space posterior. Still, this approach requires computing the heavy Jacobian matrix $\mathcal{J}_{\mathbb{E}_{q_{\mathbf{x}}}[\theta]}(\mathbf{u})$, so may not be feasible for our scenario where we have to compute such variational approximations *at each* update of the pseudocoreset $\mathbf{u}$.

Instead, we choose to directly construct a variational approximations to the finite-dimensional distributions of the function space posteriors, that is,

$$\begin{aligned}&[\tilde{\nu}_{\mathbf{x}}]_{\mathbf{u}}(\mathbf{f}_{\mathbf{u}}) \approx q_{\mathbf{x}}(\mathbf{f}_{\mathbf{u}}) = \mathcal{N}(\mathbf{f}_{\mathbf{u}} \mid g_{\hat{\mu}_{\mathbf{x}}}(\mathbf{u}), \hat{\Psi}_{\mathbf{x}}),\\ &[\tilde{\nu}_{\mathbf{u}}]_{\mathbf{u}}(\mathbf{f}_{\mathbf{u}}) \approx q_{\mathbf{u}}(\mathbf{f}_{\mathbf{u}}) = \mathcal{N}(\mathbf{f}_{\mathbf{u}} \mid g_{\hat{\mu}_{\mathbf{u}}}(\mathbf{u}), \hat{\Psi}_{\mathbf{u}}),\end{aligned} \tag{14}$$

where $(\hat{\mu}_{\mathbf{x}}, \hat{\Psi}_{\mathbf{x}})$ and $(\hat{\mu}_{\mathbf{u}}, \hat{\Psi}_{\mathbf{u}})$ are variational parameters for the full data and coreset posteriors. We obtain $\hat{\mu}_{\mathbf{x}}$ and $\hat{\mu}_{\mathbf{u}}$ using MAP solutions for $\mathbf{x}$ and $\mathbf{u}$, respectively. For the covariance matricies $\hat{\Psi}_{\mathbf{x}}$ and $\hat{\Psi}_{\mathbf{u}}$, we set them as an empirical covariance matrices of the samples collected from the optimization trajectory. Please refer to Appendix B.2 for a detailed description of the training procedure. Using the obtained variational approximations, we can derive a Monte-Carlo estimator for Equation (13),

$$\begin{aligned}&\nabla_{\mathbf{u}} D_{\mathrm{KL}}[\nu_{\mathbf{x}} | \nu_{\mathbf{u}}]\\ &\quad\approx -\nabla_{\mathbf{u}} \mathbb{E}_{p(\varepsilon_{\mathbf{x}})}[\log p(\tilde{\mathbf{y}} \mid \hat{\mu}_{\mathbf{x}} + \hat{\Psi}_{\mathbf{x}}^{1/2}\varepsilon_{\mathbf{x}})]\\ &\qquad + \mathbb{E}_{p(\varepsilon_{\mathbf{u}})}\Big[\nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \mid \hat{\mu}_{\mathbf{u}} + \hat{\Psi}_{\mathbf{u}}^{1/2}\varepsilon_{\mathbf{u}})\Big]\\ &\quad\approx \frac{1}{S}\sum_{s=1}^S \Big(-\nabla_{\mathbf{u}} \log p\Big(\tilde{\mathbf{y}} \mid \hat{\mu}_{\mathbf{x}} + \hat{\Psi}_{\mathbf{x}}^{1/2}\varepsilon_{\mathbf{x}}^{(s)}\Big)\\ &\qquad + \nabla_{\mathbf{u}} \log p\Big(\tilde{\mathbf{y}} \mid \hat{\mu}_{\mathbf{u}} + \hat{\Psi}_{\mathbf{u}}^{1/2}\varepsilon_{\mathbf{u}}^{(s)}\Big)\Big),\end{aligned} \tag{15}$$

where $p(\varepsilon_{\mathbf{x}})$ and $p(\varepsilon_{\mathbf{u}})$ are standard Gaussians, $(\varepsilon_{\mathbf{x}}^{(s)})_{s=1}^S$ and $(\varepsilon_{\mathbf{u}}^{(s)})_{s=1}^S$ are i.i.d. samples from them.

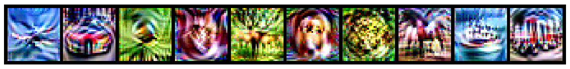


Figure 1. Example images of FBPC for CIFAR10.

Table 1. Averaged SGHMC performances of each Bayesian pseudocoreset on the CIFAR10 dataset.

ipc	SGHMC	Random	BPC-rKL	BPC-fKL	FBPC (Ours)
1	Acc ↑	16.30±0.74	20.44±1.06	34.50±1.62	**35.45**±0.31
	NLL ↓	4.66±0.03	4.51±0.10	3.86±0.13	**3.79**±0.04
10	Acc ↑	32.48±0.34	37.92±0.66	56.19±0.61	**62.33**±0.34
	NLL ↓	2.98±0.03	2.47±0.04	1.48±0.02	**1.31**±0.02
50	Acc ↑	49.68±0.46	51.86±0.38	64.74±0.32	**71.23**±0.17
	NLL ↓	2.06±0.02	1.95±0.02	1.26±0.01	**1.03**±0.05

Table 2. Averaged SGHMC performances of each Bayesian pseudocoreset on the CIFAR100 datasets.

	ipc	1	10	50
Random	Acc ↑	4.82±0.47	18.0±0.31	35.1±0.23
	NLL ↓	5.55±0.07	4.57±0.01	3.35±0.01
BPC-fKL	Acc ↑	14.7±0.16	28.1±0.60	37.1±0.33
	NLL ↓	4.17±0.05	3.53±0.05	3.28±0.24
FBPC (Ours)	Acc ↑	**21.0**±0.76	**39.7**±0.31	**44.47**±0.35
	NLL ↓	**3.76**±0.11	**2.67**±0.02	**2.63**±0.01

4. Experiments

4.1. Experimental Setup

In our study, we employed the CIFAR10, CIFAR100 and Tiny-ImageNet datasets to create Bayesian pseudocoresets of coreset size $m \in \{1, 10, 50\}$ images per class (ipc). These pseudocoresets were then evalutated by conducting the Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014) algorithm on those pseudocoresets. We measured the top-1 accuracy and negative log-likelihood of the SGHMC algorithm on the respective test datasets.

we employed three baseline methods to compare the effectiveness of function space Bayesian pseudocoresets. The first baseline is the random coresets, which involves selecting a random mini-batch of the coreset size. The others two baseline methods, BPC-rKL (Kim et al., 2022; Manousakas et al., 2020) and BPC-fKL (Kim et al., 2022), are Bayesian pseudocoresets on weight space. BPC-rKL and BPC-fKL employ reverse KL divergence and forward KL divergence as the divergence measures for their training, respectively.

4.2. Main Results

Table 1, Table 2 and Table 3 show the results of each baseline and our method for each dataset. For BPC-rKL and BPC-fKL, we used the official code from (Kim et al., 2022) for training the pseudocoresets, and only difference is that we used our own SGHMC hyperparameters during evaluation.

Table 3. Averaged SGHMC performances of each Bayesian pseudocoreset on the Tiny-ImageNet datasets.

	ipc	1	10	50
Random	Acc ↑	1.90±0.08	7.21±0.04	19.15±0.12
	NLL ↓	6.18±0.04	5.77±0.02	4.88±0.01
BPC-fKL	Acc ↑	3.98±0.13	11.4±0.45	-
	NLL ↓	5.63±0.03	5.08±0.05	-
FBPC (Ours)	Acc ↑	**10.14**±0.68	**19.42**±0.51	**26.43**±0.31
	NLL ↓	**4.69**±0.05	**4.14**±0.02	**4.30**±0.05

Table 4. The results for CIFAR10-C dataset.

	BPC-fKL	FBPC
Acc ↑	34.3	47.5
Degradation ↓	38.9	23.7

For detailed experiment setting, please refer the Appendix B. Moreover, it is worth noting that for the Tiny-ImageNet dataset, constructing BPC-fKL was not possible due to the memory constraint.

The results clearly demonstrate that our FBPC method outperforms all the baseline approaches, including random coresets, BPC-rKL and BPC-fKL, in terms of both accuracy and negative log-likelihood, especially on the large-scale datasets in Table 2 and Table 3.

Furthermore, the Bayesian pseudocoreset can be leveraged to enhance robustness against distributional shifts when combined with Bayesian model averaging. To assess the robustness of our function space Bayesian pseudocoresets on out-of-distribution inputs, we also conducted experiments using the CIFAR10-C dataset, which involves the insertion of image corruptions into the CIFAR10 images. we measure the top-1 accuracy and degradation scores, which indicate the extent to which accuracy is reduced compared to the in-distribution's test accuracy. In Table 4, we present the averaged results for 10 types of corruptions. For detailed results, please refer to the appendix. The result demonstrates that our FBPC consistently outperforms the weight space Bayesian pseudocoreset, BPC-fKL.

5. Conclusion

In this paper, we explored the function space Bayesian pseudocoreset. We constructed it by minimizing forward KL divergence between the function posteriors of pseudocoreset and the entire dataset. To optimize the divergence, we proposed a novel method to effectively approximate the function posteriors with an efficient training procedure. Finally, we empirically demonstrated the superiority of our function space Bayesian pseudocoresets compared to weight space Bayesian pseudocoresets, in terms of test performance and uncertainty quantification.

Acknowledgments

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)), Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00713).

References

Baker, J., Fearnhead, P., Fox, E. B., and Nemeth, C. Control variates for stochastic gradient mcmc, 2017.

Burt, D. R., Ober, S. W., Garriga-Alonso, A., and van der Wilk, M. Understanding variational inference in function-space, 2020.

Campbell, T. and Beronov, B. Sparse variational inference: Bayesian coresets from scratch. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019.

Campbell, T. and Broderick, T. Bayesian coreset construction via greedy iterative geodesic ascent. In *Proceedings of The 35th International Conference on Machine Learning (ICML 2018)*, 2018.

Campbell, T. and Broderick, T. Automated scalable bayesian inference via hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588, 2019.

Carvalho, E. D. C., Clark, R., Nicastro, A., and Kelly, P. H. J. Scalable uncertainty for computer vision with functional variational inference, 2020.

Cazenavette, G., Wang, T., Torralba, A., Efros, A. A., and Zhu, J.-Y. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Chen, N., Xu, Z., and Campbell, T. Bayesian inference via sparse hamiltonian flows. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.

Chen, T., Fox, E. B., and Guestrin, C. Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, 2014.

de G. Matthews, A. G., Hensman, J., Turner, R. E., and Ghahramani, Z. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. *Proceedings of Machine Learning Research*, 51: 231–239, 2016.

Huggins, J., Campbell, T., and Broderick, T. Coresets for bayesian logistic regression. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, 2016.

Kim, B., Choi, J., Lee, S., Lee, Y., Ha, J.-W., and Lee, J. On divergence measures for bayesian pseudocoresets. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.

Ma, C. and Hernández-Lobato, J. M. Functional variational inference based on stochastic process generators. In *Advances in Neural Information Processing Systems*, 2021.

Manousakas, D., Xu, Z., Mascolo, C., and Campbell, T. Bayesian pseudocoresets. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

Manousakas, D., Ritter, H., and Karaletsos, T. Black-box coreset variational inference, 2022.

Naik, C., Rousseau, J., and Campbell, T. Fast bayesian coresets via subsampling and quasi-newton refinement, 2023.

Nguyen, T., Chen, Z., and Lee, J. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations (ICLR)*, 2020.

Nguyen, T., Novak, R., Xiao, L., and Lee, J. Dataset distillation with infinitely wide convolutional networks. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, 2021.

Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R. E., and Khan, M. E. Continual deep learning by functional regularisation of memorable past. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020a.

Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R. E., and Khan, M. E. Continual deep learning by functional regularisation of memorable past. *Advances in neural information processing systems*, 2020b.

Quiroz, M., Kohn, R., Villani, M., and Tran, M.-N. Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*, 114(526):831–843, jul 2018. doi: 10.1080/01621459.2018.1448827.

Rudner, T. G., Chen, Z., Teh, Y. W., and Gal, Y. Tractable function-space variational inference in bayesian neural networks, 2022a.

Rudner, T. G. J., Chen, G., and Gal, Y. Rethinking function space variational inference in Bayesian neural networks. *3rd Symposium on Advances in Approximate Bayesian Inference*, 2020.

Rudner, T. G. J., Bickford Smith, F., Feng, Q., Teh, Y. W., and Gal, Y. Continual learning via sequential function-space variational inference. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 18871–18887, 2022b.

Rudner, T. G. J., Chen, Z., Teh, Y. W., and Gal, Y. Tractable function-space variational inference in bayesian neural networks. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022c.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Titsias, M. K., Schwarz, J., de G. Matthews, A. G., Pascanu, R., and Teh, Y. W. Functional regularisation for continual learning with gaussian processes, 2020.

Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Wang, Z., Ren, T., Zhu, J., and Zhang, B. Function space particle optimization for Bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Zhao, B. and Bilen, H. Dataset condensation with differentiable siamese augmentation. In *Proceedings of The 38th International Conference on Machine Learning (ICML 2021)*, 2021.

Zhao, B., Mopuri, K. R., and Bilen, H. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021.

Zhou, Y., Nezhadarya, E., and Ba, J. Dataset distillation using neural feature regression. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

A. Proofs

Proposition A.1. *The gradient of the forward KL divergence with respect to the coreset* $\mathbf{u}$ *is*

$$\begin{aligned}&\nabla_{\mathbf{u}} D_{\mathrm{KL}}[\nu_{\mathbf{x}} \| \nu_{\mathbf{u}}] \\ &= -\nabla_{\mathbf{u}} \mathbb{E}_{[\nu_{\mathbf{x}}]_{\mathbf{u}}}[\log p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}})] + \mathbb{E}_{[\nu_{\mathbf{u}}]_{\mathbf{u}}}[\nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}})],\end{aligned} \tag{10}$$

where $[\nu_{\mathbf{x}}]_{\mathbf{u}}$ *and* $[\nu_{\mathbf{u}}]_{\mathbf{u}}$ *are finite-dimensional distributions of the stochastic processes* $\nu_{\mathbf{x}}$ *and* $\nu_{\mathbf{u}}$*, respectively,* $\mathbf{f}_{\mathbf{u}} := (f(u_j))_{j=1}^m$*, and* $p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}}) = \prod_{j=1}^m p(\tilde{y}_j \,|\, f(u_j))$*.*

Proof. We follow the arguments in de G. Matthews et al. (2016); Rudner et al. (2020). The forward KL divergence is defined as,

$$D_{\mathrm{KL}}[\nu_{\mathbf{x}} \| \nu_{\mathbf{u}}] = \int \log \frac{\mathrm{d}\nu_{\mathbf{x}}}{\mathrm{d}\nu_{\mathbf{u}}}(f) \mathrm{d}\nu_{\mathbf{x}}(f). \tag{16}$$

By the chain rule for the Radon-Nikodym derivative, we have

$$D_{\mathrm{KL}}[\nu_{\mathbf{x}} \| \nu_{\mathbf{u}}] = \int \log \frac{\mathrm{d}\nu_{\mathbf{x}}}{\mathrm{d}\nu_0}(f) \mathrm{d}\nu_{\mathbf{x}}(f) - \int \log \frac{\mathrm{d}\nu_{\mathbf{u}}}{\mathrm{d}\nu_0}(f) \mathrm{d}\nu_{\mathbf{x}}(f). \tag{17}$$

The first term does not depend on $\mathbf{u}$, so we investigate the second term. By the measure theoretic Bayes' rule,

$$\frac{\mathrm{d}\nu_{\mathbf{u}}}{\mathrm{d}\nu_0}(f) = \frac{p(\tilde{\mathbf{y}} \,|\, f, \mathbf{u})}{p(\tilde{\mathbf{y}} \,|\, \mathbf{u})}, \tag{18}$$

where $p(\tilde{\mathbf{y}} \,|\, f, \mathbf{u}) := \prod_{j=1}^m p(\tilde{y}_j \,|\, u_j, f)$ and,

$$p(\tilde{\mathbf{y}}|\mathbf{u}) = \int p(\tilde{\mathbf{y}}|f, \mathbf{u}) \mathrm{d}\nu_0(f). \tag{19}$$

Now let $\rho_A : (\mathcal{X} \to \mathbb{R}^d) \to (A \to \mathbb{R}^d)$ be a projection function that takes a function f and returns its restriction on a set $A \subseteq \mathcal{X}$. Assuming that the likelihood depends only on the finite index set $\mathbf{u}$, we can write

$$\frac{\mathrm{d}\nu_{\mathbf{u}}}{\mathrm{d}\nu_0}(f) = \frac{\mathrm{d}[\nu_{\mathbf{u}}]_{\mathbf{u}}}{\mathrm{d}[\nu_0]_{\mathbf{u}}}(\rho_{\mathbf{u}}(f)) = \frac{p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}})}{p(\tilde{\mathbf{y}} \,|\, \mathbf{u})}, \tag{20}$$

where $[\cdot]_{\mathbf{u}}$ denotes the finite-dimensional distribution of stochastic process evalauted at $\mathbf{u}$ and $\rho_{\mathbf{u}}(f) := \mathbf{f}_{\mathbf{u}} := (f(u_j))_{j=1}^m$ are corresponding function values at $\mathbf{u}$. Putting this back into the above equation,

$$\begin{aligned}\int \log \frac{\mathrm{d}\nu_{\mathbf{u}}}{\mathrm{d}\nu_0}(f) \mathrm{d}\nu_{\mathbf{x}}(f) &= \int \log \frac{\mathrm{d}[\nu_{\mathbf{u}}]_{\mathbf{u}}}{\mathrm{d}[\nu_0]_{\mathbf{u}}}(\mathbf{f}_{\mathbf{u}}) \mathrm{d}[\nu_{\mathbf{x}}]_{\mathbf{u}}(\mathbf{f}_{\mathbf{u}}) \\ &= \int \log \frac{p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}})}{p(\tilde{\mathbf{y}} \,|\, \mathbf{u})} \mathrm{d}[\nu_{\mathbf{x}}]_{\mathbf{u}}(\mathbf{f}_{\mathbf{u}}) \\ &= \mathbb{E}_{[\nu_{\mathbf{x}}]_{\mathbf{u}}}[\log p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}})] - \log p(\tilde{\mathbf{y}} \,|\, \mathbf{u}).\end{aligned} \tag{21}$$

Now taking the gradient w.r.t. $\mathbf{u}$, we get

$$\nabla_{\mathbf{u}} D_{\mathrm{KL}}[\nu_{\mathbf{x}} \| \nu_{\mathbf{u}}] = -\nabla_{\mathbf{u}} \mathbb{E}_{[\nu_{\mathbf{x}}]_{\mathbf{u}}}[\log p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}})] + \nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \,|\, \mathbf{u}). \tag{22}$$

Note also that

$$\begin{aligned}\nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \,|\, \mathbf{u}) &= \nabla_{\mathbf{u}} \log \int p(\tilde{\mathbf{y}} \,|\, f, \mathbf{u}) \mathrm{d}\nu_0(f) \\ &= \int \frac{\nabla_{\mathbf{u}} p(\tilde{\mathbf{y}} \,|\, f, \mathbf{u})}{p(\tilde{\mathbf{y}} \,|\, \mathbf{u})} \mathrm{d}\nu_0(f) \\ &= \int \nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \,|\, f, \mathbf{u}) \frac{p(\tilde{\mathbf{y}} \,|\, f, \mathbf{u})}{p(\tilde{\mathbf{y}} \,|\, \mathbf{u})} \mathrm{d}\nu_0(f) \\ &= \int \nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \,|\, f, \mathbf{u}) \frac{\mathrm{d}\nu_{\mathbf{u}}}{\mathrm{d}\nu_0}(f) \mathrm{d}\nu_0(f) \\ &= \int \nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \,|\, f, \mathbf{u}) \mathrm{d}\nu_{\mathbf{u}}(f) \\ &= \int \nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}}) \mathrm{d}[\nu_{\mathbf{u}}]_{\mathbf{u}}(f) = \mathbb{E}_{[\nu_{\mathbf{u}}]_{\mathbf{u}}}[\nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \,|\, \mathbf{f}_{\mathbf{u}})].\end{aligned} \tag{23}$$

As a result, we conclude that

$$\nabla_{\mathbf{u}} D_{\mathrm{KL}}[\nu_{\mathbf{x}} \| \nu_{\mathbf{u}}] = -\nabla_{\mathbf{u}} \mathbb{E}_{[\nu_{\mathbf{x}}]_{\mathbf{u}}}[\log p(\tilde{\mathbf{y}} \mid \mathbf{f}_{\mathbf{u}})] + \mathbb{E}_{[\nu_{\mathbf{u}}]_{\mathbf{u}}}[\nabla_{\mathbf{u}} \log p(\tilde{\mathbf{y}} \mid \mathbf{f}_{\mathbf{u}})]. \tag{24}$$

□

B. Experimental Details

B.1. Expert trajectories

Approximating the full data and coreset posteriors with variational distributions requires the MAP solutions $\mu_{\mathbf{u}}$ and $\mu_{\mathbf{x}}$ as consequences of running optimization algorithms untill convergence. While this may be feasible for small datasets, for large-scale setting of our interest, obtaining $\mu_{\mathbf{u}}$ and $\mu_{\mathbf{x}}$ from scratch at each iteration for updating $\mathbf{u}$ can be time-consuming. To alleviate this, in the dataset distillation literature, Cazenavette et al. (2022) proposed to use the *expert trajectories*, the set of pretrained optimization trajectories constructed in advance to the coreset learning. Kim et al. (2022) brought this idea to Bayesian pseudocoresets, where a pool of pretrained trajectories are assume to be given before pseudocoreset learning. At each step of pseudocoreset update, a checkpoint θ_0 from an expert trajectory is randomly drawn from the pool, and then $\mu_{\mathbf{u}}$ and $\mu_{\mathbf{x}}$ are quickly constructed by taking few optimization steps from θ_0.

For expert trajectory, we trained the network with the entire dataset and saved their snapshot parameters at every epoch, following the setup described in (Cazenavette et al., 2022). For training, we used an SGD optimizer with a learning rate of 0.01. We saved 100 training trajectories, with each trajectory consisting of 50 epochs.

B.2. Detailed training procedure

Inspired by Kim et al. (2022), we construct the variational parameters in Equation (14) using expert trajectories. Unlike (Kim et al., 2022), we simply let the MAP solution computed for $\mathbf{x}$, $\theta_{\mathbf{x}}$, by sampling a checkpoint from the later part of the expert trajectories, and obtain the MAP solution of $\mathbf{u}$, $\theta_{\mathbf{u}}$, by directly optimizing an initial random parameter. Then we obtain $\hat{\mu}_{\mathbf{x}}$ and $\hat{\mu}_{\mathbf{u}}$ using $\mathbf{u}$. For the covariance matricies $\hat{\Psi}_{\mathbf{x}}$ and $\hat{\Psi}_{\mathbf{u}}$, while Kim et al. (2022) proposed to use spherical Gaussian noises, we instead set them as an empirical covariance matrices of the samples collected from the optimization trajectory. Specifically, we take additional K steps from each MAP solution to compute the empirical covariance.

$$\theta_{\mathbf{x}}^{(0)} = \theta_{\mathbf{x}}, \quad \theta_{\mathbf{x}}^{(t)} = \mathtt{opt}(\theta_{\mathbf{x}}^{(t-1)}, (\mathbf{x}, \mathbf{y})), \quad \hat{\mu}_{\mathbf{x}} = g_{\mathtt{sg}(\theta_{\mathbf{x}}^{(0)})}(\mathbf{u}), \tag{25}$$

$$\hat{\Psi}_{\mathbf{x}} := \mathtt{sg}\left(\mathrm{diag}\left(\frac{1}{K}\sum_{k=1}^{K} g^2_{\theta_{\mathbf{x}}^{(k)}}(\mathbf{u}) - \left(\frac{1}{K}\sum_{k=1}^{K} g_{\theta_{\mathbf{x}}^{(k)}}(\mathbf{u})\right)^2\right)\right), \tag{26}$$

where $\mathtt{opt}(\theta, \mathbf{x})$ is a step of SGD optimization applied to θ with data $\mathbf{x}$ and the squares in the $\mathrm{diag}(\cdot)$ are applied in element-wise manner. Note also that we are applying the stop-gradient operations for to block the gradient flow that might lead to complication in the backpropagation procedure. The variational parameters $(\hat{\mu}_{\mathbf{u}}, \hat{\Psi}_{\mathbf{u}})$ are constructed in a similar fashion, but using the psedocoreset $(\mathbf{u}, \tilde{\mathbf{y}})$ instead of the original data $(\mathbf{x}, \mathbf{y})$. The overview of proposed method is provided in Figure 2.

B.3. Evaluation

To implement the SGHMC algorithm, as discussed in (Chen et al., 2014) and following the recommendations of (Chen et al., 2014), we employed the SGD with momentum along with an auxiliary noise term.

$$\begin{cases} \Delta\theta = v \\ \Delta v = -\eta \nabla \tilde{U}(x) - \alpha v + \mathcal{N}(0, 2d). \end{cases} \tag{27}$$

we set $\eta = 0.03$, $\alpha = 0.1$, and $d = 0.01/m$, where m represents the coreset size. We perform 1000 epochs of SGHMC and collect samples every 100 epochs.

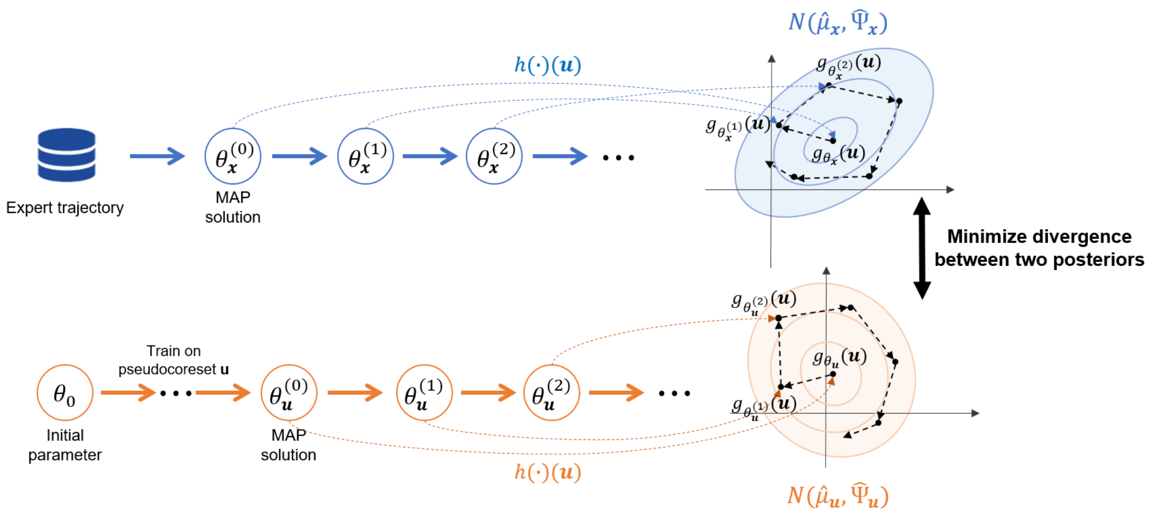


Figure 2. The conceptual overview of our proposed training procedure.

Algorithm 1 Multi-architecture Function space Bayesian Pseudocoreset

```
Input: Set of architectures $\mathcal{A}$, expert trajectories $\{\mathcal{E}^{(a)} : a \in \mathcal{A}\}$, prior distributions of parameters $\{\pi_0^{(a)} : a \in \mathcal{A}\}$, an optimizer opt.
Initialize $\mathbf{u}$ with random minibatch of coreset size $m$.
for $i = 1, \dots, N$ do
  Initialize the gradient of pseudocoreset: $\mathbf{g} \leftarrow 0$.
  for $a \in \mathcal{A}$ do
    Sample the MAP solution computed for $\mathbf{x}$: $\theta_{\mathbf{x}} \in \mathcal{E}^{(a)}$.
    Sample an initial random parameter: $\theta_0 \sim \pi_0^{(a)}(\theta)$.
    repeat
      $\theta_t \leftarrow \mathtt{opt}(\theta_{t-1}, (\mathbf{u}, \tilde{\mathbf{y}}))$
    until converges to obtain the MAP solution computed for $\mathbf{u}$: $\theta_{\mathbf{u}}$
    Obtain $\hat{\mu}_{\mathbf{x}}$, $\hat{\mu}_{\mathbf{u}}$, $\hat{\Psi}_{\mathbf{x}}$, $\hat{\Psi}_{\mathbf{u}}$ by Equation (25).
    Compute the pseudocoreset gradient $\mathbf{g}^{(a)}$ using Equation (15).
    $\mathbf{g} \leftarrow \mathbf{g} + \mathbf{g}^{(a)}$.
  end for
  Update the pseudocoreset $\mathbf{u}$ using the gradient $\mathbf{g}$.
end for
```

C. Additional Experiments

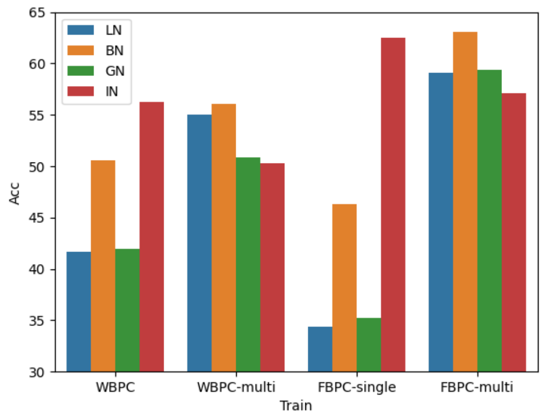
C.1. Multiple architectures FBPC training

Another benefit of function space matching is that it does not constrain the number of architectures of the neural networks to be matched, provided that their inherited function space posteriors are likely to be similar. Indeed, the function space is typically of much lower dimension compared to the weight space. This makes it more likely for function spaces to exhibit similar posterior shapes in the vicinity of the MAP solutions. This characteristic of function space encourages the exploration of function space pseudocoreset training in the context of multiple architectures. Because, the task of training a coreset that matches the highly complex weight space posterior across multiple architectures is indeed challenging, while the situation becomes relatively easier when dealing with architectures that exhibit similar function posteriors.

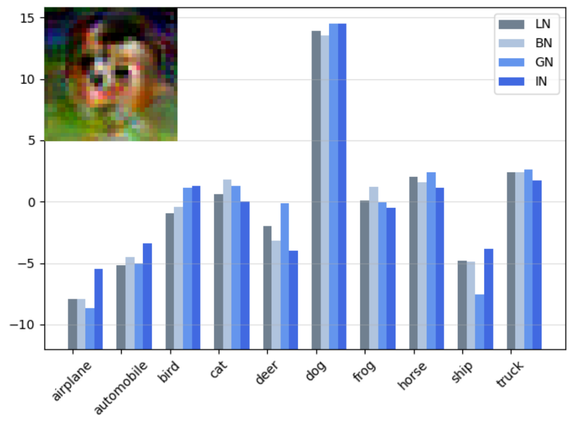
Therefore we propose a novel multi-architecture FBPC algorithm in Algorithm 1. The training procedure involves calculating the FBPC losses for each individual architecture separately and then summing them together to update. This approach allows us to efficiently update the pseudocoreset by considering the contributions of each architecture simultaneously.

To assess the performance of multi-architecture training, we compare a single architecture trained pseudocoreset and a multiple architecture trained pseudocoreset. we specifically focus on investigating the impact of varying normalization layers on the generalizability of the pseudocoreset, since it is widely recognized that a pseudocoreset trained using one architecture may struggle to generalize effectively to a model that employs different normalization layers. For single architecture training, we initially train a pseudocoreset using one architecture with a specific normalization layer, for instance we use instance normalization. Subsequently, we evaluate the performance of this pseudocoreset on four different types of normalization layers: instance normalization, group normalization, layer normalization, and batch normalization. For multiple architecture training, we aggregate four losses for single architecture training of each architecture, and train the pseudocoreset with the sum of all losses, as mentioned above.

As depicted in Figure 3(a), we observe that both WBPC (Weight space Bayesian pseudocoresets) and FBPC-single (Function space Bayesian pseudocoresets trained on a single architecture) exhibit a notable trend, that they tend to not perform well when evaluated on the architecture that incorporates different normalizations, regardless of whether it is trained on weight space or function space. On the other hand, when trained with multiple architectures, both WBPC-multi and FBPC-multi perform well across the all architectures, while notably FBPC-multi significantly outperforms WBPC-multi.



(a) Performance evaluation with different normalization layers. Color represents the test architectures. The multiple architecture FBPC training enhances the generalization ability to other architectures.



(b) A sample image and its corresponding function values. Despite variations in the normalization layers of different architectures, the function values exhibit similarity across the architectures.

Figure 3. Results for multiple architectures FBPC training.

We hypothesize that the superior performance of FBPC compared to WBPC can be attributed to the likelihood of having similar function space posterior across architectures. To validate this, we conduct an examination of the logit values for each sample across different architectures. As an illustration, we provide an example pseudocoreset image belonging to the class label “dog” along with its corresponding logits for all four architectures. As Figure 3(b) shows, it can be observed that the logits display a high degree of similarity, indicating a strong likelihood of matching function posterior distributions. Our analysis confirms that, despite architectural disparities, the function spaces generated by these architectures exhibit significant similarity and it contributes to superiority of FBPC in terms of architecture generalizability.

C.2. Out-of-distribution Inputs Robustness

In Section 4.2, we have shown consistent superiority of FBPC over weight space BPC in terms of OOD robustness. In Table 5, we present the results for 10 different types of corruptions.

D. A Comparison between WBPC and FBPC

By working directly on the function space, FBPC could bypass several challenges that may arise when working on a weight space. Indeed, a legitimate concern arises regarding multi-modality, as the posterior distributions of deep neural networks are highly complex. It makes the optimization of pseudocoresets on weight space difficult. Moreover, minimization of weight space divergence does not necessarily guarantee proximity in the function space. Consequently, although we try to minimize the weight space divergence, there is a possibility that the obtained function posterior may significantly deviate from the true posterior. However, if we directly minimize the divergence between the function distributions, we can effectively address

Table 5. Test accuracy and degradation scores of models trained on each Bayesian pseudocoreset from scratch using SGHMC on the CIFAR10-C. Degradation refers to the extent by which a model's accuracy decreases when evaluated on the CIFAR10-C dataset compared to the CIFAR10 test dataset.

	corruption	BN	DB	ET	FT	GB	JPEG	MB	PIX	SN	SP	Avg.
BPC-fKL	Acc ↑	33.5	34	35.9	25.1	33.7	39.1	32.7	38.3	28.9	41.2	34.3
	Degradation ↓	40.3	39.4	36	55.2	39.9	30.3	41.6	31.6	48.4	26.5	38.9
FBPC	Acc ↑	48.9	46.4	47.6	41.7	44.0	52.0	44.3	51.0	47.1	52.3	**47.5**
	Degradation ↓	21.5	25.7	23.7	33.0	29.3	16.4	28.8	18.1	24.4	16.1	**23.7**

this issue.

On the other hand, there is an additional concern related to memory limitations for WBPC. While it has been demonstrated in Kim et al. (2022) that the memory usage of Bayesian pseudocoresets employing forward KL divergence is not excessively high, we can see that Equation (8) requires Monte-Carlo samples of weights, which requires $\mathcal{O}(Sp)$ where S and p represent the number of Monte-Carlo samples and the dimensionality of the weights, respectively. This dependence on Monte-Carlo sampling poses a limitation for large-scale networks when memory resources are constrained. In contrast, FBPC requires significantly less memory, $\mathcal{O}(Sd)$ where d represents the dimensionality of the functions. Indeed, all the results presented in this paper were obtained using a single NVIDIA RTX-3090 GPU with 24GB VRAM.

E. Related works

E.1. Bayesian coresets

Bayesian Coreset (Campbell & Beronov, 2019; Campbell & Broderick, 2018; 2019; Huggins et al., 2016) is a field of research aimed at addressing the computational challenges of MCMC and VI on large datasets in terms of time and space complexity (Chen et al., 2014; Baker et al., 2017; Quiroz et al., 2018). It aims to approximate the energy function of the entire dataset using a weighted sum of a small subset. However for high-dimensional data, Manousakas et al. (2020) demonstrates that considering only subsets as Bayesian coreset is not sufficient, as the KL divergence between the approximated coreset posterior and the true posterior increases with the data dimension, and they proposed Bayesian pseudocoresets. There are recent works on constructing pseudocoreset variational posterior to be more flexible (Chen et al., 2022) or how to effectively optimize the divergences between posteriors (Kim et al., 2022; Chen et al., 2022; Manousakas et al., 2022; Naik et al., 2023). However, there is still a limitation in constructing high-dimensional Bayesian pseudocoresets specifically for deep neural networks.

E.2. Dataset distillation

Dataset distillation also aims to synthesize the compact datasets that capture the essence of the original dataset. However, the dataset distillation places particulary on optimizing the test performance of the distilled dataset. Consequently, the primary objective in dataset distillation is to maximize the performance of models trained using the distilled dataset, and researchers provides how to effectively solve this bi-level optimization (Wang et al., 2018; Nguyen et al., 2021; 2020; Zhou et al., 2022; Zhao & Bilen, 2021; Zhao et al., 2021; Cazenavette et al., 2022). In recent work, Kim et al. (2022) established a link between specific dataset distillation methods and optimizing certain divergence measures associated with Bayesian pseudocoresets.

E.3. Function space variational inference

Although Bayesian neural networks exhibit strong capabilities in performing variational inference, defining meaningful priors or efficiently inferring the posterior on weight space is still challenging due to their over-parametrization. To overcome this issue, researchers have increasingly focused on function space variational inference (Carvalho et al., 2020; Burt et al., 2020; Ma & Hernández-Lobato, 2021; Rudner et al., 2022b; Pan et al., 2020b; Titsias et al., 2020). For instance, Sun et al. (2019) introduced a framework that formulates the KL divergence between functions as the supremum of marginal KL divergences over finite sets of inputs. Wang et al. (2019) utilizes particle-based optimization directly in the function space. Furthermore, Rudner et al. (2022c) recently proposed a scalable method for function space variational inference on deep neural networks.