EFFICIENTLY WARMSTARTING MCMC FOR BNNS

David Rundel, Emanuel Sommer, Bernd Bischl, David Rügamer, Matthias Feurer
Department of Statistics, LMU Munich, Munich, Germany
Munich Center for Machine Learning, Munich, Germany
{firstname.lastname}@stat.uni-muenchen.de

ABSTRACT

Markov Chain Monte Carlo (MCMC) algorithms are widely regarded as the gold standard for approximate inference in Bayesian neural networks (BNNs). However, they remain computationally expensive and prone to inefficiencies, such as *dying samplers*, frequently leading to substantial waste of computational resources. While prior work has presented warmstarting techniques as an effective method to mitigate these inefficiencies, we provide a more comprehensive empirical analysis of how initializations of samplers affect their behavior. Based on various experiments examining the dynamics of warmstarting MCMC, we propose novel warmstarting strategies that leverage performance predictors and adaptive termination criteria to achieve better-performing, yet more cost-efficient, models. In numerical experiments, we demonstrate that this approach provides a practical pathway to more resource-efficient approximate inference in BNNs.

1 INTRODUCTION

Recent advancements in deep learning (DL) have led to models achieving outstanding predictive accuracy across diverse tasks. However, these models often fail to provide reliable uncertainty quantification (UQ), limiting their broader applicability. Bayesian deep learning (BDL) offers a compelling framework for addressing this challenge, with Markov Chain Monte Carlo (MCMC) being one widely used approach to approximate inference (Neal, 2012a; Papamarkou et al., 2024). Nevertheless, MCMC-based inference in Bayesian neural networks (BNNs) is computationally expensive and susceptible to inefficiencies (Izmailov et al., 2021). To address this, prior work has proposed warmstarting, where MCMC samplers are initialized with specifically chosen starting points, and demonstrated its effectiveness in preventing *dying samplers* (Sommer et al., 2024) - a phenomenon where samplers become trapped in regions of very low probability after initialization. However, a comprehensive analysis of how warmstarting impacts both the UQ performance of sampling algorithms and their associated computational costs is lacking.

Our Contributions: In this work, we close this gap by investigating the dynamics of warmstarting MCMC and how they relate to properties of the posterior. By first analyzing this through an extensive set of numerical experiments, we find that the performance of samplers and their functional outputs can be linked to the characteristics of the initial proposals. Furthermore, we are the first to link the potential performance gains of warmstarting to its associated computational cost, identifying significant pipeline *backloading*. These insights motivate the development of novel warmstarting strategies that leverage performance predictors and adaptive termination criteria to yield models with improved performance at lower computational cost. Finally, we provide empirical evidence validating the effectiveness of our approach.

2 BACKGROUND AND RELATED WORK

Let $\mathcal{D}_{train} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n_{train}}$ represent a training dataset consisting of n_{train} independent and identically distributed (*iid*) observations, where each $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^p$ is a p-dimensional feature vector and $y_i \in \mathcal{Y}$ is the corresponding target. In supervised machine learning (ML), the objective is to learn a model $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that accurately approximates the conditional distribution $p_0(y^* \mid \boldsymbol{x}^*)$ of the underlying data generating process (DGP) for previously unseen $\boldsymbol{x}^*$. In this work we focus on deep

neural networks (DNNs) $\hat{f}_{\boldsymbol{\theta}}$, parameterized by $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$. BDL approximates $p_0(y^* \mid \boldsymbol{x}^*)$ by the posterior predictive density (PPD) $p(y^* \mid \boldsymbol{x}^*, \mathcal{D}_{train})$, which integrates over the posterior distribution of the model parameters $\boldsymbol{\theta}$.

2.1 Background

However, exact Bayesian inference is typically intractable, and designing approximate inference methods presents a significant challenge: Contemporary DNNs give rise to posterior surfaces that are extremely high-dimensional and multi-modal, making exhaustive exploration intractable and increasing the risk of neglecting important regions. Furthermore, effectively disconnected posterior modes (Yao et al., 2022; Sommer et al., 2024) and posterior symmetries (Wiese et al., 2023; Hecht-Nielsen, 1990), potentially leading to redundant computations that do not improve predictive uncertainty, further exacerbate this issue (Neal, 2012a; Papamarkou et al., 2024).

2.2 Markov Chain Monte Carlo

MCMC unites a class of algorithms for approximate inference, designed to generate Markov chains with stationary distribution corresponding to the posterior distribution. Specifically, given an initial proposal (IP) $\boldsymbol{\theta}^{(0)} \in \Theta$, a sampling algorithm produces approximate posterior samples collected in $\mathbb{S} = \{\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(S)}\} \in \Theta^S$. The algorithms operate in two phases: an initial warmup phase, during which samples are generated to ensure effective mixing of the chains but are later discarded, and the main sampling phase, where the actual posterior samples are drawn (Gelman et al., 2013). Compared to alternative approximate inference methods for BNNs, MCMC is often considered the gold standard because it does not rely on restrictive assumptions about the posterior distribution (Farquhar et al., 2020). However, this advantage comes at the cost of significantly increased computational cost. State-of-the-art (SOTA) algorithms like Hamiltonian Monte Carlo (HMC) (Neal, 2012b) require extensive warmup phases and must repeatedly iterate over the entire training dataset to generate a single approximate posterior sample (Izmailov et al., 2021). Nevertheless, phenomena such as chains becoming stuck in isolated regions or the *dying sampler* problem (Sommer et al., 2024) — where chains become trapped in regions of near-zero probability from which they were initialized — suggest that, despite the high computational burden, the true posterior distribution is often not efficiently recovered and a significant amount of computational resources is wasted.

Motivated by this, Sommer et al. (2024) proposed Deep Ensemble Initialized MCMC (DEI-MCMC), a strategy that leverages multiple chains simultaneously to explore multimodal surfaces more effectively. Empirical evidence has shown that this approach often generates samples with greater functional diversity compared to those from single chains, thereby improving UQ performance (Wiese et al., 2023; Fort et al., 2019). The method first optimizes several DNNs and uses their parameters as IPs for the parallel MCMC runs in the subsequent sampling phase — a process known as *warmstarting*. With this approach, initializing samplers in regions of high posterior density rather than using random IPs, issues related to *dying samplers* can be avoided, and the length of the warmup phase can be significantly reduced. The detailed algorithm is outlined in Appendix A.1. However, this approach still has several limitations: Although warmstarting samplers proves to be an efficient strategy, there is no guarantee that the samplers will explore the posterior in regions that contribute most to the PPD or capture functional diversity that enables effective interactions across chains.

3 Analyzing MCMC Warmstarting Dynamics

In the following section, we will explore typical dynamics of warmstarting MCMC runs with optimized DNNs. We consider the Bike Sharing dataset (Fanaee-T, 2013), a medium-sized regression task, modeled with a simple multi-layer perceptron (MLP) and sampled using the No-U-Turn Sampler (NUTS) (Hoffman et al., 2014), a SOTA full-batch sampling algorithm. This preliminary analysis provides valuable insights that motivate our subsequent methodology for enhancing DEI-MCMC.

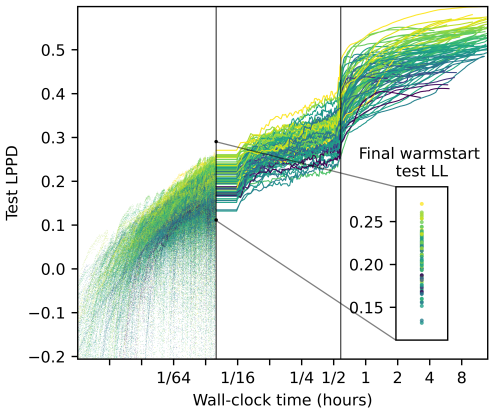

Figure 1: Performance difference evolution across various warm-started MCMC runs. Vertical bars separate the phases, and an inset plot zooms into the final warmstart step. The runs are colored according to their final performance.

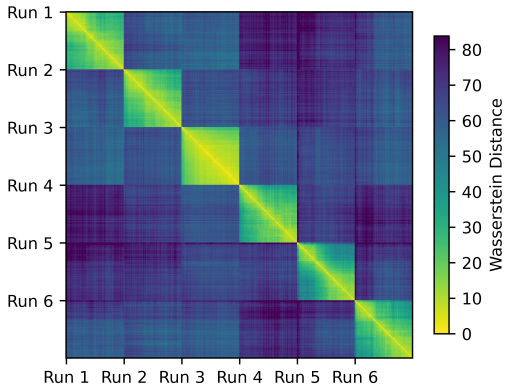

Figure 2: Functional diversity, measured by the Wasserstein distance of predictive distributions, between different steps within and between six warmstarted MCMC runs, each consisting of 100 warmup and 200 sampling steps. The axes are sorted by run, phase, and step.

3.1 Performance differences

In Figure 1, we illustrate the evolution of UQ performance across different phases and 100 warmstarted MCMC runs. For each run, the warmstart phase lasts for 4096 epochs, followed by a warmup phase of 100 steps, as proposed by Sommer et al. (2024), and then a sampling phase of 2048 steps. The x-axis shows the wall-clock time in hours on a logarithmic scale, with vertical bars separating the three phases. The y-axis shows the corresponding log pointwise predictive density (LPPD) (Equation 4 in Appendix A.3) on the test set $\mathcal{D}_{test}$ at each step, computed using all previous approximate posterior samples from the respective chain during the sampling phase and only the current parameter state in the preceding phases. Furthermore, each run is colored according to the LPPD of the entire chain.

Generally, after the random initialization of DNNs in regions of low LPPDs, optimization elevates all runs into areas of substantially higher LPPDs, with each run converging at slightly different levels. During the warmup phase of the DNN-initialized MCMC runs, another slight increase in LPPDs is observed. Subsequently, in the sampling phase, the samplers achieve even higher UQ performance and exhibit convergence. This can be attributed to the marginalization over multiple approximate posterior samples rather than considering single parameter states. The final UQ performance of the chains varies significantly between runs. Moreover, while the ultimate ranking of chains differs from their ranking in earlier phases, as indicated by the color gradient across runs at each step, there appears to be a relationship with the ranking of partial chains and the rankings in the warmup phase. Notably, as indicated by the inset plot (bottom right), which displays the final step of the warmstart phase in isolation, there even seems to be a link between the final performances of the warmstart phase and the final chain performances.

3.2 Functional diversity

Furthermore, we investigate the functional diversity between different steps within and between chains. Therefore, we consider six warmstarted MCMC runs with warmup and sampling phases of 100 and 200 steps, respectively. For each run and step across phases, we compute the predictive distribution over the test set and estimate the functional diversity between all pairs using the Wasserstein-2 distances between the associated predictive distributions.

Figure 2 visualizes the Wasserstein distance between predictive distribution pairs, with axes sorted by run, phase, and step, and colors indicating distance. Generally, one can observe that functional

diversity is typically much higher between runs than within runs. This suggests that, at least for a limited number of steps, the samplers remain largely confined to a local subspace of the functional output space, which is related to the functional outputs from the initialization.

3.3 BACKLOADING

As observed in Figure 1, the time required for the warmstarting phase accounts for only a fraction of the subsequent warmup and sampling times: While the warmstarting phase takes approximately 2 minutes (141 seconds), the combined warmup and sampling phases require, on average, more than 9 hours (33,762 seconds). This difference arises from the distinct algorithms used in these phases: During the latter phases, NUTS is employed, which performs several integration steps for each posterior sample. In popular implementations such as BlackJAX (Cabezas et al., 2024), PyMC (Abril-Pla et al., 2023), and Stan (Carpenter et al., 2017), the default configuration allows up to 1,024 integration steps, each requiring gradient computation of the posterior with respect to all model parameters for the entire dataset. In contrast, during warmstarting, a well-performing DNN can be trained in our experiments with only thousands of optimization steps, each requiring just a single gradient evaluation on the dataset. Consequently, the warmstarting phase is significantly cheaper than the sampling phase, a phenomenon we refer to as *backloading* of the pipeline.

4 METHODOLOGY

Previous findings clearly indicate the potential to improve DEI-MCMC's efficiency and motivate the following exposition.

4.1 EXAMINING ANYTIME PERFORMANCE

Existing work has primarily evaluated DEI-MCMC from an UQ performance perspective, without considering its computational cost. We suggest instead to frame it as a multi-objective optimization problem, relating the potential performance gains of warmstarting to its computational overhead. This consideration is particularly important, as the methods introduced later in this work aim to further enhance warmstarting approaches but at the cost of even greater computational overhead. Using wall-clock time as a proxy for computational cost, we propose considering the anytime performance of DEI-MCMC. This refers to the progression of performance — such as UQ performance measured by LPPD on an unseen test set — over time, incorporating all approximate posterior samples generated up to that point. Hence, it indicates a method's ability to produce high-quality results at any stage of its execution, rather than solely at a single point.

4.2 CONFIGURING WARMSTARTS

Our empirical results in Section 3 suggest a correlation between warmstarting and sampler behavior that goes beyond the mere occurrence of *dying samplers* but extends to the overall performance and functional characteristics of chains. In this work, we propose a method to exploit these findings by identifying warmstarting configurations that, across diverse tasks, improve the anytime performance of DEI-MCMC. This can be regarded as an algorithm configuration problem (Hutter et al., 2009).

Algorithm configuration for DEI-MCMC While in the vanilla variant of DEI-MCMC, the warmstarting procedure consists of sequential Hyperparameter optimization (HPO) and the optimization of M DNNs, we aim to exploit the significant backloading of the pipeline. We propose considering a larger number of IP candidates and strategically selecting an optimal subset for sampling. With this approach, employing adaptive termination criteria, we can significantly increase the number of considered IP candidates while greatly reducing computational cost compared to executing the entire sampling process for all candidates. This may be effective, as the enhanced set of IP candidates could include IPs that lead to better-performing chains and chains that capture a broader range of functional diversity, potentially interacting more effectively.

Optimization process We outline the proposed algorithm in Appendix A.2. Specifically, for each task, we first optimize an expanded set of $M' > M$ IP candidates, denoted as $\mathbb{M}' =$

$\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M')}\}$. These candidates give rise to a variety of potential IP set candidates $\mathbb{M} \subseteq \mathbb{M}'$ with $|\mathbb{M}| = M$ for DEI-MCMC with M chains (while discarding all other candidates). We then aim to select the IP set candidate $\mathbb{M}^*$ that maximizes downstream performance under a fixed computational budget $t \in \mathbb{R}^+$. Mathematically, this can be formulated as

$$\mathbb{M}^* \in \underset{(\mathbb{M} \subseteq \mathbb{M}' : |\mathbb{M}| = M)}{\arg\max} \left[\rho_{\textit{(LPPD, t)}} \Big(\mathcal{A}_s \big(\mathbb{M}, \boldsymbol{\lambda}_s, \mathcal{D}_{\textit{train}} \big), \mathcal{D}_{\textit{test}} \Big) \right], \tag{1}$$

where $\mathcal{D}_{\textit{test}} = \mathcal{D} \setminus \mathcal{D}_{\textit{train}}$, $\mathcal{A}_s : \Theta^M \times \Lambda_s \times \mathbb{D} \to (\Theta^S)^M$ is a parallel sampling procedure that, given M IPs, a hyperparameter (HP) configuration $\boldsymbol{\lambda}_s \in \Lambda_s$, and a training split $\mathcal{D}_{\textit{train}}$, generates S posterior samples for each of the M chains, and the LPPD on the unseen test set at a given time step t is defined via the performance measure $\rho_{\textit{(LPPD, t)}}$. However, since leaking the test dataset is not admissible and repeatedly evaluating $\mathcal{A}_s$, thus executing the entire sampling process, is prohibitively expensive, we propose resorting to proxy optimization instead:

$$\mathbb{M}^* \approx \underset{(\mathbb{M} \subseteq \mathbb{M}' : |\mathbb{M}| = M)}{\arg\max} \Big[\rho_{\textit{PP}}(\mathbb{M}, \mathcal{D}_{\textit{val}}) \Big]. \tag{2}$$

Proxy metric As a proxy-metric, we propose utilizing low-fidelity performance predictors $\rho_{\textit{PP}} : \Theta^M \times \mathbb{D} \to \mathbb{R}$, which estimate the final performance of IP set candidates using only a previously unseen validation split $\mathcal{D}_{\textit{val}}$ (of the training data) and the parameter states of the IPs. Specifically, we propose the following proxy-LPPD-based performance predictor:

$$\rho_{\textit{PP}}\left(\mathbb{M}, \mathcal{D}_{\textit{val}}\right) = \frac{1}{n_{\textit{val}}} \sum_{(\boldsymbol{x}^*, y^*) \in \mathcal{D}_{\textit{val}}} \log \Big(\frac{1}{M} \sum_{\tilde{\boldsymbol{\theta}} \in \mathbb{M}} p(y^* \mid \boldsymbol{x}^*, \tilde{\boldsymbol{\theta}}) \Big). \tag{3}$$

It is closely related to our downstream UQ performance metric (Equation 4 in Appendix A.3), as it approximates the LPPD of an ensemble of chains on a test set by computing the same metric on a validation set and using only a single proxy posterior sample for each chain — namely, the last parameter state of the warmstarting phase. Thereby, it does not only consider IP candidates in isolation but also take interactions into consideration.

Ensemble selection via greedy search Furthermore, since an exhaustive search for the combinatorial optimization problem in Equation 2 quickly becomes infeasible, we instead apply ensemble selection algorithms (Caruana et al., 2004) that select among IP set candidates using the performance predictors in a feasible way. We propose a greedy search as follows: After initializing $\mathbb{M}$ as empty, at each of the M steps, the IP candidate that yields the largest performance improvement in the performance predictor, computed on the current $\mathbb{M}$, is added to $\mathbb{M}$.

5 Experiments

Setup We conduct 100 independent warmstarted MCMC runs on the Bike Sharing dataset (Fanaee-T, 2013) and the Protein Tertiary Structure dataset (Rana, 2013) — both medium-sized regression tasks modeled with simple MLPs and sampled using NUTS (Hoffman et al., 2014). We use the experimental setup as delineated in Appendix A.3, including a warmstart phase of 4096 epochs, a warmup phase of 100 steps, and a sampling phase of 2048 steps for each run. We then repeatedly sample random subsets of runs of size M'. In each trial, we compare the performance of our proposed method (referred to as "Ours" in the following) to that of the vanilla variant of DEI-MCMC (referred to as "Baseline"). Specifically, for our proposed method, in each trial, we select M out of the M' runs according to our proposed performance predictors (Equation 3), coupled with the proposed greedy search algorithm (Subsection 4.2). To simulate the behavior of vanilla DEI-MCMC (Algorithm outlined in Appendix A.1), which optimizes only M IPs, we simply select the first M runs of each trial. Furthermore, for each specified M, we assume a parallel computing environment with M parallel cores, each independently performing warmstarted MCMC runs.

Results Figure 3 depicts the mean anytime LPPDs of the approaches for various combinations of M and M' across 100 trials on the Bike Sharing dataset. An equivalent plot for the Protein Tertiary Structure dataset is provided in Appendix A.4. The x-axis indicates the wall-clock time in

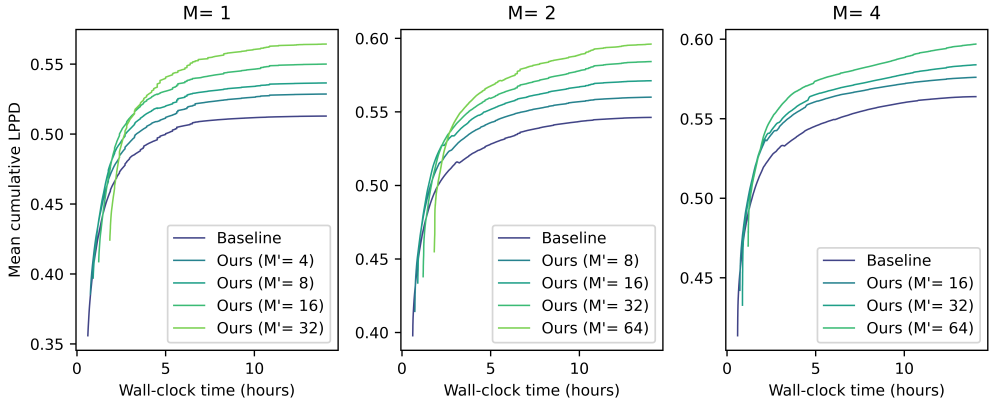

Figure 3: Mean anytime LPPDs (across 100 trials) of our proposed method compared to the baseline (vanilla DEI-MCMC) for various combinations of M and M' on the Bike Sharing dataset.

seconds. Due to extended warmstarting procedures, the configurations of our proposed method start generating posterior samples at a later stage than the baselines. However, across all configurations, our proposed method surpasses the respective baseline after a relatively short amount of time and seems to converge at higher LPPD levels.

In Table 1, we compare the final mean LPPDs across all configurations and trials on the Bike Sharing dataset and the Protein Tertiary Structure dataset. The results suggest that across tasks, for all considered variants of M and up to the investigated values of M', the ultimate performance increases monotonically with M'. These findings indicate that, despite the computational overhead — due to backloading and the ability to predict highly performant chains — a large number of IP candidates is worthwhile.

6 Conclusion

In this work, we investigated the causes of the computational cost in MCMC-based inference for BNNs by analyzing the dynamics of warmstarting Markov chains. We found that performance predictors can be used in order to determine better-performing chains when the computational budget is limited. We demonstrated the potential of our proposed method to achieve better-performing yet more cost-efficient MCMC-based approximate inference in BNNs.

Table 1: LPPD mean and standard deviation (across 100 trials) for different M values and configurations of our proposed method compared to the baseline (DEI-MCMC), after approximately 14 hours (50,546 seconds) on the Bike Sharing dataset and approximately 34 hours (123,595 seconds) on the Protein Tertiary Structure dataset. For our proposed method, the value of M' is given by the product $M \cdot c$. The highest mean value for each M is highlighted in bold.

Dataset	M	**Baseline** ($c = 1$)	**Ours** $c = 4$	$c = 8$	$c = 16$	$c = 32$
Bike	1	0.513 ± 0.041	0.529 ± 0.029	0.536 ± 0.032	0.550 ± 0.031	**0.564** ± 0.024
	2	0.546 ± 0.023	0.560 ± 0.019	0.571 ± 0.019	0.584 ± 0.016	**0.596** ± 0.013
	4	0.564 ± 0.013	0.576 ± 0.012	0.584 ± 0.011	**0.597** ± 0.007	—
Protein	1	-0.869 ± 0.022	-0.866 ± 0.018	-0.863 ± 0.018	-0.859 ± 0.017	**-0.857** ± 0.019
	2	-0.816 ± 0.013	-0.814 ± 0.012	-0.810 ± 0.010	-0.808 ± 0.011	**-0.803** ± 0.008
	4	-0.789 ± 0.009	-0.785 ± 0.008	-0.783 ± 0.008	**-0.778** ± 0.007	—

Future work There are three main directions for future research that follow from our work: 1) More advanced warmstarting configurations, potentially incorporating more sophisticated performance predictors and ensemble selection algorithms, could result in even greater cost savings (or, equivalently, increased performance under constraints). 2) Exploring these effects in large-scale, structured studies using, for example, algorithm configuration approaches such as SMAC (Hutter et al., 2011) could provide highly valuable insights for practitioners, as the computational costs of MCMC-based inference still prevent its application to large-scale computer vision or language models and bigger datasets. 3) Lastly, from an anytime performance perspective, it may be worthwhile to merge the sequential HPO and DNN optimization phases of DEI-MCMC into a single stage, where models trained during HPO are recycled as IP candidates.

REFERENCES

Oriol Abril-Pla, Virgile Andreani, Colin Carroll, Larry Dong, Christopher J Fonnesbeck, Maxim Kochurov, Ravin Kumar, Junpeng Lao, Christian C Luhmann, Osvaldo A Martin, et al. PyMC: a modern, and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science*, 9:e1516, 2023.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2), 2012.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Alberto Cabezas, Adrien Corenflos, Junpeng Lao, and Rémi Louf. Blackjax: Composable Bayesian inference in JAX, 2024.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76, 2017.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *International Conference on Machine Learning*, pp. 18, 2004.

Hadi Fanaee-T. Bike Sharing. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5W894.

Sebastian Farquhar, Lewis Smith, and Yarin Gal. Liberty or depth: Deep Bayesian neural nets do not need complex weight posterior approximations. *Advances in Neural Information Processing Systems*, 33:4346–4357, 2020.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. Bayesian data analysis. 2013.

Andrew Gelman, Jessica Hwang, and Aki Vehtari. Understanding predictive information criteria for Bayesian models. *Statistics and computing*, 24:997–1016, 2014.

Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pp. 129–135. Elsevier, 1990.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL `http://github.com/google/flax`.

Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference*, pp. 507–523. Springer, 2011.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are Bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pp. 4629–4640. PMLR, 2021.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012a.

Radford M Neal. MCMC using Hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012b.

Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, José Miguel Hernández-Lobato, et al. Position: Bayesian Deep Learning is Needed in the Age of Large-Scale AI. In *International Conference on Machine Learning*, 2024.

Prashant Rana. Physicochemical Properties of Protein Tertiary Structure. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5QW3H.

Emanuel Sommer, Lisa Wimmer, Theodore Papamarkou, Ludwig Bothmann, Bernd Bischl, and David Rügamer. Connecting the dots: Is mode-connectedness the key to feasible sample-based inference in bayesian neural networks? In *International Conference on Machine Learning*, pp. 45988–46018. PMLR, 2024.

Jonas Gregor Wiese, Lisa Wimmer, Theodore Papamarkou, Bernd Bischl, Stephan Günnemann, and David Rügamer. Towards efficient MCMC sampling in Bayesian neural networks by exploiting symmetry. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 459–474. Springer, 2023.

Yuling Yao, Aki Vehtari, and Andrew Gelman. Stacking for non-mixing Bayesian computations: The curse and blessing of multimodal posteriors. *Journal of Machine Learning Research*, 23(79): 1–45, 2022.

Erik Štrumbelj, Alexandre Bouchard-Côté, Jukka Corander, Andrew Gelman, Håvard Rue, Lawrence Murray, Henri Pesonen, Martyn Plummer, and Aki Vehtari. Past, Present and Future of Software for Bayesian Inference. *Statistical Science*, 39(1):46 – 61, 2024.

A Appendix

A.1 Deep Ensemble Initialized MCMC

Algorithm 1 DEI-MCMC (Sommer et al., 2024)

Require: Training dataset $\mathcal{D}_{train}$, HPO configuration $\boldsymbol{\lambda}_{hpo}$, Sampling configuration $\boldsymbol{\lambda}_s$

▷ *First phase: Warmstarting*
1: $\boldsymbol{\lambda}^*_{de} = \mathcal{A}_{hpo}(\boldsymbol{\lambda}_{hpo}, \mathcal{D}_{train})$ ▷ DNN HPO
2: $\mathbb{M} = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}\} = \mathcal{A}_{de}(\boldsymbol{\lambda}^*_{de}, \mathcal{D}_{train})$ ▷ DNN Optimization

▷ *Second phase: Sampling*
3: $\mathbb{S} = \mathcal{A}_s(\mathbb{M}, \boldsymbol{\lambda}_s, \mathcal{D}_{train})$ ▷ Warmup & Sampling

For DEI-MCMC, HPO is initially performed on the training dataset via $\mathcal{A}_{hpo} : \Lambda_{hpo} \times \mathbb{D} \to \Lambda_{de}$ to identify well-suited HPs $\boldsymbol{\lambda}^*_{de} \in \Lambda_{de}$ for DNNs optimization. Thereby, $\boldsymbol{\lambda}_{hpo} \in \Lambda_{hpo}$ defines the configuration of the HPO routine and its search space. Next, M DNNs are independently optimized on the training dataset according to $\boldsymbol{\lambda}^*_{de}$, each with a different random initialization of the parameters, following $\mathcal{A}_{de} : \Lambda_{de} \times \mathbb{D} \to \Theta^M$. This results in a so-called deep ensemble (DE) (Lakshminarayanan et al., 2017), collected in $\mathbb{M} = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}\}$, which serves as the set of IPs for M parallel MCMC runs in the subsequent phase — a process known as *warmstarting*. The parallel sampling procedure $\mathcal{A}_s : \Theta^M \times \Lambda_s \times \mathbb{D} \to (\Theta^S)^M$ then generates M chains, each producing S approximate posterior samples following a warm-up phase of W steps, as specified by the HP configuration $\boldsymbol{\lambda}_s \in \Lambda_s$.

A.2 Advanced DEI-MCMC

Algorithm 2 Advanced DEI-MCMC (ours)

Require: Training dataset $\mathcal{D}_{train}$, HPO configuration $\boldsymbol{\lambda}_{hpo}$, Sampling configuration $\boldsymbol{\lambda}_s$

▷ *First phase: Warmstarting*
1: $\boldsymbol{\lambda}^*_{de} = \mathcal{A}_{hpo}(\boldsymbol{\lambda}_{hpo}, \mathcal{D}_{train})$ ▷ DNN HPO
2: $\mathbb{M}' = \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M')}\} = \mathcal{A}_{de}(\boldsymbol{\lambda}^*_{de}, \mathcal{D}_{train})$ ▷ Extended DE Optimization
3: $\mathbb{M} = \mathcal{A}_{es}(\mathbb{M}', \mathcal{D}_{train})$ ▷ Ensemble Selection

▷ *Second phase: Sampling*
4: $\mathbb{S} = \mathcal{A}_s(\mathbb{M}, \boldsymbol{\lambda}_s, \mathcal{D}_{train})$ ▷ Warmup & Sampling

In our proposed method to advance DEI-MCMC, we first optimize an expanded set of $M' > M$ IP candidates, $\mathbb{M}'$, and then select a subset $\mathbb{M}$ for sampling using an ensemble selection algorithm $\mathcal{A}_{es} : \Theta^{M'} \times \mathbb{D} \to \Theta^M$.

A.3 Experimental Setup

Tasks We consider the Bike Sharing dataset (Fanaee-T, 2013) and the Protein Tertiary Structure dataset (Rana, 2013) as simple regression tasks that fall within the medium data-size regime, containing 17,379 and 45,730 observations with 13 and 9 numerical features, respectively. The datasets are split into a training set, $\mathcal{D}_{train}$, a validation set, $\mathcal{D}_{val}$, and a test set, $\mathcal{D}_{test}$, with the validation and test sets each comprising 3,000 observations for the former dataset and 7,500 observations for the latter, respectively.

Models We employ simple MLPs with ReLU activation functions and two hidden layers, each containing 16 neurons. To model the natural parameters of Gaussian likelihoods with unknown variance, we use two output heads.

Warmstarting In the optimization procedure for warmstarting, we utilize the AdamW optimizer (Loshchilov, 2017) in conjunction with the negative log likelihood (NLL) of a heteroscedastic Gaussian distribution as the loss function. To enhance regularization, we apply weight decay, early stopping, and dropout. The associated hyperparameters — as well as batch size, learning rate, and weight decay — are tuned via simple random search (Bergstra & Bengio, 2012). Additionally, we incorporate gradient clipping to improve training stability.

Sampling For sampling, we rely on NUTS (Hoffman et al., 2014), a variant of HMC. We choose this method because it is considered a SOTA full-batch sampling algorithm (Štrumbelj et al., 2024) and because it automatically tunes its HPs during the warmup phase, largely eliminating the need for additional HPO in the sampling phase. We use a fixed target acceptance probability of 0.8, allow up to 1,024 integration steps, and adopt unit Gaussian priors to reflect the absence of strong assumptions about the relationships between model parameters and outputs.

Performance measures To evaluate chains, we assess UQ performance using the log pointwise predictive density (LPPD) over the test set (Gelman et al., 2014), defined as

$$\frac{1}{n_{test}} \sum_{(\boldsymbol{x}^*, y^*) \in \mathcal{D}_{test}} \log\left(\frac{1}{(|\mathbb{S}|)} \sum_{\tilde{\boldsymbol{\theta}} \in \mathbb{S}} p(y^* \mid \boldsymbol{x}^*, \tilde{\boldsymbol{\theta}})\right). \tag{4}$$

Additionally, we track the wall-clock time for each optimization and sampling step.

Implementation The implementation is written in Python, primarily leveraging JAX (Bradbury et al., 2018), Flax (Heek et al., 2023) and BlackJAX (Cabezas et al., 2024). Furthermore, the experiments were conducted on a single CPU instance with 32 cores and 64GB of RAM.

A.4 Experimental Results

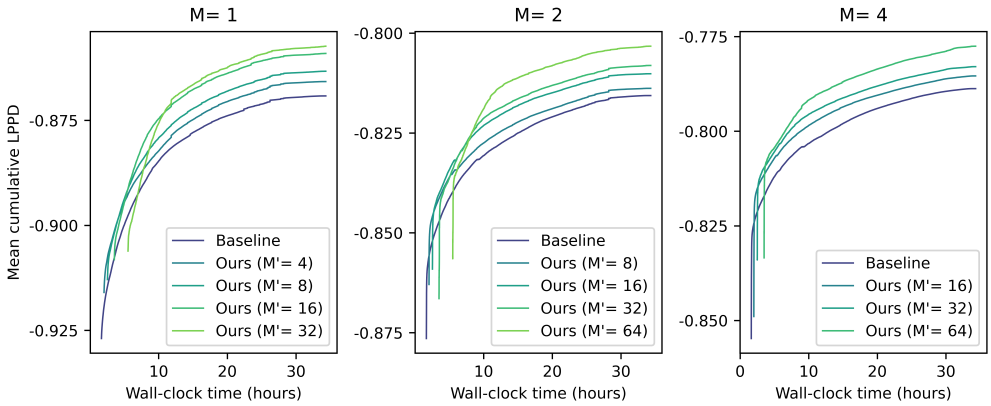

Figure 4: Mean anytime LPPDs (across 100 trials) of our proposed method compared to the baseline for various combinations of M and M' on the Protein Tertiary Structure dataset.

Figure 4 depicts the mean anytime LPPDs of the DEI-MCMC variants for various combinations of M and M' across 100 trials on the Protein Tertiary Structure dataset. As with the results for the Bike Sharing dataset (Section 5), our proposed method surpasses the respective baseline after a relatively short amount of time and appears to converge at higher LPPD levels across configurations.

B LIST OF ABBREVIATIONS

BDL	Bayesian deep learning
BNN	Bayesian neural network
DEI-MCMC	Deep Ensemble Initialized MCMC
DE	deep ensemble
DL	deep learning
DGP	data generating process
DNN	deep neural network
HMC	Hamiltonian Monte Carlo
HP	hyperparameter
HPO	Hyperparameter optimization
MCMC	Markov Chain Monte Carlo
ML	machine learning
MLP	multi-layer perceptron
NUTS	No-U-Turn Sampler
NLL	negative log likelihood
PPD	posterior predictive density
SOTA	State-of-the-art
UQ	uncertainty quantification
IP	initial proposal
LPPD	log pointwise predictive density