SCALABLE BAYESIAN DEEP KERNEL NETWORKS VIA ENSEMBLE-INDUCED FINITE FUNCTION SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce Bayesian Deep Kernel Networks, a framework for predictive regression designed to enhance interpretability while maintaining competitive predictive performance and computational efficiency. Our method combines the statistical rigor of Bayesian inference with the scalability of deep ensembles, providing calibrated uncertainty estimates that enable its use not only for standalone prediction but also as a component within broader learning systems. To achieve these goals, our work relies on three key design components: **(i) low-dimensional ensemble representation:** predictions are expressed as a combination of a small number of trained neural predictors, enabling scalable inference whose cost depends on ensemble size rather than dataset size; **(ii) closed-form Bayesian aggregation:** ensemble predictions are combined using Bayesian linear regression, yielding interpretable posterior weights and calibrated uncertainty without approximate inference; and **(iii) Independent ensemble training:** multiple neural networks are trained separately, producing diverse predictive representations that improve robustness and uncertainty calibration. Empirical results on standard regression benchmarks demonstrate that the proposed approach achieves competitive predictive performance while maintaining reliable uncertainty estimates across settings.

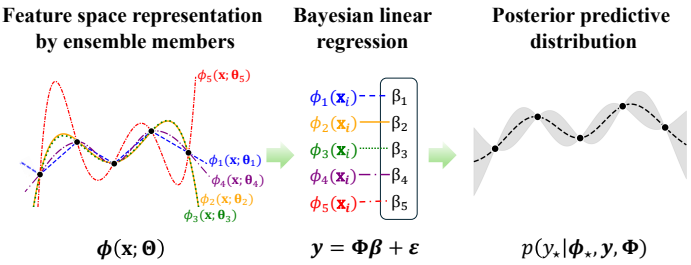


Figure 1: **Proposed framework.** An ensemble of neural networks induces a feature space representation $\{\phi_h(\mathbf{x}_i;\boldsymbol{\theta}_h)\}_{h=1}^{H}$. A Bayesian linear model is placed on top of these ensemble-induced features, and the linear weights $\boldsymbol{\beta}$ and noise variance σ_ε^2 are integrated out with respect to their posteriors, yielding a closed-form posterior predictive distribution for a target $y_\star$.

1 INTRODUCTION

We consider predictive regression models in settings where reliable uncertainty estimates are required alongside **(i)** computational efficiency, **(ii)** strong predictive performance, and **(iii)** interpretability. This requirement is particularly important in safety-critical and risk-sensitive applications (Filos et al., 2019), where deep neural networks (NNs) are known to produce poorly calibrated predictions that can be overconfident (Guo et al., 2017; Lakshminarayanan et al., 2017; Watson et al., 2021), especially under distribution shift. Moreover, regression models with quantified uncertainty serve as core components in broader learning systems, including reinforcement learning (Sutton et al., 1998; Arulkumaran et al., 2017; Levine et al., 2020; Deisenroth & Rasmussen, 2011) and physics-informed neural networks (Raissi et al., 2019). Goals **(i)**–**(iii)** are inherently in tension. In what follows, we examine several widely used paradigms that address these requirements to varying degrees while exposing important trade-offs.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In terms of predictive performance and principled uncertainty estimation, Gaussian Processes (GPs) provide a canonical Bayesian framework for nonlinear regression with exact posterior inference (Williams & Rasmussen, 2006). However, their cubic computational complexity and sensitivity to kernel choice fundamentally limit scalability, particularly in high-dimensional settings where distance-based kernels suffer from the curse of dimensionality. Deep Kernel Learning (DKL) partially mitigates this limitation by learning data-dependent covariance functions using neural networks, but exact inference remains computationally intensive, and scalable implementations typically rely on approximations (Wilson et al., 2016).

Fully Bayesian neural networks (BNNs) represent a different point in the performance–computation trade-off by placing probability distributions over network parameters (Neal, 2012; Wilson & Izmailov, 2020). While they provide a flexible probabilistic framework for modeling uncertainty, exact inference is generally intractable, and practical implementations rely on approximate inference methods such as mean-field variational inference (Graves, 2011). These approximations can impose restrictive assumptions on the posterior distribution, which may lead to miscalibrated uncertainty estimates (Foong et al., 2020; Wenzel et al., 2020).

Bayesian last-layer (BLL) methods address the computational limitations of fully Bayesian neural networks by restricting Bayesian inference to the final linear layer of a neural network. This design enables efficient inference while retaining a probabilistic interpretation of predictions (Lázaro-Gredilla & Figueiras-Vidal, 2010). However, because uncertainty is placed only on a single learned representation, the representation itself is treated as fixed. As a result, uncertainty estimates can become overconfident when the learned representation fails to capture variability in the data, particularly outside the training distribution (Watson et al., 2021). Extensions such as functional regularization have been proposed to mitigate this limitation, but they introduce additional computational overhead without fundamentally addressing the reliance on a single representation (Watson et al., 2021).

Deep ensembles (DE) provide a simple and scalable approach to uncertainty quantification by training multiple neural networks independently and uniformly averaging their predictive distributions (Lakshminarayanan et al., 2017). These methods typically yield strong empirical performance while keeping computational costs moderate. However, they lack an explicit probabilistic interpretation, as uncertainty is quantified implicitly through model diversity rather than through a well-defined probability measure over functions. Moreover, uniform averaging is non-diagnostic: it does not provide probabilistic evidence for which ensemble members are supported by the data or how individual members contribute to predictions. As a result, variation in member quality can degrade predictive accuracy and uncertainty calibration.

Taken together, these approaches illustrate a recurring pattern: methods that provide principled uncertainty estimation often incur substantial computational cost or restrict model flexibility, while scalable approaches typically rely on heuristic aggregation rules that offer limited interpretability. Designing models that simultaneously achieve computational efficiency, strong predictive performance, and interpretable uncertainty estimation, therefore, remains an open challenge.

1.1 Main contributions

In this work, we propose *Bayesian Deep Kernel Networks* (BDKN), a regression framework that combines independently trained neural networks with closed-form Bayesian aggregation. The method trains H neural networks on a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, producing multiple learned feature representations that serve as basis functions. A Bayesian linear regression layer is then applied to these ensemble outputs, yielding an analytic posterior over aggregation weights and a closed-form posterior predictive distribution (Figure 1). This design enables principled uncertainty estimation while retaining the computational efficiency and predictive performance of DE.

In the proposed framework, we leverage DE's scalability while replacing uniform averaging with Bayesian aggregation. By performing exact inference at the aggregation stage, the method produces interpretable posterior weights and calibrated uncertainty estimates without relying on approximate inference. Taken together, this design addresses the competing requirements **(i)–(iii)**. We summarize the resulting contributions below.

Scalable Bayesian aggregation with learned neural predictors and finite-rank GP interpretation. We aggregate independently trained neural networks using closed-form Bayesian linear regression,

yielding an analytic posterior predictive distribution with computational cost comparable to DE. The resulting model can be interpreted as a finite-rank Gaussian Process whose basis functions are learned directly through ensemble training, enabling principled Bayesian aggregation without kernel design or cubic-cost GP inference.

Separation of representation learning and probabilistic inference. Our framework maintains the standard DE training pipeline and performs Bayesian inference only at the aggregation stage. Unlike Bayesian last-layer (BLL) methods that place uncertainty on a single latent representation, our method defines uncertainty directly over ensemble-member contributions. This separation maintains implementation simplicity while providing interpretable probabilistic weights for individual predictors.

Posterior-driven aggregation robust to heterogeneous ensemble member quality. Instead of uniform averaging, a posterior distribution over the aggregation weights is inferred from the data. Ensemble members weakly supported by the data are automatically downweighted, improving robustness when model quality varies due to optimization variability or training conditions. This robustness is achieved without manual model selection or additional tuning heuristics.

To evaluate the practical implications of these design choices and their impact on computational efficiency, predictive performance, and interpretability, we conduct experiments on standard UCI regression benchmarks across a range of architectural and optimization settings.

2 Bayesian deep kernel networks

In this section, we introduce our framework while highlighting each improvement in comparison to existing methods. We start by introducing BLL and proceed to build upon it. We then discuss its connection to GP to provide further insight into the proposed framework.

2.1 Learning the feature space representation by ensemble members

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denote the dataset and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^\top$ the input matrix, where $\mathbf{x}_i \in \mathbb{R}^P$ and $y_i \in \mathbb{R}$ are input vectors and scalar responses. We consider scalar responses throughout this work for simplicity and without loss of generality. Denote by $\boldsymbol{\psi}(\cdot;\boldsymbol{\theta}) : \mathbb{R}^P \to \mathbb{R}^d$ the feature map of a single neural network with parameter vector $\boldsymbol{\theta}$. In BLLs the latent function f is modeled as

$$y_i = f(\mathbf{x}_i;\boldsymbol{\theta}) + \varepsilon_i, \quad f(\mathbf{x}_i;\boldsymbol{\theta}) = \boldsymbol{\psi}(\mathbf{x}_i;\boldsymbol{\theta})^\top \boldsymbol{\beta}, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2) \tag{1}$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\mu}_\beta, \boldsymbol{\Sigma}_\beta)$ denotes the feature weights.

Stacking the responses into a vector $\mathbf{y} \in \mathbb{R}^N$ yields the compact representation $\mathbf{y} = \boldsymbol{\Psi}(\mathbf{X};\boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where $\boldsymbol{\Psi}(\mathbf{X};\boldsymbol{\theta}) = [\boldsymbol{\psi}(\mathbf{x}_1;\boldsymbol{\theta}), \dots, \boldsymbol{\psi}(\mathbf{x}_N;\boldsymbol{\theta})]^\top \in \mathbb{R}^{N\times d}$. Training estimates $\boldsymbol{\theta}$, σ_ε^2, and $\boldsymbol{\Sigma}_\beta$ by maximizing an exact or approximate marginal likelihood, while the posterior $p(\boldsymbol{\beta} \mid \mathcal{D}, \boldsymbol{\theta})$ is available in closed form (Watson et al., 2021). The feature map $\boldsymbol{\psi}(\mathbf{x};\boldsymbol{\theta})$ is learned once and then treated as fixed during Bayesian inference. This implies that the variability employed to capture uncertainty stems only from the posterior over the weights $\boldsymbol{\beta}$, with the representation itself remaining deterministic. Therefore, uncertainty is restricted only to linear combinations within a single learned feature space.

To reduce reliance on a single learned representation while retaining the tractability of BLL inference, we combine ideas from DE-style representation learning with a BLL, aiming to improve predictive performance and uncertainty calibration, including in out-of-distribution regions. Unlike standard BLL, BDKN constructs its representation from a collection of H feature maps $\{\phi_h(\cdot;\boldsymbol{\theta}_h)\}_{h=1}^H$. Let $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_h\}_{h=1}^H$ and define $\boldsymbol{\phi}(\mathbf{x};\boldsymbol{\Theta}) = [\phi_1(\mathbf{x};\boldsymbol{\theta}_1), \dots, \phi_H(\mathbf{x};\boldsymbol{\theta}_H)]^\top \in \mathbb{R}^H$. The BDKN design matrix and model are

$$\begin{aligned} \boldsymbol{\Phi}(\mathbf{X};\boldsymbol{\Theta}) &= [\boldsymbol{\phi}(\mathbf{x}_1;\boldsymbol{\Theta}), \dots, \boldsymbol{\phi}(\mathbf{x}_N;\boldsymbol{\Theta})]^\top \in \mathbb{R}^{N\times H}, \\ \mathbf{y} &= \boldsymbol{\Phi}(\mathbf{X};\boldsymbol{\Theta})\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\beta} \in \mathbb{R}^H. \end{aligned} \tag{2}$$

Each of the H feature maps is implemented by an independently trained neural network. Training maximizes the Gaussian likelihood $y_i \sim \mathcal{N}(\phi_h(\mathbf{x}_i;\boldsymbol{\theta}_h), \sigma_h^2(\mathbf{x}_i;\boldsymbol{\theta}_h))$, where both $\phi_h(\cdot)$ and $\sigma_h^2(\cdot)$ are network outputs. Note that $\sigma_h^2(\cdot)$ is used only during training to induce a heteroscedastic objective with the goal of preventing all networks from fitting the data with the same smoothness profile. It

reweights the contribution of each training point to the loss: points with larger predicted variance incur a smaller penalty, while those with smaller variance are fitted more tightly. This sample-dependent reweighting leads to heterogeneous feature maps across independently trained ensemble members. The independent training yields the following benefits for our model. First, feature maps can be trained fully in parallel, allowing the ensemble size to scale without increasing training time beyond that of a single neural network. Second, each feature map represents a complete predictive model of the data, rather than a latent basis. In this way, the collection of independently learned feature maps captures multiple distinct functional representations of the data, which is particularly relevant when data are scarce compared to model size.

2.2 Bayesian inference in the induced function space

Once $\mathbf{\Phi}$ has been obtained, the posterior distribution over $\boldsymbol{\beta} \in \mathbb{R}^H$ is inferred via Bayesian linear regression in the function space (2).

We place a conjugate prior over the aggregation weights $\boldsymbol{\beta}$ and the observation-noise variance σ_ε^2. We adopt Jeffreys' prior for the noise variance to improve generalizability and uncertainty quantification,

$$p(\sigma_\varepsilon^2) \propto \frac{1}{\sigma_\varepsilon^2}, \tag{3}$$

which is invariant under reparameterization and introduces no additional scale hyperparameters (See Appendix A.1 for derivation). Given σ_ε^2, the aggregation weights are assigned a Gaussian prior,

$$\boldsymbol{\beta} \mid \sigma_\varepsilon^2 \sim \mathcal{N}\big(\boldsymbol{\mu}_0,\ \sigma_\varepsilon^2 \mathbf{\Lambda}_0^{-1}\big), \tag{4}$$

where $\mathbf{\Lambda}_0 \in \mathbb{R}^{H \times H}$ is positive definite.

This prior specification can be viewed as a special case of the Normal-Inverse-Gamma prior, where $\sigma_\varepsilon^2 \sim \text{Inv-Gamma}(a_0, b_0)$, obtained in the limit $a_0 \to 0$ and $b_0 \to 0$. Jeffreys' prior is a common choice when no prior information on variance is provided (Jeffreys, 1946; Gelman et al., 1995), while maintaining analytical tractability and eliminating the need to estimate additional hyperparameters for the variance.

Posterior updates: Given the design matrix $\mathbf{\Phi}$ and targets $\mathbf{y}$, conjugacy is maintained and the posterior factorizes as $p(\boldsymbol{\beta}, \sigma_\varepsilon^2 \mid \mathbf{y}, \mathbf{\Phi}) = p(\boldsymbol{\beta} \mid \sigma_\varepsilon^2, \mathbf{y}, \mathbf{\Phi})\ p(\sigma_\varepsilon^2 \mid \mathbf{y}, \mathbf{\Phi})$. The conditional posterior over aggregation weights is Gaussian, $\boldsymbol{\beta} \mid \sigma_\varepsilon^2, \mathbf{y}, \mathbf{\Phi} \sim \mathcal{N}\big(\boldsymbol{\mu}_n,\ \sigma_\varepsilon^2 \mathbf{\Lambda}_n^{-1}\big)$, with $\boldsymbol{\mu}_n = \mathbf{\Lambda}_n^{-1}\big(\mathbf{\Lambda}_0 \boldsymbol{\mu}_0 + \mathbf{\Phi}^\top \mathbf{y}\big)$ and $\mathbf{\Lambda}_n = \mathbf{\Lambda}_0 + \mathbf{\Phi}^\top \mathbf{\Phi}$. Under the Jeffreys prior on σ_ε^2, the marginal posterior over σ_ε^2 is $\sigma_\varepsilon^2 \mid \mathbf{y}, \mathbf{\Phi} \sim \text{Inv-Gamma}(a_n, b_n)$, where $a_n = \frac{N}{2}$ and $b_n = \frac{1}{2}\big(\mathbf{y}^\top \mathbf{y} + \boldsymbol{\mu}_0^\top \mathbf{\Lambda}_0 \boldsymbol{\mu}_0 - \boldsymbol{\mu}_n^\top \mathbf{\Lambda}_n \boldsymbol{\mu}_n\big)$. See Appendix A.3 for derivations.

Posterior predictive: For a new input with feature vector $\boldsymbol{\phi}_\star = \boldsymbol{\phi}(\mathbf{x}_\star)$, the predictive distribution integrates out both $\boldsymbol{\beta}$ and σ_ε^2, yielding a Student-t distribution,

$$p(y_\star \mid \boldsymbol{\phi}_\star, \mathbf{y}, \mathbf{\Phi}) = \text{St}\Big(y_\star;\ \nu = 2a_n,\ m = \boldsymbol{\phi}_\star^\top \boldsymbol{\mu}_n, s^2 = \frac{b_n}{a_n}\big(1 + \boldsymbol{\phi}_\star^\top \mathbf{\Lambda}_n^{-1} \boldsymbol{\phi}_\star\big)\Big). \tag{5}$$

Here, s^2 denotes the Student-t scale parameter.

In (5), $\boldsymbol{\mu}_0$ and $\mathbf{\Lambda}_0$ denote hyperparameters that must be specified. In the Bayesian linear regression and BLL literature, these are commonly set to $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\mathbf{\Lambda}_0 = \sigma_\beta^{-2} \mathbf{I}_H$, where σ_β^2 is estimated via empirical Bayes, which we also adopt (Watson et al., 2021). Specifically, σ_β^2 is obtained by maximizing the closed-form marginal likelihood. See Appendix A.2 and Appendix A.3 for details and derivations. Algorithm 1 in Appendix A.3 outlines the main steps of BDKN.

2.3 Connection to finite-rank GP

BDKN can also be interpreted as a finite-rank GP, conditional on the learned feature maps $\boldsymbol{\phi}$ and σ_ε^2.

Proposition 2.1 (Finite-rank GP representation)**.** *For fixed learned feature maps $\boldsymbol{\phi}(\cdot)$, any prior mean $\boldsymbol{\mu}_0$, and any positive-definite prior precision $\mathbf{\Lambda}_0$, consider the prior $\boldsymbol{\beta} \mid \sigma_\varepsilon^2 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma_\varepsilon^2 \mathbf{\Lambda}_0^{-1})$. Conditional on any fixed $\sigma_\varepsilon^2 > 0$, the function $f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\beta}$ is a finite-rank Gaussian process with mean function $m(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\mu}_0$ and covariance kernel $k(\mathbf{x}, \mathbf{x}' \mid \sigma_\varepsilon^2) = \sigma_\varepsilon^2 \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{\Lambda}_0^{-1} \boldsymbol{\phi}(\mathbf{x}')$. Its sample paths lie in $\mathcal{H}_H = \text{span}\{\phi_1, \ldots, \phi_H\}$, and its kernel matrices have rank at most H.*

For our isotropic prior, $\mathbf{\Lambda}_0 = \sigma_\beta^{-2}\mathbf{I}_H$, the kernel becomes $k(\mathbf{x}, \mathbf{x}' \mid \sigma_\varepsilon^2) = \sigma_\varepsilon^2\sigma_\beta^2 \sum_{h=1}^{H} \phi_h(\mathbf{x})\phi_h(\mathbf{x}')$. The proof of Proposition 2.1 is provided in Appendix A.4. The GP interpretation clarifies the roles of ensemble training and Bayesian aggregation from a function-space perspective. Ensemble training determines the finite-dimensional function space $\mathcal{H}_H$, whose dimension is at most H. Conditional on σ_ε^2, the Gaussian prior on the aggregation weights defines a GP prior over this space, and Bayesian inference using the training data yields the GP posterior. Marginalizing σ_ε^2 from the conditional GP posterior of f gives a Student-t posterior process (Shah et al., 2014) with sample paths in $\mathcal{H}_H$, consistent with the closed-form Student-t posterior predictive distribution presented in (5).

2.4 Computational complexity

Computationally, BDKN has nearly the same practical cost as a standard DE. A DE trains H independent neural networks and aggregates their predictions by uniform averaging, so its dominant cost is the cost of training the H ensemble members. BDKN uses the same ensemble training stage, but replaces uniform averaging with a Bayesian aggregation layer over the ensemble-member outputs. Given the ensemble-induced design matrix $\mathbf{\Phi} \in \mathbb{R}^{N\times H}$, this aggregation requires forming an $H \times H$ matrix and solving a small H-dimensional Bayesian linear regression problem, giving an additional cost of $O(NH^2 + H^3)$ time and $O(H^2)$ memory. The $O(NH^2)$ term comes from forming $\mathbf{\Phi}^\top\mathbf{\Phi}$, which sums pairwise interactions between ensemble members over all N training points, while the $O(H^3)$ term comes from solving the resulting $H \times H$ linear system. Since H is small and fixed in our proposed method, this additional cost is negligible relative to neural network training and remains linear in N.

3 Experiments

We evaluate the performance of the proposed model, namely BDKN, on 11 standard UCI regression benchmarks (Kelly et al.): Boston, Concrete, Energy, Kin8nm, Naval, Power, Protein, Sarcos, Song, Wine, and Yacht. We compare it against a diverse set of baseline methods, including Gaussian Processes (GP) (Williams & Rasmussen, 2006), Deep Ensemble (DE) (Lakshminarayanan et al., 2017), Deep Kernel Learning (DKL) (Wilson et al., 2016), Bayesian Last-Layer (BLL), Latent Derivative Bayesian Last-Layer (LD-BLL), and Mean-Field Variational Inference Bayesian Neural Networks (MFVI-BNN) (Watson et al., 2021), and Variational Bayesian Last-Layer (VBLL) (Harrison et al., 2024). The UCI datasets are widely used benchmarks (Lakshminarayanan et al., 2017; Watson et al., 2021; Harrison et al., 2024) for probabilistic nonlinear regression, covering a range of regression problems with varying input dimensionalities and sample sizes. Detailed dataset characteristics are provided in Appendix A.5.

We report the root mean squared error (RMSE) to evaluate predictive accuracy and the negative log-likelihood (NLL) to assess uncertainty calibration. RMSE is defined as $\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$, where $\hat{y}_i = \mathbb{E}[y_i \mid \mathbf{x}_i]$ denotes the predictive mean. NLL is computed as $\mathrm{NLL} = -\frac{1}{N}\sum_{i=1}^{N}\log p(y_i \mid \mathbf{x}_i)$, where $p(y_i \mid \mathbf{x}_i)$ denotes the predictive distribution of the model.

The main experimental results follow the lightweight configuration of Lakshminarayanan et al. (2017), including the data splits, training configuration, and ensemble size described below, which we refer to as Setting 1. We use 20 repeated random training/test splits. For each split, 90% of the data are used for training, and the remaining 10% for testing. We report the mean and standard deviation of the evaluation metrics over the 20 splits; for the Protein dataset, which is substantially larger, 5 repeated splits are used, and for the Song dataset, a single split is used. This configuration uses an aggressive learning rate of 0.1 and a short training schedule of 40 epochs for all models. In Setting 1, all neural-network-based approaches use two fully connected hidden layers with 50 neurons each and are trained with a batch size of 32. Our proposed model uses $H = 5$ feature maps (ensemble members). Additional details for this experimental setting and each method, including activation functions, are provided in Appendix A.6.

Results for this setting are reported in Table 1. BDKN achieves competitive performance across datasets in both RMSE and NLL, and consistently outperforms DE. While DE exhibits large variance and unstable behavior on several datasets (e.g., Power, Sarcos), BDKN remains stable, demonstrating robustness to variability in optimization quality among ensemble members.

Table 1: **Experimental setting 1 (high learning rate): performance on standard regression benchmarks. (a)** RMSE and **(b)** NLL reported as mean ± standard deviation across splits. Lower values are better for both metrics. **Bold** indicates the best performance, and underlined indicates the second-best performance. GP could not be evaluated on the Protein, Sarcos, and Song datasets, and LD-BLL on the Song dataset, due to their memory requirements, which exceeded our GPU memory.

(a) RMSE (mean ± std).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
Boston	3.03 ± 0.78	3.60 ± 1.76	7.80 ± 3.90	**2.83 ± 0.76**	3.45 ± 0.81	4.04 ± 1.67	5.45 ± 0.99	3.68 ± 1.05
Concrete	5.90 ± 0.48	7.18 ± 3.46	13.71 ± 6.76	**5.64 ± 0.66**	7.18 ± 0.67	8.84 ± 3.20	12.97 ± 2.41	6.62 ± 1.19
Energy	1.38 ± 0.33	2.54 ± 1.11	6.67 ± 6.10	**0.65 ± 0.06**	2.65 ± 0.42	5.77 ± 4.82	5.64 ± 1.78	1.64 ± 0.37
Kin8nm	0.15 ± 0.04	0.22 ± 0.04	0.72 ± 1.01	**0.07 ± 0.00**	0.10 ± 0.02	0.10 ± 0.00	0.23 ± 0.03	0.19 ± 0.03
Naval	0.01 ± 0.00	3.17 ± 7.88	0.03 ± 0.06	**0.00 ± 0.00**	0.01 ± 0.00	**0.00 ± 0.01**	0.01 ± 0.00	0.33 ± 0.33
Power	4.79 ± 1.02	3424.99 ± 6918.44	14.93 ± 10.67	**3.95 ± 0.16**	4.44 ± 0.19	6.33 ± 3.26	9.46 ± 1.86	8.38 ± 3.16
Protein	4.94 ± 0.09	5.31 ± 0.25	12.85 ± 7.03	N/A	5.03 ± 0.10	**4.42 ± 0.16**	5.58 ± 0.16	4.94 ± 0.09
Sarcos	3.94 ± 0.18	53.62 ± 157.00	12.11 ± 1.36	N/A	5.61 ± 1.06	**3.68 ± 0.06**	15.44 ± 3.32	5.47 ± 1.24
Song	10.85 ± 0.00	10.86 ± 0.00	N/A	N/A	8.98 ± 0.00	**8.91 ± 0.00**	11.60 ± 0.00	10.85 ± 0.00
Wine	0.64 ± 0.04	1.97 ± 3.03	0.78 ± 0.09	**0.56 ± 0.04**	0.65 ± 0.04	0.80 ± 0.05	0.73 ± 0.06	0.71 ± 0.04
Yacht	**0.82 ± 0.38**	1.68 ± 1.06	4.16 ± 4.20	1.88 ± 0.43	1.22 ± 0.64	4.72 ± 1.06	9.42 ± 2.99	1.70 ± 0.57

(b) NLL (mean ± std).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
Boston	2.61 ± 0.42	3.03 ± 0.79	4.73 ± 0.31	**2.42 ± 0.28**	4.45 ± 1.60	3.32 ± 0.14	3.24 ± 0.16	2.83 ± 0.49
Concrete	**3.22 ± 0.10**	3.67 ± 0.67	4.81 ± 0.25	3.53 ± 0.04	13.16 ± 2.07	3.94 ± 0.15	4.03 ± 0.20	3.37 ± 0.28
Energy	1.73 ± 0.24	2.79 ± 0.60	3.39 ± 1.31	**1.40 ± 0.02**	3.42 ± 0.66	3.44 ± 0.23	3.27 ± 0.20	1.96 ± 0.30
Kin8nm	-0.48 ± 0.22	-0.02 ± 0.30	1.29 ± 1.12	**-0.68 ± 0.00**	393.44 ± 953.78	-0.34 ± 0.00	0.01 ± 0.15	-0.21 ± 0.24
Naval	-2.80 ± 0.01	-0.07 ± 2.83	-2.05 ± 0.88	**-5.11 ± 0.01**	-1.14 ± 0.00	-3.20 ± 0.22	-3.03 ± 0.18	-0.51 ± 1.50
Power	2.97 ± 0.17	6.89 ± 2.06	5.46 ± 2.89	**2.83 ± 0.03**	6.91 ± 0.62	3.85 ± 0.11	3.88 ± 0.22	3.97 ± 1.57
Protein	3.00 ± 0.02	3.12 ± 0.11	4.21 ± 0.08	N/A	6.11 ± 0.15	**2.99 ± 0.02**	3.17 ± 0.04	3.03 ± 0.03
Sarcos	**2.80 ± 0.04**	4.02 ± 1.52	5.23 ± 0.44	N/A	9.71 ± 2.74	3.95 ± 0.00	4.16 ± 0.18	3.11 ± 0.23
Song	**3.49 ± 0.00**	3.81 ± 0.00	N/A	N/A	3.61 ± 0.00	3.65 ± 0.00	4.05 ± 0.00	3.81 ± 0.00
Wine	0.98 ± 0.06	1.59 ± 1.08	2.07 ± 0.23	**0.86 ± 0.05**	0.98 ± 0.07	1.19 ± 0.06	1.12 ± 0.10	1.08 ± 0.06
Yacht	**1.23 ± 0.55**	2.10 ± 0.73	2.79 ± 0.47	2.07 ± 0.08	1.74 ± 0.71	3.77 ± 0.09	3.75 ± 0.25	1.89 ± 0.45

The large DE RMSEs on Power and Sarcos arise from individual networks with extreme prediction errors that distort the equally weighted ensemble mean. To understand how such poor fits can persist, consider the Gaussian NLL objective used to train each member, which predicts both a mean $\phi_h(\mathbf{x}_i)$ and a variance $\sigma_h^2(\mathbf{x}_i)$. The loss divides the squared mean prediction error by the predicted variance, giving $(y_i - \phi_h(\mathbf{x}_i))^2/(2\sigma_h^2(\mathbf{x}_i))$. A network can therefore reduce this error penalty by increasing its predicted variance without improving the mean. The loss also penalizes larger variances through $\frac{1}{2}\log\sigma_h^2(\mathbf{x}_i)$, but this penalty grows only logarithmically, so increasing the variance can still lower the total loss when the mean prediction error is large. For the same mean prediction error, a larger variance also produces a smaller gradient for updating the mean, which can allow poor fits to persist during training (Seitzer et al., 2022). BDKN and DE use exactly the same trained ensemble, so their comparison highlights the effect of aggregation. DE assigns equal weights to all members, whereas BDKN infers posterior aggregation weights that describe each network's contribution and its uncertainty. Figure 2 compares individual network predictions with those of DE and BDKN and shows the corresponding posterior aggregation weights for one split from each of the Power and Sarcos datasets. On Power, Networks 2, 4, and 5 receive aggregation weights near zero, while Networks 1 and 3 receive the largest weights. On Sarcos, Network 2 receives an aggregation weight near zero, while Networks 3, 4, and 5 receive the largest weights. These individual splits illustrate how BDKN reduces the influence of poorly fitted members and achieves lower prediction errors than DE, consistent with the lower mean RMSE across all splits on Power and Sarcos reported in Table 1. Appendix A.7 provides further analysis of DE's large prediction errors on Power and Sarcos and BDKN's posterior aggregation weights, along with training and test figures and performance tables for individual ensemble members, DE, and BDKN.

GP remains a strong baseline on several datasets. However, due to its cubic computational scaling and prohibitive memory requirements, we could not evaluate GP on large datasets such as Protein, Sarcos, and Song. LD-BLL also exceeded the available GPU memory on Song due to the storage required for its derivative-based regularization. Compared to other BLL-based methods (BLL, LD-

BLL) and variational baselines (MFVI-BNN, VBLL), BDKN generally provides a better balance between predictive accuracy and uncertainty calibration. BDKN has runtime comparable to DE and is substantially faster than MFVI-BNN and VBLL on the three largest datasets (Protein, Sarcos, and Song). Detailed runtime comparisons under Setting 1 are provided in Appendix A.8.

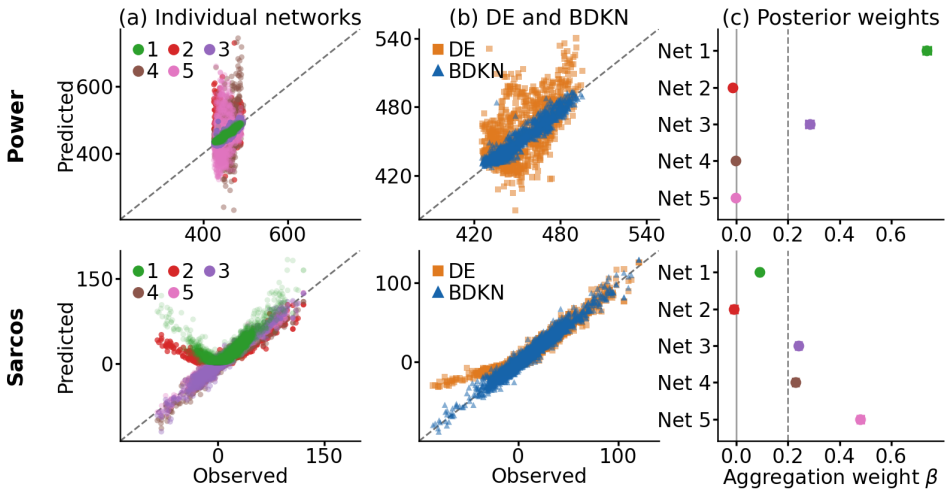


Figure 2: **Power (top) and Sarcos (bottom) - interpretation of BDKN's aggregation under Setting 1 (high learning rate). (a)** Individual networks' predictive means on the test set. **(b)** DE and BDKN predictive means on the test set. **(c)** Posterior means of the aggregation weights with marginal 95% credible intervals (the intervals are too narrow to be clearly visible at this scale; see the larger figures in Appendix A.7 for a clearer view). Dashed diagonals indicate ideal predictions in **(a–b)**; in **(c)**, solid and dashed vertical lines mark zero and DE's equal aggregation weight of 0.2, respectively.

Figure 3 shows test RMSE and NLL over training epochs on the Boston dataset under Setting 1. Most methods converge rapidly, while LD-BLL degrades after a few epochs and MFVI-BNN converges slowly despite the aggressive learning rate; GP shows the best performance. BDKN converges rapidly and maintains competitive RMSE and NLL throughout training.

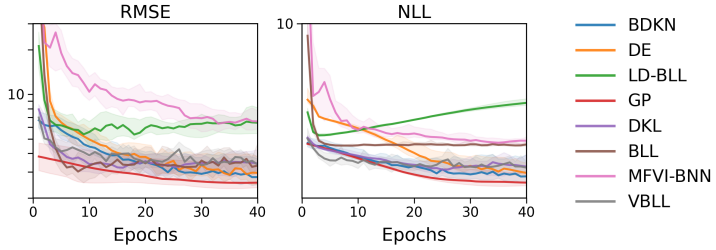


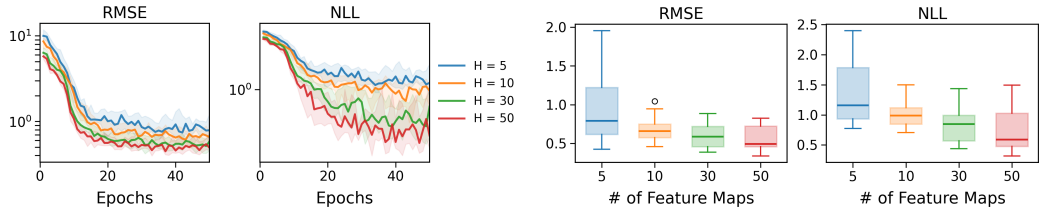
Figure 3: **Boston dataset - Test performance over training epochs under Setting 1 (high learning rate).** We show quartile plots of **test RMSE** (left) and **test NLL** (right) as functions of training epochs. Solid lines denote the median across 20 splits, while shaded regions indicate the interquartile range (25th–75th percentiles). Lower values are better for both RMSE and NLL.

To assess robustness to changes in optimization hyperparameters, we additionally evaluate BDKN and benchmark models using the more conservative, dataset-specific learning rates and training schedules of Watson et al. (2021). We refer to this additional experimental setting as Setting 2. Its experimental details and results are provided in Appendix A.9. Under this more conservative training configuration, BDKN remains competitive in both predictive accuracy and uncertainty estimation across the benchmark datasets.

We also evaluate BDKN's out-of-distribution predictive accuracy and uncertainty estimates in regions excluded from training. For this purpose, we conduct gap experiments that withhold selected input

ranges from training and use observations in those regions for testing (Foong et al., 2019; Watson et al., 2021). Appendix A.10 reports results under both aggressive (Setting 1) and conservative (Setting 2) training regimes, along with wall clock runtimes. The predictive performance metrics show trends similar to those observed on the standard regression benchmarks.

A key advantage of BDKN is that its feature maps can be trained fully in parallel as ensemble members. Although we use five ensemble members following Lakshminarayanan et al. (2017) for simplicity, the ensemble size can be increased without substantially increasing training time when sufficient computational resources are available. Figure 4 illustrates the effect of ensemble size on performance for the Yacht dataset under Setting 1 with 50 training epochs. As the number of ensemble members increases, both RMSE and NLL improve consistently, and larger ensembles exhibit faster convergence across splits. This behavior is evident throughout training (Figure 4a) and at the final epoch, where the median RMSE and NLL decrease monotonically with ensemble size (Figure 4b). These results indicate that increasing the ensemble size yields a richer and more expressive feature space, improving predictive accuracy and uncertainty calibration.



(a) (b)

Figure 4: **Yacht (Setting 1) - effect of BDKN ensemble size on the test data performance. (a)** Quartile plots of **test RMSE** (left) and **test NLL** (right) as functions of training epochs for ensembles with 5, 10, 30, and 50 feature maps. Solid lines denote the median across splits and shaded regions indicate the interquartile range (25th–75th percentiles). **(b)** Boxplots of **test RMSE** (left) and **test NLL** (right) across splits at the final training epoch. Boxes show the interquartile range with the median marked; whiskers denote the full range excluding outliers. Lower values are better for both RMSE and NLL.

We also evaluate larger ensembles ($H = 50$) across the standard regression benchmarks and report the results in Appendix A.11. These experiments show that increasing the ensemble size generally improves BDKN's performance, while the wall-clock runtimes remain comparable in scale to those of the five-network configuration.

4 Related Work

Gaussian process: GPs provide a nonparametric Bayesian framework for regression by placing a prior directly over functions (Williams & Rasmussen, 2006). However, their practical use is often limited by computational complexity and dependence on kernel choice (Quinonero-Candela & Rasmussen, 2005; Wilson & Adams, 2013), motivating a large body of approximation methods such as sparse and inducing-point approaches (Snelson & Ghahramani, 2005; Titsias, 2009). Beyond scalability, numerous extensions have been proposed to improve expressiveness, including neural-network-based kernels and deep GP variants (Cho & Saul, 2009; Calandra et al., 2016; Wilson et al., 2016; Bui et al., 2016). Related theoretical work has also established connections between infinitely wide neural networks and GPs, characterizing the function-space priors induced by neural architectures (Lee et al., 2017; Rudner et al., 2018; Khan et al., 2019).

Bayesian neural networks: BNNs place prior distributions over network parameters and perform posterior inference to capture predictive uncertainty (Neal, 2012). While fully Bayesian inference via Monte Carlo sampling methods (Andrieu et al., 2003; Hoffman et al., 2014; Chen et al., 2014) provides a principled treatment of uncertainty, such approaches are typically computationally expensive and difficult to scale to modern architectures (Welling & Teh, 2011). As a result, much of the recent work on BNNs has focused on scalable approximate inference techniques, most notably variational inference methods, including mean-field approximations, which trade posterior fidelity

for computational tractability (Shridhar et al., 2019; Coker et al., 2022; Sun et al., 2019; Cinquin & Bamler, 2025). Ensembles of BNNs trained using MFVI have also been explored to better approximate the Bayesian predictive distribution (Wilson et al., 2022).

Bayesian last layer networks: BLLs restrict Bayesian inference to a linear head on top of learned deterministic features, offering a practical trade-off between scalability and uncertainty estimation. Building on this formulation, several extensions refine training and uncertainty behavior. In particular, an efficient reformulation of BLL enables backpropagation-based marginal likelihood training and improves extrapolative uncertainty quantification (Fiedler & Lucia, 2023). To address overconfident predictions and improve uncertainty estimation outside the training distribution, LD-BLL networks introduce functional priors over network derivatives with respect to the inputs to encourage more expressive and calibrated uncertainty (Watson et al., 2021). More recently, VBLL has been proposed to further improve scalability and uncertainty estimation through a sampling-free, single-pass training objective (Harrison et al., 2024). Mixtures of BLL models fitted to the learned features of multiple independently trained neural networks have also been explored (Moberg et al., 2019).

Deep ensembles: DE combines independently trained neural networks, typically via uniform averaging, and is widely used in practice, including reinforcement learning (Osband et al., 2018; An et al., 2020; Wu & Li, 2020; Lee et al., 2021; Abe et al., 2022; Seligmann et al., 2023). Several studies have provided theoretical and analytical perspectives on standard DE and its uncertainty behavior (Fort et al., 2019; Pearce et al., 2020). Prior work has shown that promoting ensemble diversity can improve uncertainty estimation in DE (Reeve & Brown, 2018; Nam et al., 2021; Rame & Cord, 2021; D'Angelo & Fortuin, 2021; Trinh et al., 2024). For instance, ensemble diversity can be encouraged by augmenting the loss function with an auxiliary regularization term that maximizes ensemble predictive variance on inputs sampled uniformly from the input space, improving uncertainty behavior away from the training data (Jain et al., 2020). Diversity can also be induced structurally through architectural heterogeneity across ensemble members, such as by randomizing activation functions across networks (Zaidi et al., 2021; Stoyanova et al., 2023).

5 Discussion

We introduced BDKN, a scalable framework that bridges the gap between the empirical success of DE and the statistical rigor of GPs. By interpreting independently trained neural networks as induced basis functions, BDKN constructs a finite-rank deep kernel model that enables exact Bayesian inference without the cubic computational costs associated with traditional GPs. Since the ensemble members in BDKN are trained independently, the feature maps can be learned in parallel. This allows the number of feature maps to be scaled to increase model expressivity without a significant increase in training time, which we empirically show leads to improved performance. Our approach offers a clear advantage over standard weight-space BNNs and variational approximations: it yields closed-form posterior predictive distributions together with an interpretable posterior over ensemble members. BDKN also addresses a key limitation of DE, namely their non-diagnostic uniform averaging, by replacing it with a Bayesian aggregation layer that automatically downweights ensemble members weakly supported by the data. This aggregation mechanism improves robustness by reducing the impact of poorly optimized or low-quality ensemble members, which can otherwise degrade performance under aggressive training regimes. While baselines such as DE and LD-BLL show sensitivity to learning-rate selection and can become unstable in high learning-rate regimes, BDKN consistently maintains competitive predictive accuracy and well-calibrated uncertainty under the aggressive training configuration of Setting 1. Moreover, the gap experiments show that BDKN remains competitive in out-of-distribution regions excluded from training.

This study focuses on regression; extending the BDKN framework to classification tasks is a natural next step. A key open challenge is maintaining analytic tractability of the posterior predictive distribution under non-Gaussian likelihoods while retaining the robustness and interpretability observed in the regression setting. Further improvements in uncertainty calibration may be achieved by enhancing variability in the learned feature maps or by incorporating principles from latent-derivative modeling. Finally, extending BDKN to other domains, such as deep reinforcement learning and physics-informed machine learning, represents a promising direction for future work.

REFERENCES

Taiga Abe, Estefany Kelly Buchanan, Geoff Pleiss, Richard Zemel, and John P Cunningham. Deep ensembles work, but are they necessary? *Advances in Neural Information Processing Systems*, 35: 33646–33660, 2022.

Ning An, Huitong Ding, Jiaoyun Yang, Rhoda Au, and Ting FA Ang. Deep ensemble learning for alzheimer's disease classification. *Journal of biomedical informatics*, 105:103411, 2020.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE signal processing magazine*, 34(6):26–38, 2017.

Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep gaussian processes for regression using approximate expectation propagation. In *International conference on machine learning*, pp. 1472–1481. PMLR, 2016.

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. In *2016 International joint conference on neural networks (IJCNN)*, pp. 3338–3345. IEEE, 2016.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pp. 1683–1691. PMLR, 2014.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

Tristan Cinquin and Robert Bamler. Well-defined function-space variational inference in bayesian neural networks via regularized kl-divergence. In Silvia Chiappa and Sara Magliacane (eds.), *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pp. 752–776. PMLR, 21–25 Jul 2025.

Beau Coker, Wessel P Bruinsma, David R Burt, Weiwei Pan, and Finale Doshi-Velez. Wide mean-field bayesian neural networks ignore the data. In *International Conference on Artificial Intelligence and Statistics*, pp. 5276–5333. PMLR, 2022.

Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *Advances in Neural Information Processing Systems*, 34:3451–3465, 2021.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Felix Fiedler and Sergio Lucia. Improved uncertainty quantification for neural networks with bayesian last layer. *IEEE Access*, 11:123149–123160, 2023.

Angelos Filos, Sebastian Farquhar, Aidan N Gomez, Tim GJ Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud De Kroon, and Yarin Gal. A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks. *arXiv preprint arXiv:1912.10481*, 2019.

Andrew Foong, David Burt, Yingzhen Li, and Richard Turner. On the expressiveness of approximate inference in bayesian neural networks. *Advances in Neural Information Processing Systems*, 33: 15897–15908, 2020.

Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 'in-between'uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

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, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

James Harrison, John Willes, and Jasper Snoek. Variational bayesian last layers. *arXiv preprint arXiv:2404.11599*, 2024.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Matthew D Hoffman, Andrew Gelman, et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

Siddhartha Jain, Ge Liu, Jonas Mueller, and David Gifford. Maximizing overall diversity for improved uncertainty estimates in deep ensembles. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 4264–4271, 2020.

Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1007):453–461, 1946.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI Machine Learning Repository. https://archive.ics.uci.edu. University of California, Irvine.

Mohammad Emtiyaz Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate inference turns deep networks into gaussian processes. *Advances in neural information processing systems*, 32, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Miguel Lázaro-Gredilla and Aníbal R Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *IEEE transactions on neural networks*, 21(8):1345–1351, 2010.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International conference on machine learning*, pp. 6131–6141. PMLR, 2021.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

John Moberg, Lennart Svensson, Juliano Pinto, and Henk Wymeersch. Bayesian linear regression on deep representations. *arXiv preprint arXiv:1912.06760*, 2019.

Giung Nam, Jongmin Yoon, Yoonho Lee, and Juho Lee. Diversity matters when learning from ensembles. *Advances in neural information processing systems*, 34:8367–8377, 2021.

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

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, pp. 234–244. PMLR, 2020.

Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of machine learning research*, 6(Dec):1939–1959, 2005.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Alexandre Rame and Matthieu Cord. Dice: Diversity in deep ensembles via conditional redundancy adversarial estimation. *arXiv preprint arXiv:2101.05544*, 2021.

Henry WJ Reeve and Gavin Brown. Diversity and degrees of freedom in regression ensembles. *Neurocomputing*, 298:55–68, 2018.

Tim GJ Rudner, Vincent Fortuin, Yee Whye Teh, and Yarin Gal. On the connection between neural processes and gaussian processes with deep kernels. In *Workshop on Bayesian Deep Learning, NeurIPS*, pp. 14, 2018.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. *arXiv preprint arXiv:2203.09168*, 2022.

Florian Seligmann, Philipp Becker, Michael Volpp, and Gerhard Neumann. Beyond deep ensembles: A large-scale evaluation of bayesian deep learning under distribution shift. *Advances in Neural Information Processing Systems*, 36:29372–29405, 2023.

Amar Shah, Andrew Wilson, and Zoubin Ghahramani. Student-t processes as alternatives to gaussian processes. In *Artificial intelligence and statistics*, pp. 877–885. PMLR, 2014.

Kumar Shridhar, Felix Laumann, and Marcus Liwicki. A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv preprint arXiv:1901.02731*, 2019.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.

Yana Stoyanova, Soroush Ghandi, and Maryam Tavakol. Toward robust uncertainty estimation with random activation functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 15152–15160, 2023.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pp. 567–574. PMLR, 2009.

Trung Trinh, Markus Heinonen, Luigi Acerbi, and Samuel Kaski. Input-gradient space particle inference for neural network ensembles. In *International Conference on Learning Representations*, volume 2024, pp. 22188–22206, 2024.

Joe Watson, Jihao Andreas Lin, Pascal Klink, Joni Pajarinen, and Jan Peters. Latent derivative bayesian last layer networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 1198–1206. PMLR, 2021.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pp. 1067–1075. PMLR, 2013.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

Andrew Gordon Wilson, Pavel Izmailov, Matthew D Hoffman, Yarin Gal, Yingzhen Li, Melanie F Pradier, Sharad Vikram, Andrew Foong, Sanae Lotfi, and Sebastian Farquhar. Evaluating approximate inference in bayesian deep learning. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 113–124. PMLR, 2022.

Junta Wu and Huiyun Li. Deep ensemble reinforcement learning with multiple deep deterministic policy gradient algorithm. *Mathematical Problems in Engineering*, 2020(1):4275623, 2020.

Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris C Holmes, Frank Hutter, and Yee Teh. Neural ensemble search for uncertainty estimation and dataset shift. *Advances in Neural Information Processing Systems*, 34:7898–7911, 2021.

A Appendix

A.1 Jeffreys prior for the noise variance

We derive the Jeffreys prior for the Gaussian noise variance σ_ε^2 in the Bayesian linear model used in Step 2. Consider the observation model

$$\mathbf{y} \mid \boldsymbol{\beta}, \sigma_\varepsilon^2 \sim \mathcal{N}\big(\boldsymbol{\Phi}\boldsymbol{\beta},\ \sigma_\varepsilon^2 \mathbf{I}_N\big),$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{N \times H}$ is the design induced by the ensemble matrix. The log likelihood of σ_ε^2 is

$$\ell(\sigma_\varepsilon^2) = -\frac{N}{2}\log \sigma_\varepsilon^2 - \frac{1}{2\sigma_\varepsilon^2}\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|^2 + C,$$

where C denotes a constant independent of σ_ε^2. Differentiating twice with respect to σ_ε^2 yields

$$\frac{\partial^2 \ell}{\partial (\sigma_\varepsilon^2)^2} = \frac{N}{2}\,\frac{1}{(\sigma_\varepsilon^2)^2} - \frac{\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|^2}{(\sigma_\varepsilon^2)^3}.$$

Taking the negative expectation under $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\Phi}\boldsymbol{\beta}, \sigma_\varepsilon^2 \mathbf{I}_N)$ gives the Fisher information

$$\mathcal{I}(\sigma_\varepsilon^2) = \frac{N}{2}\,\frac{1}{(\sigma_\varepsilon^2)^2}.$$

By definition, the Jeffreys prior is proportional to the square root of the Fisher information, yielding

$$p(\sigma_\varepsilon^2) \ \propto\ \sqrt{\mathcal{I}(\sigma_\varepsilon^2)} \ =\ \frac{1}{\sigma_\varepsilon^2}.$$

A.2 Isotropic prior over aggregation weights

In our experiments, we adopt an isotropic Gaussian prior with zero mean over the aggregation weights by setting

$$\boldsymbol{\Lambda}_0 = \sigma_\beta^{-2}\mathbf{I}_H,$$

which corresponds to $\boldsymbol{\beta} \mid \sigma_\varepsilon^2 \sim \mathcal{N}(\mathbf{0},\ \sigma_\varepsilon^2\sigma_\beta^2\mathbf{I}_H)$. Under this choice, all basis functions induced by the ensemble are a priori treated symmetrically and assigned the same variance, reflecting the absence of additional structural assumptions about the relative importance of individual ensemble members.

This isotropic prior offers several practical and conceptual advantages. First, it avoids introducing additional biases that would favor specific ensemble members, allowing uncertainty to be driven primarily by the diversity of the learned functions themselves. Second, the single scale parameter σ_β^2 controls the overall strength of the prior in the function space induced by the ensemble, enabling efficient empirical Bayes tuning via the marginal likelihood without increasing model complexity.

A.3 BDKN inference

This section discusses the posterior updates, marginal likelihood optimization, and inference algorithm used in BDKN.

Posterior updates

Given $\mathbf{y} \in \mathbb{R}^N$, $\boldsymbol{\Phi} \in \mathbb{R}^{N \times H}$, and $\boldsymbol{\beta} \in \mathbb{R}^H$, the likelihood, conditional Gaussian prior, and Jeffreys prior have kernels

$$\begin{aligned}
p(\mathbf{y} \mid \boldsymbol{\beta}, \sigma_\varepsilon^2, \boldsymbol{\Phi}) &\propto (\sigma_\varepsilon^2)^{-N/2} \exp\left[-\frac{(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta})^\top(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta})}{2\sigma_\varepsilon^2}\right], \\
p(\boldsymbol{\beta} \mid \sigma_\varepsilon^2) &\propto (\sigma_\varepsilon^2)^{-H/2} \exp\left[-\frac{(\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top\boldsymbol{\Lambda}_0(\boldsymbol{\beta} - \boldsymbol{\mu}_0)}{2\sigma_\varepsilon^2}\right], \\
p(\sigma_\varepsilon^2) &\propto (\sigma_\varepsilon^2)^{-1}.
\end{aligned}$$

Multiplying these kernels gives

$$p(\boldsymbol{\beta}, \sigma_\varepsilon^2 \mid \mathbf{y}, \boldsymbol{\Phi}) \propto (\sigma_\varepsilon^2)^{-(N+H)/2-1} \exp\left[-\frac{Q(\boldsymbol{\beta})}{2\sigma_\varepsilon^2}\right],$$
$$\begin{aligned} Q(\boldsymbol{\beta}) &= (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta})^\top (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}) + (\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Lambda}_0 (\boldsymbol{\beta} - \boldsymbol{\mu}_0) \\ &= \boldsymbol{\beta}^\top \boldsymbol{\Lambda}_n \boldsymbol{\beta} - 2\boldsymbol{\beta}^\top (\boldsymbol{\Phi}^\top \mathbf{y} + \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0) + \mathbf{y}^\top \mathbf{y} + \boldsymbol{\mu}_0^\top \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0, \end{aligned}$$

where

$$\boldsymbol{\mu}_n = \boldsymbol{\Lambda}_n^{-1}\big(\boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0 + \boldsymbol{\Phi}^\top \mathbf{y}\big), \tag{6}$$
$$\boldsymbol{\Lambda}_n = \boldsymbol{\Lambda}_0 + \boldsymbol{\Phi}^\top \boldsymbol{\Phi}. \tag{7}$$

Completing the square gives

$$\begin{aligned} Q(\boldsymbol{\beta}) &= (\boldsymbol{\beta} - \boldsymbol{\mu}_n)^\top \boldsymbol{\Lambda}_n (\boldsymbol{\beta} - \boldsymbol{\mu}_n) + c_n, \\ c_n &= \mathbf{y}^\top \mathbf{y} + \boldsymbol{\mu}_0^\top \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0 - \boldsymbol{\mu}_n^\top \boldsymbol{\Lambda}_n \boldsymbol{\mu}_n. \end{aligned}$$

Therefore, the conditional posterior over the aggregation weights is

$$\boldsymbol{\beta} \mid \sigma_\varepsilon^2, \mathbf{y}, \boldsymbol{\Phi} \sim \mathcal{N}\big(\boldsymbol{\mu}_n, \sigma_\varepsilon^2 \boldsymbol{\Lambda}_n^{-1}\big).$$

To obtain the marginal posterior of σ_ε^2, we integrate out $\boldsymbol{\beta}$. The Gaussian integral is

$$\begin{aligned} &\int_{\mathbb{R}^H} \exp\left[-\frac{(\boldsymbol{\beta} - \boldsymbol{\mu}_n)^\top \boldsymbol{\Lambda}_n (\boldsymbol{\beta} - \boldsymbol{\mu}_n)}{2\sigma_\varepsilon^2}\right] d\boldsymbol{\beta} \\ &\qquad = (2\pi\sigma_\varepsilon^2)^{H/2} |\boldsymbol{\Lambda}_n|^{-1/2}. \end{aligned}$$

Thus, the factor $(\sigma_\varepsilon^2)^{H/2}$ from the integral cancels the corresponding contribution from the prior over aggregation weights, yielding

$$p(\sigma_\varepsilon^2 \mid \mathbf{y}, \boldsymbol{\Phi}) \propto (\sigma_\varepsilon^2)^{-N/2-1} \exp\left[-\frac{c_n}{2\sigma_\varepsilon^2}\right].$$

Using the shape–scale convention $p(s) \propto s^{-(a+1)} \exp(-b/s)$, this is Inv-Gamma$(a_n, b_n)$, where

$$a_n = \frac{N}{2}, \tag{8}$$
$$b_n = \frac{1}{2}\big(\mathbf{y}^\top \mathbf{y} + \boldsymbol{\mu}_0^\top \boldsymbol{\Lambda}_0 \boldsymbol{\mu}_0 - \boldsymbol{\mu}_n^\top \boldsymbol{\Lambda}_n \boldsymbol{\mu}_n\big). \tag{9}$$

More generally, under $\sigma_\varepsilon^2 \sim$ Inv-Gamma(a_0, b_0), the updates are $a_n = a_0 + N/2$ and $b_n = b_0 + c_n/2$. Jeffreys' prior yields $a_n = N/2$ and $b_n = c_n/2$.

Posterior predictive

For a new feature vector $\boldsymbol{\phi}_\star$, let $y_\star = \boldsymbol{\phi}_\star^\top \boldsymbol{\beta} + \varepsilon_\star$, where $\varepsilon_\star \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. Conditionally on σ_ε^2, integrating the posterior of $\boldsymbol{\beta}$ gives

$$\begin{aligned} &p(y_\star \mid \boldsymbol{\phi}_\star, \sigma_\varepsilon^2, \mathbf{y}, \boldsymbol{\Phi}) \\ &\qquad = \int p(y_\star \mid \boldsymbol{\beta}, \sigma_\varepsilon^2, \boldsymbol{\phi}_\star) p(\boldsymbol{\beta} \mid \sigma_\varepsilon^2, \mathbf{y}, \boldsymbol{\Phi})\, d\boldsymbol{\beta}. \end{aligned}$$

By the laws of total expectation and total variance,

$$\begin{aligned} \mathbb{E}[y_\star \mid \boldsymbol{\phi}_\star, \sigma_\varepsilon^2, \mathbf{y}, \boldsymbol{\Phi}] &= \boldsymbol{\phi}_\star^\top \boldsymbol{\mu}_n, \\ \mathrm{Var}(y_\star \mid \boldsymbol{\phi}_\star, \sigma_\varepsilon^2, \mathbf{y}, \boldsymbol{\Phi}) &= \mathbb{E}[\mathrm{Var}(y_\star \mid \boldsymbol{\beta}, \sigma_\varepsilon^2)] + \mathrm{Var}(\mathbb{E}[y_\star \mid \boldsymbol{\beta}, \sigma_\varepsilon^2]) \\ &= \sigma_\varepsilon^2 + \sigma_\varepsilon^2 \boldsymbol{\phi}_\star^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_\star. \end{aligned}$$

Defining

$$m_\star = \boldsymbol{\phi}_\star^\top \boldsymbol{\mu}_n, \qquad c_\star = 1 + \boldsymbol{\phi}_\star^\top \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\phi}_\star,$$

we obtain

$$y_\star \mid \boldsymbol{\phi}_\star, \sigma_\varepsilon^2, \mathbf{y}, \boldsymbol{\Phi} \sim \mathcal{N}\big(m_\star, \sigma_\varepsilon^2 c_\star\big)\,.$$

We next integrate $\sigma_\varepsilon^2 \sim \text{Inv-Gamma}(a_n, b_n)$:

$$\begin{aligned} p(y_\star \mid \boldsymbol{\phi}_\star, \mathbf{y}, \boldsymbol{\Phi}) &= \int_0^\infty \mathcal{N}(y_\star; m_\star, \sigma_\varepsilon^2 c_\star)\text{Inv-Gamma}(\sigma_\varepsilon^2; a_n, b_n)\, d\sigma_\varepsilon^2 \\ &= \frac{\Gamma(a_n + \frac{1}{2})}{\Gamma(a_n)\sqrt{2\pi b_n c_\star}} \left[1 + \frac{(y_\star - m_\star)^2}{2b_n c_\star}\right]^{-(a_n+1/2)}. \end{aligned}$$

This is a Student-t distribution with $\nu = 2a_n$, location $m = m_\star$, and scale $s^2 = (b_n/a_n)c_\star$, recovering the posterior predictive distribution in (5).

MARGINAL LIKELIHOOD

The hyperparameter $\boldsymbol{\Lambda}_0$, specifically σ_β^2, is estimated by maximizing the marginal likelihood objective. Starting from the joint model and retaining the Gaussian normalization factors gives

$$p(\mathbf{y} \mid \boldsymbol{\Phi}) \propto \int_0^\infty \int_{\mathbb{R}^H} p(\mathbf{y} \mid \boldsymbol{\beta}, \sigma_\varepsilon^2, \boldsymbol{\Phi}) p(\boldsymbol{\beta} \mid \sigma_\varepsilon^2)(\sigma_\varepsilon^2)^{-1}\, d\boldsymbol{\beta}\, d\sigma_\varepsilon^2.$$

The product of the likelihood and conditional Gaussian prior involves the same quadratic expression $Q(\boldsymbol{\beta})$ as in the posterior derivation. Using its form obtained by completing the square, integration over $\boldsymbol{\beta}$ yields

$$p(\mathbf{y} \mid \boldsymbol{\Phi}) \propto (2\pi)^{-N/2} \frac{|\boldsymbol{\Lambda}_0|^{1/2}}{|\boldsymbol{\Lambda}_n|^{1/2}} \int_0^\infty (\sigma_\varepsilon^2)^{-a_n-1} \exp\left(-\frac{b_n}{\sigma_\varepsilon^2}\right) d\sigma_\varepsilon^2.$$

The remaining integral satisfies

$$\int_0^\infty s^{-a_n-1} \exp(-b_n/s)\, ds = \Gamma(a_n) b_n^{-a_n},$$

and hence

$$p(\mathbf{y} \mid \boldsymbol{\Phi}) \propto (2\pi)^{-N/2} \frac{|\boldsymbol{\Lambda}_0|^{1/2}}{|\boldsymbol{\Lambda}_n|^{1/2}} \Gamma(a_n) b_n^{-a_n}.$$

Up to an additive constant independent of σ_β^2, the empirical Bayes objective is

$$\mathcal{L}_{\text{EB}}(\sigma_\beta^2) = -a_n \log b_n + \frac{1}{2}\log|\boldsymbol{\Lambda}_0| - \frac{1}{2}\log|\boldsymbol{\Lambda}_n|, \tag{10}$$

The omitted constant contains $-\frac{N}{2}\log(2\pi) + \log\Gamma(a_n)$ and an arbitrary contribution from the improper Jeffreys prior kernel, which is independent of σ_β^2; thus, it does not affect empirical Bayes optimization of σ_β^2. Algorithm 1 outlines the main steps of BDKN.

Algorithm 1 Bayesian Deep Kernel Networks (BDKN)

1: **Inputs:** training set $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, ensemble size H, initial prior variance $\sigma_\beta^2 = 1$
2: **Outputs:** feature maps $\{\phi_h\}_{h=1}^H$ and posterior predictive distribution $p(y_\star \mid \mathbf{x}_\star, \mathcal{D}_{\text{tr}}, \{\phi_h\})$
3: **Step 1: Learn feature maps via independent training.**
4: Train H neural networks independently on $\mathcal{D}_{\text{tr}}$ to obtain $\{\phi_h(\cdot; \boldsymbol{\theta}_h)\}_{h=1}^H$.
5: **Step 2: Bayesian inference over learned features.**
6: Construct the design matrix $\boldsymbol{\Phi}$ using $\{\phi_h\}_{h=1}^H$.
7: Optimize σ_β^2 by maximizing the log marginal likelihood objective in (10).
8: Compute posterior parameters $(\boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n, a_n, b_n)$ using (6)-(8)
9: Obtain the posterior predictive distribution using (5).

A.4 Proof of Proposition 2.1

Proof. Fix the learned feature maps $\boldsymbol{\phi}(\cdot)$ and let $\boldsymbol{\Lambda}_0 \succ 0$. Recall that

$$f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\beta}, \qquad \boldsymbol{\beta} \mid \sigma_\varepsilon^2 \sim \mathcal{N}\big(\boldsymbol{\mu}_0, \sigma_\varepsilon^2 \boldsymbol{\Lambda}_0^{-1}\big).$$

For any finite collection of inputs, the corresponding function values are a linear transformation of the Gaussian vector $\boldsymbol{\beta}$ and are therefore jointly Gaussian conditional on $\sigma_\varepsilon^2 > 0$. Thus, f is a Gaussian process conditional on σ_ε^2.

The conditional mean function is

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}\big[f(\mathbf{x}) \mid \sigma_\varepsilon^2\big] \\ &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbb{E}\big[\boldsymbol{\beta} \mid \sigma_\varepsilon^2\big] \\ &= \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\mu}_0. \end{aligned}$$

The conditional covariance kernel is

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}' \mid \sigma_\varepsilon^2) &= \mathrm{Cov}\big(f(\mathbf{x}), f(\mathbf{x}') \mid \sigma_\varepsilon^2\big) \\ &= \mathbb{E}\Big[\big(f(\mathbf{x}) - m(\mathbf{x})\big)\big(f(\mathbf{x}') - m(\mathbf{x}')\big) \,\Big|\, \sigma_\varepsilon^2\Big] \\ &= \mathbb{E}\Big[\boldsymbol{\phi}(\mathbf{x})^\top (\boldsymbol{\beta} - \boldsymbol{\mu}_0)(\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top \boldsymbol{\phi}(\mathbf{x}') \,\Big|\, \sigma_\varepsilon^2\Big] \\ &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbb{E}\Big[(\boldsymbol{\beta} - \boldsymbol{\mu}_0)(\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top \,\Big|\, \sigma_\varepsilon^2\Big] \boldsymbol{\phi}(\mathbf{x}') \\ &= \boldsymbol{\phi}(\mathbf{x})^\top \mathrm{Cov}\big(\boldsymbol{\beta} \mid \sigma_\varepsilon^2\big) \boldsymbol{\phi}(\mathbf{x}') \\ &= \sigma_\varepsilon^2 \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Lambda}_0^{-1} \boldsymbol{\phi}(\mathbf{x}'). \end{aligned}$$

Writing $\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_H(\mathbf{x})]^\top$, the kernel can equivalently be expressed as

$$k(\mathbf{x}, \mathbf{x}' \mid \sigma_\varepsilon^2) = \sigma_\varepsilon^2 \sum_{h=1}^{H} \sum_{h'=1}^{H} (\boldsymbol{\Lambda}_0^{-1})_{hh'} \phi_h(\mathbf{x}) \phi_{h'}(\mathbf{x}').$$

Inner product and finite rank representation: Let $\boldsymbol{\Lambda}_0^{-1/2}$ denote the symmetric positive-definite square root of $\boldsymbol{\Lambda}_0^{-1}$ and define $\widetilde{\boldsymbol{\phi}}(\mathbf{x}) = \boldsymbol{\Lambda}_0^{-1/2} \boldsymbol{\phi}(\mathbf{x})$. The kernel then has the inner product representation

$$k(\mathbf{x}, \mathbf{x}' \mid \sigma_\varepsilon^2) = \sigma_\varepsilon^2 \widetilde{\boldsymbol{\phi}}(\mathbf{x})^\top \widetilde{\boldsymbol{\phi}}(\mathbf{x}').$$

For any collection of inputs $\{\mathbf{x}_i\}_{i=1}^N$, let $\widetilde{\boldsymbol{\Phi}} \in \mathbb{R}^{N \times H}$ have rows $\widetilde{\boldsymbol{\phi}}(\mathbf{x}_i)^\top$. The associated kernel matrix is $\mathbf{K} = \sigma_\varepsilon^2 \widetilde{\boldsymbol{\Phi}} \widetilde{\boldsymbol{\Phi}}^\top$, implying $\mathrm{rank}(\mathbf{K}) \le H$, independently of N.

Moreover, every realization of f is a linear combination of the learned feature maps and therefore lies in $\mathcal{H}_H = \mathrm{span}\{\phi_1, \ldots, \phi_H\}$, whose dimension is at most H. This establishes the claims about sample paths and rank of Proposition 2.1. □

Unlike standard squared-exponential and Matérn kernels, whose kernel matrices are full rank for distinct inputs (Williams & Rasmussen, 2006), BDKN's kernel has rank bounded by the ensemble size H. Thus, H explicitly bounds the dimension of the learned function space.

A.5 Datasets

We evaluated all methods on standard UCI regression benchmarks. Table 2 summarizes the key characteristics of each dataset, including the total number of samples, input dimensionality, number of random splits, and the corresponding training and test set sizes. Except for Song, experiments use a 90%/10% split into training and test sets repeated across multiple random splits as reported in the table; Song uses the fixed split into training and test sets shown in the table. The CO_2 dataset is used for the temporal gap experiments in Appendix A.10. In all experiments, input features are standardized using statistics computed on the training data, resulting in inputs with zero mean and unit variance.

Table 2: Dataset statistics for the regression benchmarks.

Dataset	Total Samples	# Features	# Splits	Train size	Test size
Boston	506	13	20	455	51
Concrete	1030	8	20	927	103
Energy	768	8	20	691	77
Kin8nm	8192	8	20	7373	819
Naval	11934	16	20	10741	1193
Power	9568	4	20	8611	957
Protein	45730	9	5	41157	4573
Sarcos	39449	21	20	35000	4449
Song	515345	90	1	463715	51630
Wine	1599	11	20	1439	160
Yacht	308	6	20	277	31
CO_2 (gap)	749	3	1	498	251

A.6 Setting 1: Training configuration and implementation details

Table 3 summarizes the optimization hyperparameters used in the experimental setting 1, which corresponds to a configuration with a high learning rate. Following the lightweight training regime commonly adopted in DE benchmarks, all models are trained with a learning rate of 0.1 for the full budget of 40 epochs on all datasets. Activation functions follow standard choices for each model class (ReLU for BDKN, DE, and VBLL; Leaky ReLU for BLL, LD-BLL, and MFVI-BNN), GP uses an RBF kernel, and DKL uses a ReLU feature extractor with an RBF GP layer. Results corresponding to this configuration are referred to as Setting 1 throughout the paper.

Table 3: Learning rates and maximum number of epochs for regression tasks (Setting 1 high learning rate). All neural network models use hidden dimensions $50, 50$.

Method Activation/Kernel	BDKN ReLU	DE ReLU	BLL LReLU	LD-BLL LReLU	MFVI-BNN LReLU	GP RBF	DKL ReLU/RBF	VBLL ReLU
Boston	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40
Concrete	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40
Energy	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40
Kin8nm	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40
Naval	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40
Power	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40
Protein	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	N/A	1e-1, 40	1e-1, 40
Sarcos	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	N/A	1e-1, 40	1e-1, 40
Song	1e-1, 40	1e-1, 40	1e-1, 40	N/A	1e-1, 40	N/A	1e-1, 40	1e-1, 40
Wine	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40
Yacht	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40	1e-1, 40

Unless otherwise specified, all methods based on neural networks use architectures with two fully connected hidden layers of width 50 and are trained with a batch size of 32 using the Adam optimizer. DKL uses the same ReLU feature extractor architecture as the other neural network baselines, with an output representation with one dimension, and places an RBF GP layer on top of the learned representation following Wilson et al. (2016). For MFVI-BNN, we follow the priors and inference procedure described by Watson et al. (2021), using independent $\mathcal{N}(0, \omega/\sqrt{n_{\text{in}}})$ priors for all network weights, where n_{in} denotes the number of input features to the corresponding layer and $\omega = 4$, and independent $\mathcal{N}(0, 1)$ priors for bias terms. Variational inference is implemented in `Pyro` using diagonal Gaussian variational distributions, optimized via stochastic variational inference (SVI) with the `Trace_ELBO` objective. For validation and evaluation, predictive distributions are estimated using 100 Monte Carlo samples from the variational posterior. VBLL is implemented using the official Python library (`VBLL`) provided by Harrison et al. (2024) with the default settings reported in the paper, and GP regression is implemented using `GPyTorch`.

For both experimental settings, we initialize the prior variance to $\sigma_\beta^2 = 1$ and estimate it through empirical Bayes by minimizing the negative marginal likelihood (100 iterations) using the Adam optimizer with a learning rate of 0.1, following Algorithm 1. We adopt the conjugate prior with zero mean $\boldsymbol{\beta} \mid \sigma_\varepsilon^2 \sim \mathcal{N}(\mathbf{0}, \sigma_\varepsilon^2 \sigma_\beta^2 \mathbf{I}_H)$, a standard choice in Bayesian last layer models (Watson et al., 2021).

A.7 BDKN ROBUSTNESS TO ENSEMBLE MEMBERS OF VARYING QUALITY

We examine BDKN's robustness to variation in ensemble member quality on Power and Sarcos, two datasets with large differences in predictive performance between BDKN and DE under Setting 1 in Table 1. We first analyze errors across splits and individual networks, explaining how Gaussian NLL training can lead to poor fits and how extreme predictions affect DE's RMSE and NLL. We then examine BDKN's posterior aggregation weights to show how it reduces the influence of poorly fitted members. Finally, we interpret the aggregation weights through their connection to ridge regression.

Errors across benchmark splits: BDKN and DE use exactly the same trained ensemble within each split. Table 4 compares DE's mean and median RMSE across 20 splits, alongside its mean NLL and BDKN's mean RMSE and NLL, separately for training and test data. The large gaps between DE's mean and median RMSE show that a small number of extreme splits disproportionately worsen its average performance. Our investigation also shows that many Power ensemble members remain poorly fitted after 40 epochs, contributing to a high median DE RMSE of 46.99. BDKN achieves much lower mean RMSE and NLL on both datasets using exactly the same trained neural networks, illustrating the benefit of its aggregation when individual members are poorly fitted.

Table 4: Training and test performance of DE and BDKN on Power and Sarcos in Setting 1. Means and medians are computed across 20 splits, separately for the training and test data.

		DE			BDKN	
Dataset	Data	Mean RMSE	Median RMSE	Mean NLL	Mean RMSE	Mean NLL
Power	Train	3468.45	47.91	6.89	4.73	2.96
Power	Test	3424.99	46.99	6.89	4.79	2.97
Sarcos	Train	55.20	7.46	4.02	3.98	2.81
Sarcos	Test	53.62	7.07	4.02	3.94	2.80

Individual network errors: Table 5 summarizes training and test RMSEs of individual networks across 20 splits. Rows 1–5 report statistics for each network index across the 20 separately trained ensembles. The final row combines all 100 RMSEs of individual networks from five networks and 20 splits. Each split trains new models, so a network index does not refer to the same trained model across splits.

These statistics describe individual networks, whereas Table 4 describes the aggregated predictions. Here, we compute each network's RMSE separately and then summarize those errors. For DE in Table 4, we first average the five networks' predictions within each split and then compute the ensemble's RMSE. Consequently, the mean RMSE of individual networks need not equal DE's mean RMSE, because individual prediction errors can partially cancel during averaging.

Power exhibits both extreme failures and broader fitting difficulties. Its mean RMSE of individual networks is 3459.11, while its median is 15.97. Twenty of the 100 networks have test RMSE above 100, spread across 15 splits, and four have RMSE above 10000. Sarcos's failures are more concentrated: four networks across three splits have RMSE above 100. Its median RMSE of individual networks is 5.17, compared with a mean of 61.36, indicating that a small number of extreme errors substantially raises the mean. The corresponding mean training RMSEs of individual networks are 3502.60 for Power and 62.97 for Sarcos, close to their mean test errors. Thus, the extreme errors are already present on the training data and support a fitting difficulty rather than a failure confined to generalization.

For Power split 11 (detailed results for these individual splits are not provided here; however, the exact splits used are available in the supplementary code), Network 2 has test RMSE 98790.41, and DE's RMSE reaches 19758.99. For Sarcos split 1, Networks 1 and 5 have test RMSEs of 3594.22

Table 5: Training and test RMSE statistics for individual networks for ensembles of five networks in Setting 1. Rows 1–5 summarize each network index across 20 splits. The final row summarizes all 100 RMSEs of individual networks for each data partition. Unlike Table 4, these statistics are computed before aggregating network predictions.

(a) Power.

	Training RMSE		Test RMSE	
Network	Mean	Median	Mean	Median
1	7919.27	16.45	7797.52	16.08
2	9428.31	25.10	9334.28	25.49
3	58.58	16.15	56.46	15.74
4	29.16	8.55	29.74	8.72
5	77.71	29.26	77.55	29.66
All networks and splits	3502.60	16.78	3459.11	15.97

(b) Sarcos.

	Training RMSE		Test RMSE	
Network	Mean	Median	Mean	Median
1	189.14	4.93	189.05	4.85
2	33.30	5.06	33.41	5.02
3	7.72	5.21	7.67	5.17
4	38.83	5.16	39.25	5.09
5	45.85	7.35	37.40	7.28
All networks and splits	62.97	5.18	61.36	5.17

and 520.76, respectively, and DE's RMSE is 722.79. The worst members also have training RMSEs of approximately 105385 for Power's Network 2 and 3595 for Sarcos's Network 1. These large training errors indicate poor fitting on the training data itself. In these example splits, BDKN assigns the respective worst members aggregation weights of approximately -0.000029 and -0.000026, whereas DE assigns every member 0.2.

Why Gaussian NLL training can produce poor fits: During training, each network predicts both a mean $\phi_h(\mathbf{x})$ and a variance $\sigma_h^2(\mathbf{x})$ by minimizing Gaussian NLL. For one observation, the training loss and its derivative with respect to the predicted mean are

$$\mathcal{L}_h(\mathbf{x}, y) = \frac{\big(y - \phi_h(\mathbf{x})\big)^2}{2\sigma_h^2(\mathbf{x})} + \frac{1}{2}\log\sigma_h^2(\mathbf{x}) + \frac{1}{2}\log(2\pi),$$
$$\frac{\partial \mathcal{L}_h}{\partial \phi_h(\mathbf{x})} = \frac{\phi_h(\mathbf{x}) - y}{\sigma_h^2(\mathbf{x})}.$$

For the same mean prediction error, a larger predicted variance reduces the gradient used to correct the mean. A network can therefore lower the loss by increasing its variance while simultaneously weakening the correction of its inaccurate mean prediction. This interaction can allow poor fits to persist, as analyzed by Seitzer et al. (2022).

Why extreme predictions affect DE's RMSE and NLL differently: After training, DE combines the individual network predictions with equal weights. For test evaluation, we use a Gaussian with the ensemble's predictive mean and variance. Writing $\mu_h(\mathbf{x}) = \phi_h(\mathbf{x})$ for member h's predictive mean, these moments are

$$\bar{\mu}(\mathbf{x}) = \frac{1}{H}\sum_{h=1}^{H}\mu_h(\mathbf{x}),$$
$$v(\mathbf{x}) = \frac{1}{H}\sum_{h=1}^{H}\sigma_h^2(\mathbf{x}) + \frac{1}{H}\sum_{h=1}^{H}\big(\mu_h(\mathbf{x}) - \bar{\mu}(\mathbf{x})\big)^2.$$

The first term in $v(\mathbf{x})$ averages the variances predicted by the individual networks; the second measures disagreement between their means. This disagreement enters the ensemble evaluation, whereas the training gradient above depends on each network's own predicted variance. The corresponding Gaussian NLL for each observation is

$$-\log p_{\mathrm{DE}}(y \mid \mathbf{x}) = \frac{1}{2}\log(2\pi) + \frac{1}{2}\log v(\mathbf{x}) + \frac{\big(y - \bar{\mu}(\mathbf{x})\big)^2}{2v(\mathbf{x})}.$$

An extreme member can increase both DE's prediction error and its predictive variance $v(\mathbf{x})$. For a fixed mean prediction error, increasing $v(\mathbf{x})$ decreases the squared error term $(y-\bar{\mu}(\mathbf{x}))^2/(2v(\mathbf{x}))$ in Gaussian NLL, while the term $\frac{1}{2}\log v(\mathbf{x})$ increases only logarithmically. The decrease can outweigh the logarithmic increase and lower NLL, explaining why DE's NLL values on Power and Sarcos in Table 1 are less extreme than its RMSE values.

Interpretability of BDKN aggregation weights: In a conventional BLL, the aggregation weights combine latent features within a single learned representation. In BDKN, each aggregation weight corresponds to an independently trained neural network that provides a complete prediction, giving the weights a direct interpretation at the level of individual predictive models. The posterior mean describes how each network contributes to the aggregate, while its credible interval quantifies uncertainty about the aggregation weight conditional on the learned feature maps.

To illustrate these contributions, we examine Power split 18 and Sarcos split 14 under Setting 1, whose benchmark results are reported in Table 1. These two splits were chosen for illustrative purposes and do not represent the best or worst fits across splits. Within each example, BDKN and DE use exactly the same trained ensemble. DE assigns each network an aggregation weight for the predictive mean of $1/H = 0.2$, whereas BDKN infers the aggregation weights jointly from the training data.

Figures 5 and 6 show individual network predictions, DE and BDKN predictions, and posterior aggregation weights. In each figure, the first and second rows show predicted versus observed values for the training and test data, respectively. The left panels display predictions from individual networks, while the right panels compare DE and BDKN. The dashed diagonal represents ideal agreement between predicted and observed values. The bottom panel shows posterior means of the aggregation weights and marginal 95% credible intervals. Tables 6 and 7 report the corresponding training and test RMSE and NLL, together with posterior mean aggregation weights and marginal 95% credible intervals.

Power: As shown in Figure 5 and Table 6, Networks 2, 4, and 5 have poor individual fits, with training RMSEs of 48.212, 52.389, and 57.029 and test RMSEs of 48.311, 52.927, and 56.885, respectively. BDKN assigns them posterior mean aggregation weights of -0.01503, -0.00436, and -0.00283, with 95% credible intervals of $[-0.01764, -0.01242]$, $[-0.00648, -0.00225]$, and $[-0.00478, -0.00088]$. These weights are small and negative, with intervals that exclude zero, reducing these networks' influence relative to DE's fixed aggregation weight of 0.2. Networks 1 and 3 receive positive posterior mean aggregation weights of 0.739 and 0.284, with intervals of $[0.724, 0.753]$ and $[0.269, 0.299]$, respectively. BDKN achieves a test RMSE of 5.352 and an NLL of 3.098, compared with 19.998 and 6.081, respectively, for DE.

Table 6: Predictive performance and BDKN posterior aggregation weights on the Power split shown in Figure 5. Lower RMSE and NLL are better; boldface highlights BDKN. Weight columns report posterior means and marginal 95% credible intervals for individual networks.

Model	Train RMSE	Train NLL	Test RMSE	Test NLL	Mean β_h	95% credible interval
Network 1	5.758	3.918	5.920	3.918	0.73885	$[0.72427, 0.75343]$
Network 2	48.212	5.754	48.311	5.752	-0.01503	$[-0.01764, -0.01242]$
Network 3	8.378	4.885	8.358	4.885	0.28400	$[0.26937, 0.29864]$
Network 4	52.389	6.767	52.927	6.759	-0.00436	$[-0.00648, -0.00225]$
Network 5	57.029	5.493	56.885	5.492	-0.00283	$[-0.00478, -0.00088]$
DE	19.522	6.089	19.998	6.081	—	—
BDKN	**5.129**	**3.054**	**5.352**	**3.098**	—	—

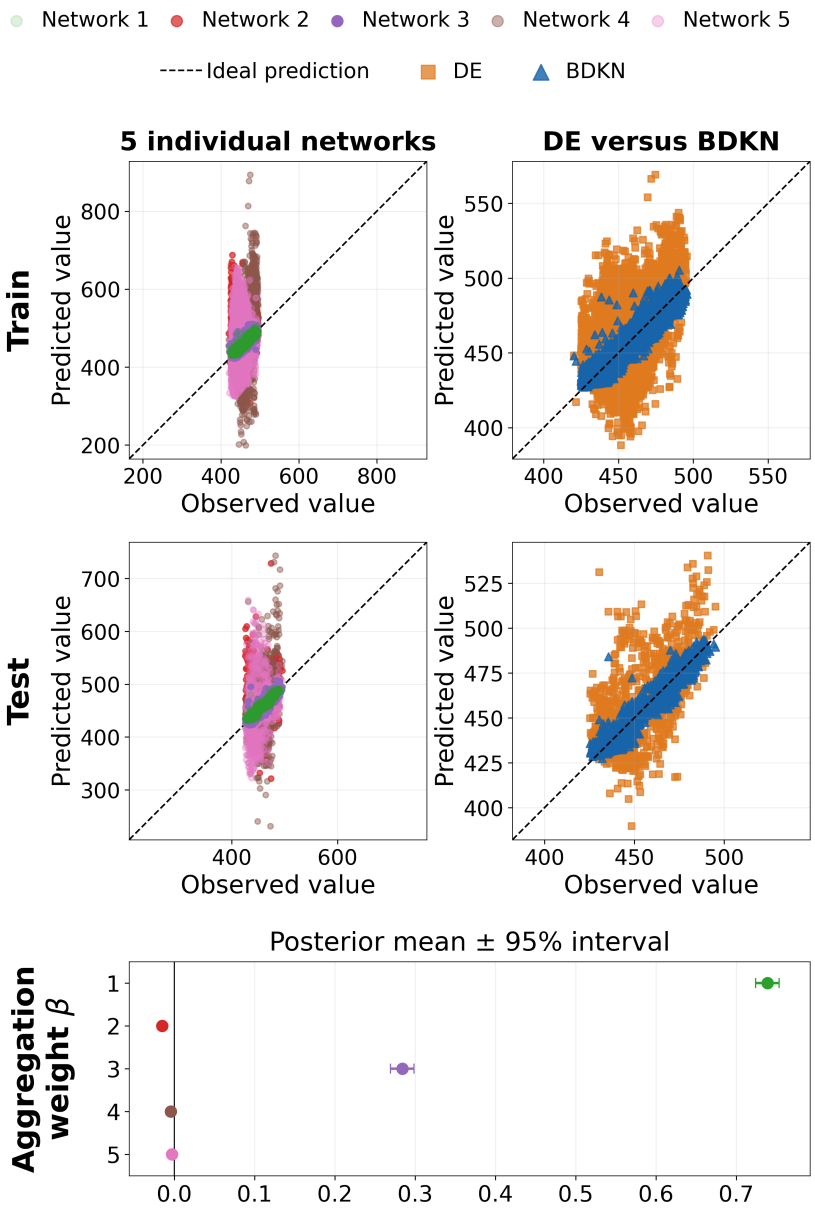


Figure 5: Interpretation of the posterior aggregation weights on Power (split 18). Training and test predictions appear in the first and second rows. The bottom panel shows posterior means of the aggregation weights and marginal 95% credible intervals.

Sarcos: As shown in Figure 6 and Table 7, Networks 1 and 2 have training RMSEs of 17.393 and 13.633 and test RMSEs of 17.467 and 13.512, respectively. Network 2 receives a posterior mean aggregation weight of -0.00950, with a 95% credible interval of $[-0.02270, 0.00369]$ that contains zero. Network 1 receives a smaller positive weight of 0.090, with an interval of $[0.083, 0.097]$. Networks 3, 4, and 5 receive the largest positive aggregation weights, with posterior means of 0.241, 0.229, and 0.481 and intervals of $[0.229, 0.253]$, $[0.218, 0.240]$, and $[0.468, 0.494]$, respectively. BDKN achieves a test RMSE of 3.798 and an NLL of 2.754, compared with 6.276 and 3.345, respectively, for DE.

Table 7: Predictive performance and BDKN posterior aggregation weights on the Sarcos split shown in Figure 6. Lower RMSE and NLL are better; boldface highlights BDKN. Weight columns report posterior means and marginal 95% credible intervals for individual networks.

Model	**Train RMSE**	**Train NLL**	**Test RMSE**	**Test NLL**	**Mean β_h**	**95% credible interval**
Network 1	17.393	3.927	17.467	3.933	0.09005	$[0.08280, 0.09731]$
Network 2	13.633	3.854	13.512	3.852	-0.00950	$[-0.02270, 0.00369]$
Network 3	4.684	2.824	4.617	2.805	0.24082	$[0.22861, 0.25302]$
Network 4	5.384	2.880	5.413	2.879	0.22904	$[0.21814, 0.23994]$
Network 5	4.308	2.826	4.227	2.805	0.48083	$[0.46809, 0.49357]$
DE	6.368	3.348	6.276	3.345	—	—
BDKN	**3.870**	**2.772**	**3.798**	**2.754**	—	—

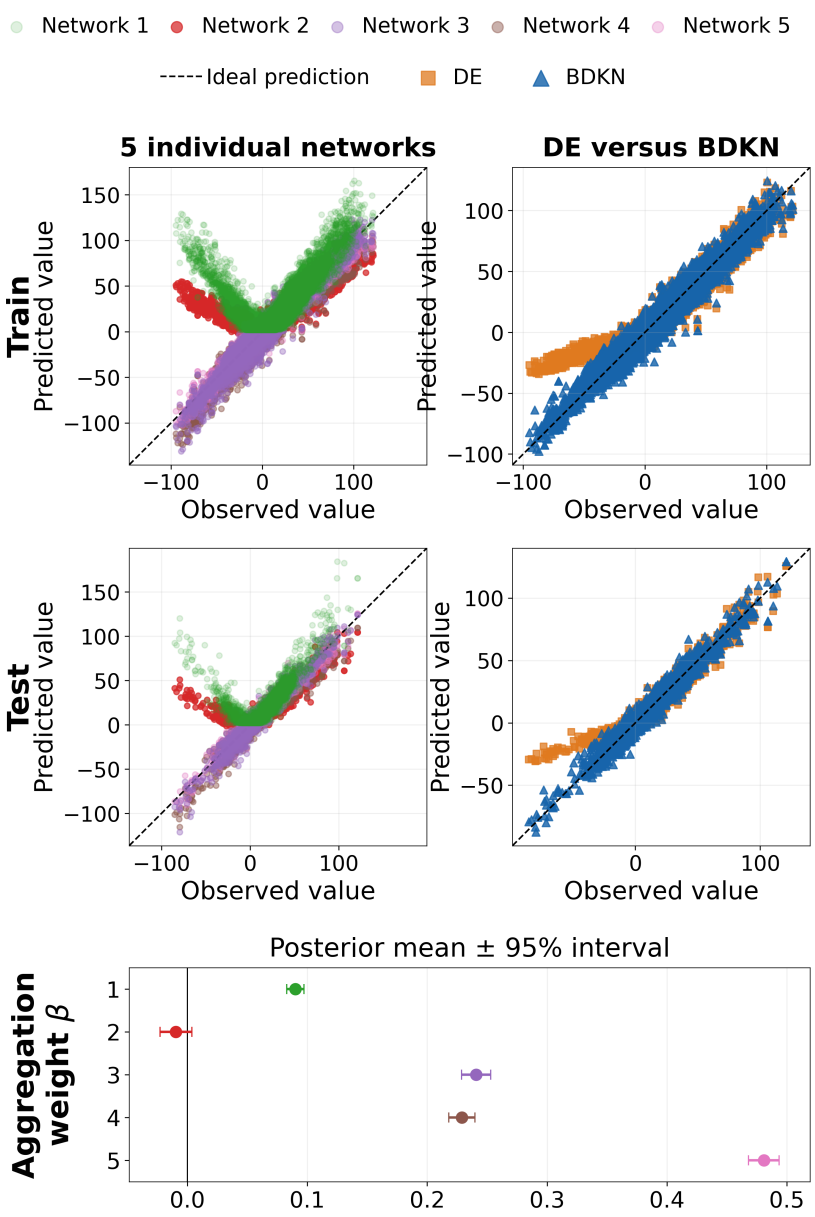


Figure 6: Interpretation of the posterior aggregation weights on Sarcos (split 14). Training and test predictions appear in the first and second rows. The bottom panel shows posterior means of the aggregation weights and marginal 95% credible intervals.

To better understand BDKN's aggregation behavior, we examine how the posterior means of the aggregation weights balance training fit with regularization from a ridge regression perspective.

Ridge regression interpretation: Ridge regression balances fitting the training data with a quadratic penalty on the aggregation weights (Hoerl & Kennard, 1970). For a learned feature map $\boldsymbol{\phi}(\mathbf{x})$, BDKN's predictive mean is $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\mu}_n$, while DE uses $\boldsymbol{\phi}(\mathbf{x})^\top \mathbf{u}_H$, where $\mathbf{u}_H := H^{-1}\mathbf{1}_H$ and $\mathbf{1}_H$ is the H-dimensional vector of ones. Both predictive means lie in the same linear function space $\mathcal{H}_H$.

Combining the Gaussian likelihood for $\mathbf{y}$ with the conditional Gaussian prior on $\boldsymbol{\beta}$ yields the objective $J_{\boldsymbol{\Lambda}_0}(\boldsymbol{\beta}) := \|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2 + (\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Lambda}_0(\boldsymbol{\beta} - \boldsymbol{\mu}_0)$. Its two terms measure squared training error and penalize deviations from the prior mean, respectively. Under our isotropic prior with zero mean on $\boldsymbol{\beta}$, the penalty becomes $\lambda\|\boldsymbol{\beta}\|_2^2$, where $\lambda := \sigma_\beta^{-2}$ is the ridge regularization parameter.

Proposition A.1 (Ridge regression equivalence)**.** *For fixed $\boldsymbol{\Phi}$, any prior mean $\boldsymbol{\mu}_0$, and any positive-definite prior precision $\boldsymbol{\Lambda}_0$, the BDKN posterior mean $\boldsymbol{\mu}_n$ is the unique minimizer of $J_{\boldsymbol{\Lambda}_0}$.*

Consequently, $J_{\boldsymbol{\Lambda}_0}(\boldsymbol{\mu}_n) \le J_{\boldsymbol{\Lambda}_0}(\mathbf{u}_H)$, with equality if and only if $\boldsymbol{\mu}_n = \mathbf{u}_H$. Thus, BDKN's posterior means of the aggregation weights achieve no larger regularized training loss than DE's uniform aggregation weights on the same learned feature maps. The isotropic prior with zero mean on $\boldsymbol{\beta}$ regularizes all aggregation weights symmetrically, while their inferred values reflect how the feature maps jointly contribute to fitting the data. A poorly fitted member that contributes little beyond the other ensemble members may therefore receive an aggregation weight close to zero.

Proof. Condition on the learned design matrix $\boldsymbol{\Phi}$ and on σ_ε^2. The Gaussian likelihood has kernel

$$p(\mathbf{y} \mid \boldsymbol{\beta}, \sigma_\varepsilon^2, \boldsymbol{\Phi}) \propto \exp\left\{-\frac{1}{2\sigma_\varepsilon^2}\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2\right\}.$$

For an arbitrary mean $\boldsymbol{\mu}_0$ and positive-definite precision $\boldsymbol{\Lambda}_0$, the scaled Gaussian prior is

$$\boldsymbol{\beta} \mid \sigma_\varepsilon^2 \sim \mathcal{N}\left(\boldsymbol{\mu}_0, \sigma_\varepsilon^2 \boldsymbol{\Lambda}_0^{-1}\right),$$

and hence has kernel

$$p(\boldsymbol{\beta} \mid \sigma_\varepsilon^2) \propto \exp\left\{-\frac{1}{2\sigma_\varepsilon^2}(\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Lambda}_0(\boldsymbol{\beta} - \boldsymbol{\mu}_0)\right\}.$$

Bayes' rule gives

$$\begin{aligned}
p(\boldsymbol{\beta} \mid \sigma_\varepsilon^2, \mathbf{y}, \boldsymbol{\Phi}) &\propto p(\mathbf{y} \mid \boldsymbol{\beta}, \sigma_\varepsilon^2, \boldsymbol{\Phi}) p(\boldsymbol{\beta} \mid \sigma_\varepsilon^2) \\
&\propto \exp\left\{-\frac{1}{2\sigma_\varepsilon^2}\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2\right\} \exp\left\{-\frac{1}{2\sigma_\varepsilon^2}(\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Lambda}_0(\boldsymbol{\beta} - \boldsymbol{\mu}_0)\right\} \\
&= \exp\left\{-\frac{1}{2\sigma_\varepsilon^2}\left(\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2 + (\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Lambda}_0(\boldsymbol{\beta} - \boldsymbol{\mu}_0)\right)\right\} \\
&= \exp\left\{-\frac{1}{2\sigma_\varepsilon^2} J_{\boldsymbol{\Lambda}_0}(\boldsymbol{\beta})\right\},
\end{aligned}$$

where

$$J_{\boldsymbol{\Lambda}_0}(\boldsymbol{\beta}) := \|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2 + (\boldsymbol{\beta} - \boldsymbol{\mu}_0)^\top \boldsymbol{\Lambda}_0(\boldsymbol{\beta} - \boldsymbol{\mu}_0).$$

Thus, minimizing the negative log posterior with respect to $\boldsymbol{\beta}$ is equivalent to minimizing $J_{\boldsymbol{\Lambda}_0}$. This is a generalized ridge regression objective: the first term measures squared training error, while the second regularizes the aggregation weights toward $\boldsymbol{\mu}_0$. For BDKN's isotropic prior with zero mean, it reduces to standard ridge regression.

To obtain its minimizer explicitly, differentiate:

$$\begin{aligned}
\nabla_{\boldsymbol{\beta}} J_{\boldsymbol{\Lambda}_0}(\boldsymbol{\beta}) &= -2\boldsymbol{\Phi}^\top(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}) + 2\boldsymbol{\Lambda}_0(\boldsymbol{\beta} - \boldsymbol{\mu}_0) \\
&= 2\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Lambda}_0\right)\boldsymbol{\beta} - 2\left(\boldsymbol{\Phi}^\top\mathbf{y} + \boldsymbol{\Lambda}_0\boldsymbol{\mu}_0\right),
\end{aligned}$$

with Hessian

$$\nabla_{\boldsymbol{\beta}}^2 J_{\boldsymbol{\Lambda}_0}(\boldsymbol{\beta}) = 2\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Lambda}_0\right).$$

For every nonzero $\mathbf{v} \in \mathbb{R}^H$,

$$\mathbf{v}^\top(\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Lambda}_0)\mathbf{v} = \|\mathbf{\Phi}\mathbf{v}\|_2^2 + \mathbf{v}^\top\mathbf{\Lambda}_0\mathbf{v} > 0.$$

Therefore, the Hessian is positive definite and $J_{\mathbf{\Lambda}_0}$ is strictly convex. Let $\widehat{\boldsymbol{\beta}}$ denote the minimizer. Setting the gradient equal to zero at this initially unknown value gives

$$\begin{aligned}\mathbf{0} &= \nabla_{\boldsymbol{\beta}} J_{\mathbf{\Lambda}_0}(\widehat{\boldsymbol{\beta}}) \\ &= -2\mathbf{\Phi}^\top(\mathbf{y} - \mathbf{\Phi}\widehat{\boldsymbol{\beta}}) + 2\mathbf{\Lambda}_0(\widehat{\boldsymbol{\beta}} - \boldsymbol{\mu}_0).\end{aligned}$$

Dividing by two and rearranging gives

$$\left(\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Lambda}_0\right)\widehat{\boldsymbol{\beta}} = \mathbf{\Phi}^\top\mathbf{y} + \mathbf{\Lambda}_0\boldsymbol{\mu}_0.$$

Since $\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Lambda}_0$ is positive definite and therefore invertible, the unique solution is

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Lambda}_0\right)^{-1}\left(\mathbf{\Phi}^\top\mathbf{y} + \mathbf{\Lambda}_0\boldsymbol{\mu}_0\right) = \boldsymbol{\mu}_n. \tag{11}$$

The final equality follows from the posterior mean update in Equation (6), where $\mathbf{\Lambda}_n = \mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Lambda}_0$. Thus, the BDKN posterior mean is mathematically identical to the generalized ridge solution.

For the exact comparison with DE, expand the objective as

$$\begin{aligned}J_{\mathbf{\Lambda}_0}(\boldsymbol{\beta}) = \mathbf{y}^\top\mathbf{y} + \boldsymbol{\mu}_0^\top\mathbf{\Lambda}_0\boldsymbol{\mu}_0 \\ - 2\boldsymbol{\beta}^\top\left(\mathbf{\Phi}^\top\mathbf{y} + \mathbf{\Lambda}_0\boldsymbol{\mu}_0\right) + \boldsymbol{\beta}^\top\mathbf{\Lambda}_n\boldsymbol{\beta}.\end{aligned}$$

Equation (11) implies

$$\mathbf{\Phi}^\top\mathbf{y} + \mathbf{\Lambda}_0\boldsymbol{\mu}_0 = \mathbf{\Lambda}_n\boldsymbol{\mu}_n.$$

Substituting this equality into the expansion and subtracting $J_{\mathbf{\Lambda}_0}(\boldsymbol{\mu}_n)$ cancels the terms that do not depend on $\boldsymbol{\beta}$, leaving

$$\begin{aligned}J_{\mathbf{\Lambda}_0}(\boldsymbol{\beta}) - J_{\mathbf{\Lambda}_0}(\boldsymbol{\mu}_n) &= \boldsymbol{\beta}^\top\mathbf{\Lambda}_n\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top\mathbf{\Lambda}_n\boldsymbol{\mu}_n + \boldsymbol{\mu}_n^\top\mathbf{\Lambda}_n\boldsymbol{\mu}_n \\ &= (\boldsymbol{\beta} - \boldsymbol{\mu}_n)^\top\mathbf{\Lambda}_n(\boldsymbol{\beta} - \boldsymbol{\mu}_n).\end{aligned}$$

For DE, set $\boldsymbol{\beta} = \mathbf{u}_H$. Then

$$J_{\mathbf{\Lambda}_0}(\mathbf{u}_H) - J_{\mathbf{\Lambda}_0}(\boldsymbol{\mu}_n) = (\mathbf{u}_H - \boldsymbol{\mu}_n)^\top\mathbf{\Lambda}_n(\mathbf{u}_H - \boldsymbol{\mu}_n) \geq 0,$$

which proves the stated inequality. Positive definiteness implies equality if and only if $\mathbf{u}_H = \boldsymbol{\mu}_n$. □

The ridge equivalence characterizes the posterior means of the aggregation weights, explaining how the learned predictors are combined through a joint balance of training fit and regularization. The posterior credible intervals complement this interpretation by quantifying uncertainty in each aggregation weight conditional on the learned predictors. Together, the means and intervals describe both each network's contribution to the aggregate and the uncertainty associated with that contribution.

A.8 Runtime comparison under Setting 1

All experiments were conducted on a workstation running Windows 10 Enterprise (64-bit), equipped with a 12th-generation Intel Core i7-12700 CPU (12 cores, 20 logical processors), 32 GB of RAM, and an NVIDIA RTX A2000 GPU with 12 GB of dedicated memory. The system uses NVMe SSD storage. GPU computations were performed using CUDA 12.8 with NVIDIA driver version 573.44. The runtimes correspond to the complete training procedure on standard regression benchmarks using the same hardware and implementation framework.

Table 8 reports the training time (in minutes) for all methods under Setting 1, using the hardware and measurement procedure described above (Runtime results for Setting 2 are reported in Appendix A.9). Under Setting 1 (high learning rate), BDKN exhibits runtime comparable to standard DE, reflecting that its additional Bayesian aggregation step incurs negligible overhead relative to training the ensemble members. In contrast, fully Bayesian or variational approaches, such as MFVI-BNN and VBLL, incur substantially higher computational costs on larger datasets due to inference based on sampling or variational optimization. GP is efficient on smaller datasets but becomes expensive or infeasible on larger datasets, such as *Protein, Sarcos*, and *Song*.

Table 8: Wall clock runtime (minutes) on standard regression benchmarks (Setting 1).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
Boston	0.76	0.66	0.36	0.10	0.67	0.06	2.06	1.43
Concrete	0.75	0.66	1.40	0.59	0.30	0.23	3.59	2.19
Energy	0.54	0.46	0.15	0.45	0.52	0.09	3.36	1.79
Kin8nm	6.08	5.99	1.64	2.55	0.72	0.19	5.14	17.61
Naval	7.30	7.21	0.46	4.90	1.91	0.05	10.43	18.90
Power	6.40	6.32	0.24	3.91	1.29	0.02	8.17	17.68
Protein	8.47	8.45	0.44	N/A	0.27	0.13	16.24	25.56
Sarcos	30.04	29.95	2.52	N/A	2.29	0.47	63.65	80.45
Song	17.51	17.50	N/A	N/A	15.04	0.39	37.28	99.67
Wine	2.08	1.94	0.19	1.31	1.19	0.04	5.45	6.83
Yacht	0.61	0.45	0.32	0.16	0.89	0.15	5.86	1.33

A.9 Results for Setting 2 (low learning rate)

Training configuration: Setting 2 uses the more conservative, dataset-specific learning rates and training schedules of Watson et al. (2021) to assess robustness to variation in optimization quality and whether BDKN remains competitive under more extensive training. Table 9 summarizes the optimization hyperparameters. All other experimental details remain unchanged from Setting 1, except that BLL, LD-BLL, and MFVI-BNN use three hidden layers with widths 50, 200, and 200 on Sarcos. The remaining neural-network architectures retain two fully connected hidden layers with 50 neurons each, and the batch size remains 32 for all neural-network-based approaches.

BDKN, DE, DKL, and VBLL are trained for the full budget of 500 epochs on all datasets. For BLL, LD-BLL, and MFVI-BNN, the number of training epochs is selected via validation: for each split, 80% of the training data are used for optimization and the remaining 20% determine the epoch with the best validation performance. The model is then retrained on the full training set using the selected number of epochs. For GP regression, validation-based early stopping is computationally expensive. We therefore terminate optimization when the relative decrease in the average marginal likelihood over the most recent κ iterations, compared to the average over the preceding κ iterations, falls below a threshold ρ, following a procedure similar to that of Watson et al. (2021). We set $\kappa = 11$ and $\rho = 10^{-4}$ for all datasets.

Table 9: Learning rates and maximum number of epochs for regression tasks (Setting 2 low learning rate). All neural network models use hidden dimensions $50, 50$, except on Sarcos where BLL, LD-BLL, and MFVI-BNN use hidden dimensions $50, 200, 200$.

Method Activation/Kernel	BDKN ReLU	DE ReLU	BLL LReLU	LD-BLL LReLU	MFVI-BNN LReLU	GP RBF	DKL ReLU/RBF	VBLL ReLU
Boston	1e-3, 500	1e-3, 500	1e-3, 3000	1e-3, 4000	1e-3, 10000	1e-2, 1000	1e-3, 500	1e-3, 500
Concrete	1e-3, 500	1e-3, 500	1e-3, 3000	1e-3, 4000	1e-3, 15000	1e-2, 1000	1e-3, 500	1e-3, 500
Energy	1e-3, 500	1e-3, 500	1e-3, 8000	1e-3, 10000	1e-3, 30000	1e-2, 2000	1e-3, 500	1e-3, 500
Kin8nm	1e-3, 500	1e-3, 500	1e-3, 3000	1e-3, 5000	1e-3, 20000	1e-2, 1000	1e-3, 500	1e-3, 500
Naval	1e-3, 500	1e-3, 500	1e-3, 8000	1e-3, 10000	1e-3, 100000	1e-3, 3000	1e-3, 500	1e-3, 500
Power	1e-3, 500	1e-3, 500	1e-3, 5000	1e-3, 5000	1e-3, 20000	1e-2, 1000	1e-3, 500	1e-3, 500
Protein	1e-3, 500	1e-3, 500	1e-3, 2000	1e-3, 5000	1e-3, 30000	N/A	1e-3, 500	1e-3, 500
Sarcos	1e-3, 500	1e-3, 500	2e-4, 30000	2e-4, 30000	5e-3, 10000	N/A	1e-3, 500	1e-3, 500
Song	1e-3, 500	1e-3, 500	1e-3, 500	N/A	1e-3, 500	N/A	1e-3, 500	1e-3, 500
Wine	1e-3, 500	1e-3, 500	1e-4, 1000	1e-4, 1000	1e-4, 20000	1e-2, 1000	1e-3, 500	1e-3, 500
Yacht	1e-3, 500	1e-3, 500	1e-3, 8000	1e-3, 10000	1e-3, 20000	1e-2, 2000	1e-3, 500	1e-3, 500

Benchmark results: Table 10 summarizes test RMSE and NLL under Setting 2. Methods exhibit increased stability under this more conservative optimization regime. BDKN remains competitive in both metrics and continues to provide well-calibrated uncertainty across datasets. DE performs strongly, with reduced variations in optimization quality, while GP again achieves strong performance on several benchmarks. BLL-based and variational methods exhibit dataset-dependent behavior. The results for LD-BLL and VBLL are consistent with those reported by Harrison et al. (2024).

MFVI-BNN also shows strong and stable performance, but at a substantially higher computational cost, as shown in Table 11.

Table 10: **Experimental setting 2 (low learning rate): performance on standard regression benchmarks. (a)** RMSE and **(b)** NLL reported as mean ± standard deviation across splits. Lower values indicate better performance. **Bold** indicates the best performance, and underlined indicates the second-best performance. GP could not be evaluated on the Protein, Sarcos, and Song datasets, and LD-BLL on the Song dataset, due to their memory requirements, which exceeded our GPU memory.

(a) RMSE (mean ± std).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
Boston	3.01 ± 0.78	2.99 ± 0.85	4.41 ± 1.20	**2.83 ± 0.73**	4.11 ± 1.18	3.20 ± 0.73	2.89 ± 0.68	2.92 ± 0.77
Concrete	5.22 ± 0.58	5.23 ± 0.59	7.03 ± 0.58	5.63 ± 0.62	5.83 ± 0.93	5.24 ± 0.69	**4.55 ± 0.60**	5.11 ± 0.62
Energy	0.76 ± 0.21	1.42 ± 0.29	0.91 ± 0.71	0.47 ± 0.06	1.02 ± 0.50	0.47 ± 0.07	**0.43 ± 0.07**	0.48 ± 0.06
Kin8nm	**0.07 ± 0.00**	**0.07 ± 0.00**	0.10 ± 0.03	**0.07 ± 0.00**	0.09 ± 0.01	0.10 ± 0.00	**0.07 ± 0.00**	**0.07 ± 0.00**
Naval	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	0.01 ± 0.00	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
Power	3.93 ± 0.16	3.98 ± 0.16	3.87 ± 0.18	**3.72 ± 0.17**	4.36 ± 0.51	4.05 ± 0.17	3.87 ± 0.15	4.22 ± 0.20
Protein	4.05 ± 0.02	4.06 ± 0.03	4.17 ± 0.14	N/A	4.22 ± 0.06	4.09 ± 0.05	**3.87 ± 0.01**	3.95 ± 0.04
Sarcos	2.13 ± 0.03	2.14 ± 0.03	9.25 ± 1.13	N/A	3.46 ± 0.90	**1.61 ± 0.06**	2.09 ± 0.07	2.21 ± 0.05
Song	8.71 ± 0.00	**8.69 ± 0.00**	N/A	N/A	9.08 ± 0.00	9.05 ± 0.00	9.61 ± 0.00	9.14 ± 0.00
Wine	0.65 ± 0.06	0.63 ± 0.05	0.65 ± 0.03	**0.56 ± 0.04**	0.76 ± 0.06	0.64 ± 0.04	0.63 ± 0.04	0.72 ± 0.06
Yacht	0.85 ± 0.38	1.17 ± 0.44	3.25 ± 5.68	**0.40 ± 0.13**	2.12 ± 1.89	2.36 ± 0.52	0.86 ± 0.42	0.63 ± 0.29

(b) NLL (mean ± std).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
Boston	2.81 ± 0.62	2.52 ± 0.37	3.30 ± 0.05	**2.41 ± 0.28**	13.05 ± 7.29	3.21 ± 0.03	3.27 ± 0.95	3.25 ± 1.16
Concrete	3.23 ± 0.22	**3.01 ± 0.18**	3.83 ± 0.02	3.42 ± 0.04	23.59 ± 7.38	3.79 ± 0.01	3.80 ± 0.69	3.24 ± 0.26
Energy	1.17 ± 0.30	1.71 ± 1.74	1.56 ± 0.49	**0.66 ± 0.16**	1.62 ± 1.23	3.23 ± 0.00	3.25 ± 1.32	0.88 ± 0.27
Kin8nm	-1.18 ± 0.07	**-1.35 ± 0.03**	-0.68 ± 0.21	-0.58 ± 0.05	0.55 ± 0.00	-0.34 ± 0.01	-1.13 ± 0.04	-0.97 ± 0.07
Naval	-4.76 ± 0.03	-5.33 ± 0.18	-4.59 ± 0.76	-4.70 ± 0.01	0.54 ± 0.00	-3.30 ± 0.00	**-8.10 ± 0.19**	-3.70 ± 0.01
Power	2.79 ± 0.05	2.79 ± 0.04	**2.78 ± 0.05**	3.19 ± 0.01	12.11 ± 3.06	3.78 ± 0.00	**2.78 ± 0.04**	2.86 ± 0.05
Protein	2.80 ± 0.01	**2.74 ± 0.23**	2.86 ± 0.03	N/A	10.68 ± 0.25	2.95 ± 0.01	2.81 ± 0.01	2.80 ± 0.01
Sarcos	2.19 ± 0.02	**1.83 ± 0.01**	4.14 ± 0.43	N/A	9.04 ± 5.00	3.94 ± 0.00	2.17 ± 0.03	2.21 ± 0.02
Song	**3.30 ± 0.00**	3.33 ± 0.00	N/A	N/A	3.63 ± 0.00	3.66 ± 0.00	3.99 ± 0.00	3.65 ± 0.00
Wine	1.64 ± 0.36	1.11 ± 0.24	1.03 ± 0.03	**0.86 ± 0.06**	1.16 ± 0.11	1.02 ± 0.03	0.94 ± 0.06	3.32 ± 0.67
Yacht	1.48 ± 0.84	**-0.20 ± 0.26**	1.79 ± 0.39	0.17 ± 0.12	6.12 ± 13.77	3.66 ± 0.01	1.41 ± 1.20	2.22 ± 2.63

Test data performance over training epochs: Figure 7 shows test RMSE and NLL over the first 500 training epochs on Boston under Setting 2. GP demonstrates good performance, although with slower convergence. BLL exhibits limited improvement, converging quickly but to higher RMSE and NLL values. LD-BLL initially degrades after the first few epochs and then begins to converge very slowly. DKL also shows degradation after a few epochs. In contrast, BDKN and DE maintain fast and stable convergence throughout training.

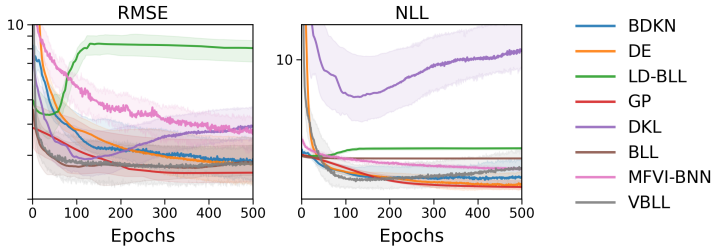


Figure 7: **Boston dataset - Test data performance over training epochs under Setting 2 (low learning rate).** We show quartile plots of **test RMSE** (left) and **test NLL** (right) as functions of training epochs. Solid lines denote the median across 20 splits, while shaded regions indicate the interquartile range (25th–75th percentiles). Lower values are better for both RMSE and NLL.

Wall-clock runtime: Table 11 reports training times measured using the workstation and procedure described in Appendix A.8. Compared with Setting 1 (Table 8), runtimes increase across all neural-network-based methods because of the longer training schedules, with particularly large increases for MFVI-BNN on datasets with many features or observations. BDKN retains runtime comparable to DE across both settings and remains significantly more efficient than fully Bayesian baselines. On the three largest datasets (Protein, Sarcos, and Song), it is substantially more efficient than MFVI-BNN and VBLL, while GP could not be evaluated due to high memory requirements. Entries marked as *N/A* indicate cases where a method could not be evaluated due to GPU memory constraints.

Table 11: Wall clock runtime (minutes) on standard regression benchmarks (Setting 2).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
Boston	4.07	4.07	14.68	33.42	8.93	20.07	902.33	13.27
Concrete	8.31	8.26	11.36	30.79	14.86	10.59	829.53	28.13
Energy	6.73	6.72	113.53	14.79	11.75	29.54	3176.77	20.19
Kin8nm	68.56	68.54	122.19	55.82	35.90	0.94	2357.65	234.30
Naval	91.10	91.09	241.29	336.05	22.15	127.35	14625.90	302.48
Power	80.87	80.87	22.39	81.07	11.58	85.03	1960.26	421.37
Protein	111.35	111.35	106.79	N/A	2.70	14.54	6099.07	358.99
Sarcos	362.98	362.98	8494.25	N/A	19.62	669.92	7993.33	1256.18
Song	229.29	229.28	N/A	N/A	78.31	5.62	488.23	765.11
Wine	12.18	12.17	4.81	19.03	7.99	34.93	2267.03	47.35
Yacht	3.20	3.19	117.50	11.15	8.34	37.28	2031.11	8.49

A.10 Out-of-distribution gap experiments

We evaluate predictive performance and uncertainty in regions excluded from training using feature based and temporal gap splits. For the UCI datasets, we follow Foong et al. (2019); for each input dimension j, observations are sorted by that feature, the central third is withheld for testing, and the lower and upper thirds are retained for training. A dataset with D input dimensions therefore has D gap splits. The selected feature determines only the split; all methods receive the complete D-dimensional input vector. We also consider the CO_2 dataset, a time series of atmospheric carbon dioxide concentrations. We use the temporal split adopted by Watson et al. (2021), with selected time intervals reserved for testing, as shown in Figure 8.

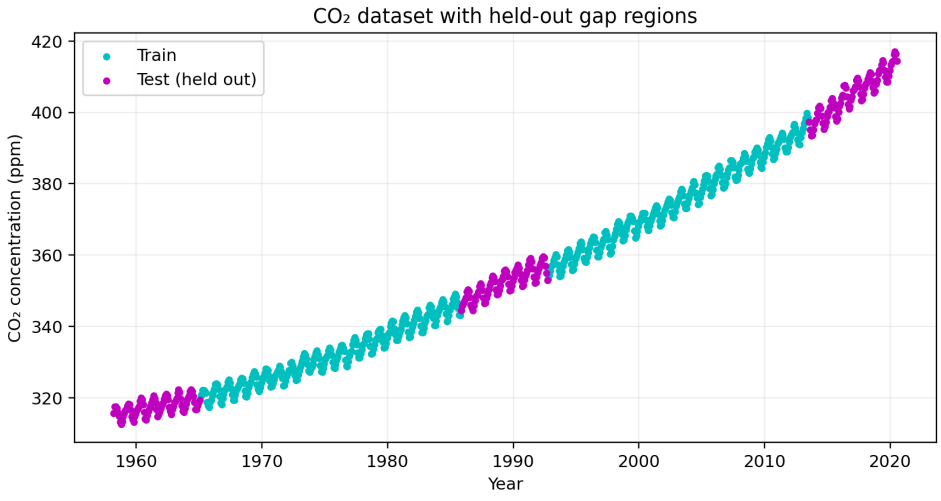


Figure 8: CO_2 temporal gap split. Cyan points are used for training and magenta points for testing. The regions excluded from training include an internal time interval and periods before and after the training observations.

All eight methods are evaluated under both training configurations: Setting 1 (Appendix A.6) and Setting 2 (Appendix A.9). Setting 1 uses the configuration with a high learning rate, and Setting 2 uses the more conservative configuration.

Table 12 reports test RMSE and NLL for the gap experiments under Setting 1. BDKN achieves competitive results across the datasets in both metrics. These results broadly follow the trends observed on the standard regression benchmarks in Table 1, supporting BDKN's predictive performance in regions excluded from training.

Table 12: **Setting 1: gap regression performance.** Test RMSE and NLL are reported in the original target space as mean ± standard deviation across feature based gap splits; CO_2 has one temporal split. Lower is better. Bold and underlined entries indicate the best and second best results, respectively, based on unrounded values. N/A denotes GP memory limitations on Protein.

(a) Test RMSE (mean ± standard deviation).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
CO_2	3.50 ± 0.00	7.69 ± 0.00	40.07 ± 0.00	**2.20 ± 0.00**	7.36 ± 0.00	11.09 ± 0.00	9.60 ± 0.00	3.54 ± 0.00
Boston	3.93 ± 0.67	4.12 ± 0.79	9.31 ± 3.08	**3.79 ± 0.39**	4.36 ± 0.60	4.57 ± 0.85	6.37 ± 1.55	4.54 ± 0.95
Concrete	**7.95 ± 0.85**	10.24 ± 3.17	42.42 ± 37.99	8.06 ± 1.17	8.76 ± 0.94	11.06 ± 2.18	13.55 ± 1.23	7.95 ± 0.73
Kin8nm	0.14 ± 0.02	1.66 ± 3.82	0.42 ± 0.45	**0.08 ± 0.00**	0.10 ± 0.01	0.11 ± 0.01	0.24 ± 0.03	0.20 ± 0.04
Power	5.35 ± 0.56	18.32 ± 5.32	6.93 ± 2.17	**4.22 ± 0.22**	4.36 ± 0.26	6.09 ± 1.70	8.19 ± 2.23	6.80 ± 1.03
Protein	5.15 ± 0.18	54.96 ± 96.38	8.35 ± 2.87	N/A	**4.93 ± 0.19**	5.26 ± 0.28	5.62 ± 0.26	5.24 ± 0.15
Wine	0.63 ± 0.03	1.31 ± 1.56	1.16 ± 0.62	**0.63 ± 0.03**	0.65 ± 0.03	0.86 ± 0.04	0.72 ± 0.08	0.68 ± 0.04
Yacht	**1.46 ± 0.46**	2.88 ± 1.67	9.94 ± 8.87	7.36 ± 3.24	3.15 ± 1.26	11.74 ± 4.82	12.57 ± 3.91	1.81 ± 0.70

(b) Test NLL (mean ± standard deviation).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
CO_2	4.49 ± 0.00	3.46 ± 0.00	10.78 ± 0.00	**2.30 ± 0.00**	4.61 ± 0.00	4.31 ± 0.00	4.26 ± 0.00	2.84 ± 0.00
Boston	3.04 ± 0.43	3.09 ± 0.58	5.05 ± 0.22	**2.68 ± 0.11**	3.12 ± 0.31	3.35 ± 0.07	3.34 ± 0.17	3.24 ± 0.74
Concrete	3.72 ± 0.26	3.97 ± 0.52	4.79 ± 0.22	**3.43 ± 0.13**	3.69 ± 0.18	3.99 ± 0.07	4.06 ± 0.09	3.81 ± 0.31
Kin8nm	-0.53 ± 0.12	0.49 ± 1.48	1.03 ± 0.32	-0.69 ± 0.01	**-0.89 ± 0.10**	-0.31 ± 0.02	0.01 ± 0.16	-0.16 ± 0.21
Power	3.12 ± 0.13	5.36 ± 0.37	3.37 ± 0.32	**2.87 ± 0.05**	2.93 ± 0.08	3.93 ± 0.07	3.82 ± 0.16	3.34 ± 0.14
Protein	3.07 ± 0.04	4.07 ± 1.49	4.06 ± 0.11	N/A	**3.04 ± 0.06**	3.10 ± 0.04	3.22 ± 0.07	3.10 ± 0.04
Wine	0.97 ± 0.06	1.57 ± 0.89	2.31 ± 0.19	**0.96 ± 0.03**	0.99 ± 0.05	1.27 ± 0.05	1.13 ± 0.11	1.05 ± 0.05
Yacht	2.44 ± 1.17	2.35 ± 0.70	3.31 ± 0.60	3.28 ± 0.32	222.13 ± 490.81	4.00 ± 0.17	3.93 ± 0.25	**1.91 ± 0.40**

Table 13 reports wall clock runtimes across all gap splits. BDKN and DE have wall clock runtimes within 0.01 minutes of each other at the reported precision.

Table 13: Wall clock runtime (minutes) on gap regression benchmarks (Setting 1).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
CO_2	0.02	0.02	0.01	0.00	0.04	0.00	0.01	0.09
Boston	0.20	0.20	0.07	0.04	0.58	0.03	0.12	0.60
Concrete	0.26	0.26	0.05	0.03	0.36	0.02	0.08	0.91
Kin8nm	2.02	2.02	0.12	0.42	0.45	0.02	0.07	7.63
Power	1.19	1.19	0.06	0.23	0.25	0.01	0.04	4.18
Protein	11.96	11.95	0.49	N/A	0.75	0.03	0.09	44.73
Wine	0.56	0.56	0.11	0.18	0.58	0.02	0.11	1.82
Yacht	0.08	0.08	0.05	0.03	0.26	0.01	0.06	0.26

Table 14 reports test RMSE and NLL for the gap experiments under Setting 2. BDKN also achieves competitive results across the datasets in both metrics under the more conservative training configuration.

Table 15 shows similarly close wall clock runtimes for BDKN and DE, within 0.01 minutes of each other at the reported precision.

Table 14: **Setting 2: gap regression performance.** Test RMSE and NLL are reported in the original target space as mean ± standard deviation across feature based gap splits; CO_2 has one temporal split. Lower is better. Bold and underlined entries indicate the best and second best results, respectively, based on unrounded values. N/A denotes GP memory limitations on Protein.

(a) Test RMSE (mean ± standard deviation).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
CO_2	1.62 ± 0.00	**1.60 ± 0.00**	2.45 ± 0.00	3.53 ± 0.00	7.36 ± 0.00	2.35 ± 0.00	8.46 ± 0.00	2.23 ± 0.00
Boston	4.01 ± 0.57	**3.64 ± 0.46**	5.11 ± 0.83	3.82 ± 0.39	6.69 ± 2.26	3.98 ± 0.64	3.68 ± 0.71	3.96 ± 0.67
Concrete	7.61 ± 1.07	**7.26 ± 0.93**	8.97 ± 1.28	8.13 ± 1.13	12.42 ± 2.40	10.46 ± 2.13	8.15 ± 1.54	7.36 ± 0.91
Kin8nm	**0.07 ± 0.00**	0.07 ± 0.00	0.11 ± 0.02	0.08 ± 0.00	0.12 ± 0.07	0.11 ± 0.00	0.08 ± 0.00	0.08 ± 0.00
Power	4.41 ± 0.30	**4.25 ± 0.18**	5.49 ± 0.87	4.89 ± 0.66	5.86 ± 0.70	4.43 ± 0.20	4.47 ± 0.22	4.50 ± 0.32
Protein	4.98 ± 0.31	**4.94 ± 0.31**	5.12 ± 0.30	N/A	5.35 ± 0.37	5.19 ± 0.35	5.55 ± 0.48	5.24 ± 0.45
Wine	0.70 ± 0.03	0.69 ± 0.03	0.65 ± 0.03	**0.63 ± 0.03**	0.90 ± 0.05	0.67 ± 0.04	0.64 ± 0.04	0.84 ± 0.03
Yacht	1.93 ± 0.85	2.12 ± 0.75	28.19 ± 18.70	9.78 ± 5.67	7.82 ± 7.56	12.27 ± 6.39	2.08 ± 1.40	**1.41 ± 0.45**

(b) Test NLL (mean ± standard deviation).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
CO_2	4.15 ± 0.00	3.56 ± 0.00	3.88 ± 0.00	3.81 ± 0.00	3.82 ± 0.00	4.10 ± 0.00	3.77 ± 0.00	**2.83 ± 0.00**
Boston	3.38 ± 0.42	3.01 ± 0.34	3.39 ± 0.06	**2.69 ± 0.12**	33.95 ± 23.13	3.29 ± 0.05	4.56 ± 1.27	4.55 ± 1.26
Concrete	4.19 ± 0.56	**3.43 ± 0.15**	3.91 ± 0.05	3.45 ± 0.16	104.74 ± 39.62	3.94 ± 0.09	8.88 ± 3.46	4.43 ± 0.64
Kin8nm	-1.06 ± 0.16	**-1.30 ± 0.05**	-0.57 ± 0.17	-0.58 ± 0.05	0.56 ± 0.03	-0.32 ± 0.02	-0.90 ± 0.13	-0.77 ± 0.17
Power	2.93 ± 0.09	**2.87 ± 0.04**	3.22 ± 0.26	3.30 ± 0.10	21.23 ± 4.97	3.87 ± 0.06	2.96 ± 0.06	2.94 ± 0.09
Protein	3.12 ± 0.12	**2.98 ± 0.10**	3.08 ± 0.09	N/A	16.78 ± 2.15	3.09 ± 0.05	3.78 ± 0.33	3.31 ± 0.22
Wine	2.33 ± 0.43	1.60 ± 0.25	1.05 ± 0.03	**0.96 ± 0.04**	1.44 ± 0.09	1.06 ± 0.04	0.97 ± 0.06	6.04 ± 1.44
Yacht	2.49 ± 0.87	**1.45 ± 0.59**	3.81 ± 0.53	3.28 ± 0.90	78.26 ± 134.10	4.04 ± 0.30	9.04 ± 10.34	5.60 ± 3.14

Table 15: Wall clock runtime (minutes) on gap regression benchmarks (Setting 2).

Dataset	BDKN	DE	LD-BLL	GP	DKL	BLL	MFVI-BNN	VBLL
CO_2	0.41	0.41	0.10	0.05	0.46	0.03	0.12	0.96
Boston	3.90	3.90	7.37	1.12	6.47	2.34	29.32	7.57
Concrete	3.48	3.47	4.87	0.82	4.00	1.41	30.05	9.14
Kin8nm	21.95	21.95	12.94	8.85	4.69	1.15	32.50	79.67
Power	12.43	12.43	6.27	4.89	2.46	0.99	16.75	44.29
Protein	134.59	134.59	60.69	N/A	8.83	1.11	68.45	574.56
Wine	6.41	6.41	1.95	4.91	5.30	0.71	51.64	21.70
Yacht	0.99	0.99	8.37	1.00	3.45	2.49	28.29	2.16

Together, the gap experiments show that BDKN's competitive predictive performance extends to regions excluded from training. The overall trends are similar to those observed on the standard regression benchmarks.

A.11 EFFECT OF ENSEMBLE SIZE

Here, we compare the standard ensembles of five networks reported in Tables 1 and 10 with ensembles of 50 networks across all standard regression datasets under both experimental settings. Tables 16 and 18 report the corresponding test RMSE and NLL. At each ensemble size, BDKN and DE use exactly the same trained networks. BDKN generally benefits from the larger ensemble. The improvements vary across datasets and metrics, but the overall results support using more networks to improve BDKN's predictive performance. The ensemble members are trained jointly through vectorized GPU operations while retaining separate parameters and independent training objectives. This enables efficient parallel computation and limits the increase in wall-clock training time as the ensemble size grows.

Table 16: Effect of ensemble size on test performance (Setting 1). Values are mean ± standard deviation; lower is better. **Bold** indicates the best result and underlining the second best result in each row. Ties at the displayed precision share the same formatting.

(a) Test RMSE.

Dataset	BDKN ($H = 5$)	BDKN ($H = 50$)	DE ($H = 5$)	DE ($H = 50$)
Boston	3.03 ± 0.78	**2.84 ± 0.81**	3.60 ± 1.76	3.39 ± 1.24
Concrete	5.90 ± 0.48	**5.33 ± 0.46**	7.18 ± 3.46	6.06 ± 0.55
Energy	1.38 ± 0.33	**0.58 ± 0.07**	2.54 ± 1.11	1.89 ± 0.63
Kin8nm	0.15 ± 0.04	**0.11 ± 0.01**	0.22 ± 0.04	0.43 ± 0.38
Naval	**0.01 ± 0.00**	**0.01 ± 0.00**	3.17 ± 7.88	219.72 ± 329.23
Power	**4.79 ± 1.02**	4.97 ± 1.51	3424.99 ± 6918.44	1739.61 ± 1172.82
Protein	4.94 ± 0.09	**4.72 ± 0.05**	5.31 ± 0.25	110.63 ± 67.06
Sarcos	3.94 ± 0.18	**3.49 ± 0.05**	53.62 ± 157.00	54.09 ± 52.08
Song	**10.85 ± 0.00**	10.86 ± 0.00	10.86 ± 0.00	19306.14 ± 0.00
Wine	**0.64 ± 0.04**	**0.64 ± 0.04**	1.97 ± 3.03	0.85 ± 0.42
Yacht	0.82 ± 0.38	**0.51 ± 0.15**	1.68 ± 1.06	1.32 ± 0.48

(b) Test NLL.

Dataset	BDKN ($H = 5$)	BDKN ($H = 50$)	DE ($H = 5$)	DE ($H = 50$)
Boston	2.61 ± 0.42	**2.52 ± 0.41**	3.03 ± 0.79	4.08 ± 0.45
Concrete	3.22 ± 0.10	**3.12 ± 0.12**	3.67 ± 0.67	4.14 ± 0.34
Energy	1.73 ± 0.24	**0.87 ± 0.13**	2.79 ± 0.60	3.56 ± 0.42
Kin8nm	-0.48 ± 0.22	**-0.81 ± 0.07**	-0.02 ± 0.30	1.04 ± 1.28
Naval	**-2.80 ± 0.01**	-2.79 ± 0.30	-0.07 ± 2.83	5.39 ± 2.05
Power	**2.97 ± 0.17**	3.13 ± 0.68	6.89 ± 2.06	9.38 ± 1.08
Protein	3.00 ± 0.02	**2.95 ± 0.01**	3.12 ± 0.11	7.16 ± 0.48
Sarcos	2.80 ± 0.04	**2.68 ± 0.01**	4.02 ± 1.52	6.11 ± 1.51
Song	**3.49 ± 0.00**	3.80 ± 0.00	3.81 ± 0.00	12.82 ± 0.00
Wine	**0.98 ± 0.06**	**0.98 ± 0.08**	1.59 ± 1.08	2.11 ± 0.93
Yacht	1.23 ± 0.55	**0.64 ± 0.29**	2.10 ± 0.73	2.64 ± 0.43

Robustness to extreme ensemble members: The Song and Protein results with 50 networks in Setting 1 show the same pattern as the Power and Sarcos results discussed in Appendix A.7: a few poorly fitted networks produce extreme predictions that dominate DE's average with equal weights. BDKN reduces their influence through its learned aggregation weights. With 50 networks, BDKN achieves test RMSEs of 10.86 on Song and 4.72 on Protein, compared with 19306.14 and 110.63 for DE, respectively. On Song, three of the 50 networks produce extreme predictions. On Protein, 13 of the 250 networks trained across five splits have training RMSE above 100, with two or three such members in each split.

Table 17 summarizes the training and test errors of individual networks. The means increase sharply with 50 networks on both data partitions, while the medians remain relatively stable, indicating that extreme members drive the deterioration. For Song with 50 networks, the mean training and test RMSEs of individual networks are 32913.15 and 33378.94, respectively; for Protein, they are

132.99 and 141.75. Large training errors indicate that the affected members fit the training data poorly, so their large test errors cannot be attributed solely to poor generalization. On Song, just three poorly fitted members produce extreme predictions that drive DE's large error. BDKN assigns these members aggregation weights with magnitudes of approximately 10^{-7}–10^{-6}, compared with DE's fixed weight of $1/50 = 0.02$, limiting their contributions to the aggregate prediction.

Table 17: Training and test RMSE statistics for individual networks in the runs with a high learning rate. These are means and medians of RMSEs of individual networks, not RMSEs of the aggregated predictions. Song uses one split; Protein statistics summarize individual network errors across five splits (25 values for $H = 5$ and 250 for $H = 50$), separately for training and test data.

		Training RMSE		Test RMSE	
Dataset	Networks (H)	Mean	Median	Mean	Median
Song	5	10.97	10.97	10.88	10.88
Song	50	32913.15	10.97	33378.94	10.89
Protein	5	6.00	5.30	6.05	5.45
Protein	50	132.99	5.57	141.75	5.64

Under the lower learning rate of Setting 2, Table 18 shows that increasing the ensemble size to 50 generally lowers BDKN's test RMSE, with improvements in both RMSE and NLL on datasets such as Energy and Yacht. As in the main benchmark results, DE is more competitive under this training configuration, and the large errors observed in Setting 1 are absent.

Table 18: Effect of ensemble size on test performance (Setting 2). Values are mean ± standard deviation; lower is better. **Bold** indicates the best result and underlining the second best result in each row. Ties at the displayed precision share the same formatting.

(a) Test RMSE.

Dataset	BDKN ($H = 5$)	BDKN ($H = 50$)	DE ($H = 5$)	DE ($H = 50$)
Boston	3.01 ± 0.78	2.98 ± 0.83	2.99 ± 0.85	**2.97 ± 0.88**
Concrete	5.22 ± 0.58	**4.93 ± 0.55**	5.23 ± 0.59	5.11 ± 0.51
Energy	0.76 ± 0.21	**0.50 ± 0.06**	1.42 ± 0.29	1.44 ± 0.23
Kin8nm	0.07 ± 0.00	0.07 ± 0.00	0.07 ± 0.00	**0.06 ± 0.00**
Naval	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
Power	3.93 ± 0.16	**3.85 ± 0.17**	3.98 ± 0.16	3.93 ± 0.16
Protein	4.05 ± 0.02	**3.98 ± 0.01**	4.06 ± 0.03	4.02 ± 0.01
Sarcos	2.13 ± 0.03	**1.99 ± 0.02**	2.14 ± 0.03	2.08 ± 0.01
Song	8.71 ± 0.00	8.67 ± 0.00	8.69 ± 0.00	**8.64 ± 0.00**
Wine	0.65 ± 0.06	0.67 ± 0.06	0.63 ± 0.05	**0.62 ± 0.05**
Yacht	0.85 ± 0.38	**0.61 ± 0.19**	1.17 ± 0.44	1.14 ± 0.48

(b) Test NLL.

Dataset	BDKN ($H = 5$)	BDKN ($H = 50$)	DE ($H = 5$)	DE ($H = 50$)
Boston	2.81 ± 0.62	2.96 ± 0.74	2.52 ± 0.37	**2.45 ± 0.33**
Concrete	3.23 ± 0.22	3.22 ± 0.25	3.01 ± 0.18	**2.95 ± 0.13**
Energy	1.17 ± 0.30	**0.81 ± 0.21**	1.71 ± 1.74	1.52 ± 1.22
Kin8nm	-1.18 ± 0.07	-1.16 ± 0.07	-1.35 ± 0.03	**-1.40 ± 0.02**
Naval	-4.76 ± 0.03	**-5.89 ± 0.05**	-5.33 ± 0.18	-5.37 ± 0.05
Power	2.79 ± 0.05	**2.77 ± 0.05**	2.79 ± 0.04	**2.77 ± 0.03**
Protein	2.80 ± 0.01	2.79 ± 0.00	2.74 ± 0.23	**2.67 ± 0.13**
Sarcos	2.19 ± 0.02	2.12 ± 0.01	**1.83 ± 0.01**	**1.83 ± 0.00**
Song	**3.30 ± 0.00**	3.58 ± 0.00	3.33 ± 0.00	3.32 ± 0.00
Wine	1.64 ± 0.36	2.18 ± 0.44	1.11 ± 0.24	**0.89 ± 0.14**
Yacht	1.48 ± 0.84	1.02 ± 0.76	-0.20 ± 0.26	**-0.24 ± 0.22**

Runtime comparison: Tables 19 and 20 compare wall clock runtimes for ensembles of 5 and 50 networks on the same workstation. Runtimes with 50 networks remain comparable in scale to those with 5 networks, although the differences vary by dataset. BDKN and DE use exactly the same trained ensemble, and their runtimes with 50 networks differ by at most 0.01 minutes at the reported precision.

Table 19: Wall clock runtime in minutes across all splits (Setting 1). Values for five networks are reproduced from Table 8.

Dataset	BDKN ($H = 5$)	BDKN ($H = 50$)	DE ($H = 5$)	DE ($H = 50$)
Boston	0.76	0.56	0.66	0.55
Concrete	0.75	1.10	0.66	1.09
Energy	0.54	0.91	0.46	0.91
Kin8nm	6.08	6.82	5.99	6.81
Naval	7.30	12.13	7.21	12.12
Power	6.40	8.83	6.32	8.83
Protein	8.47	8.33	8.45	8.33
Sarcos	30.04	32.90	29.95	32.90
Song	17.51	18.64	17.50	18.64
Wine	2.08	1.25	1.94	1.24
Yacht	0.61	0.43	0.45	0.42

Table 20: Wall clock runtime in minutes across all splits (Setting 2). Values for five networks are reproduced from Table 11.

Dataset	BDKN ($H = 5$)	BDKN ($H = 50$)	DE ($H = 5$)	DE ($H = 50$)
Boston	4.07	8.05	4.07	8.04
Concrete	8.31	12.63	8.26	12.63
Energy	6.73	11.98	6.72	11.98
Kin8nm	68.56	99.82	68.54	99.81
Naval	91.10	146.81	91.09	146.80
Power	80.87	116.79	80.87	116.79
Protein	111.35	112.08	111.35	112.08
Sarcos	362.98	356.20	362.98	356.20
Song	229.29	239.79	229.28	239.79
Wine	12.18	13.81	12.17	13.81
Yacht	3.20	3.77	3.19	3.77

A.12 Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning by developing a scalable and analytically tractable Bayesian framework for uncertainty quantification in regression. The proposed Bayesian Deep Kernel Networks (BDKN) are intended to improve the reliability and interpretability of predictive uncertainty relative to commonly used baselines (e.g., GP, Bayesian last layer models, and DE), which can be beneficial for downstream decision making in settings sensitive to risk.

Potential positive societal impacts include enabling more trustworthy model deployment by reducing overconfident predictions, supporting better decisions informed by uncertainty in applications such as scientific modeling, engineering design, and other domains where regression is a core component. By providing a transparent posterior over contributions of ensemble members and predictive uncertainty available in closed form, this work may also facilitate diagnostic analyses that help practitioners detect brittle predictors and improve model accountability.

To the best of our knowledge, this work does not introduce direct negative societal impacts or concerns about misuse posing a high risk.