

# Towards Scalable Bayesian Optimization via Gradient-Informed Bayesian Neural Networks

Georgios Makrygiorgos<sup>\*,\*</sup> Joshua Hang Sai Ip<sup>\*,\*</sup>  
Ali Mesbah<sup>\*</sup>

*<sup>\*</sup> Department of Chemical and Biomolecular Engineering, University of  
California, Berkeley, CA 94720, USA.*

---

**Abstract:** Bayesian optimization (BO) is a widely used method for data-driven optimization that generally relies on zeroth-order data of objective function to construct probabilistic surrogate models. These surrogates guide the exploration-exploitation process toward finding global optimum. While Gaussian processes (GPs) are commonly employed as surrogates of the unknown objective function, recent studies have highlighted the potential of Bayesian neural networks (BNNs) as scalable and flexible alternatives. Moreover, incorporating gradient observations into GPs, when available, has been shown to improve BO performance. However, the use of gradients within BNN surrogates remains unexplored. By leveraging automatic differentiation, gradient information can be seamlessly integrated into BNN training, resulting in more informative surrogates for BO. We propose a gradient-informed loss function for BNN training, effectively augmenting function observations with local gradient information. The effectiveness of this approach is demonstrated on well-known benchmarks in terms of improved BNN predictions and faster BO convergence as the number of decision variables increases.

*Keywords:* Data-driven optimization; Bayesian optimization; Bayesian neural networks.

---

## 1. INTRODUCTION

Bayesian optimization (BO) (Moćkus, 1974) has been a powerhouse for optimization of functions with unknown or partially known structure in diverse decision-making applications (Frazier, 2018). BO tackles black-box optimization problems by sequentially solving easier optimization sub-problems. This is done by combining a probabilistic surrogate model, representing the objective as a function of decision inputs, and an acquisition function (AF) that, based on the current belief about the objective, suggests points in the design space to query the objective.

BO is traditionally a zeroth-order optimization method, exclusively relying on objective function observations to refine its belief of the objective function. As for surrogate modeling of the objective, Gaussian process (GP) regression is a popular choice (Williams and Rasmussen, 2006). Recent work has demonstrated the benefits of leveraging derivative information in BO via GPs; fully observed gradients can yield  $D + 1$  extra observations, where  $D$  is the input dimensionality, per system query at a given input. This provides useful local information around the observed points, leading to a more accurate posterior for objective function approximation (Wu et al., 2017b). Gradients have been shown to improve regret bounds in BO (Shekhar and Javidi, 2021), or used for combined global and local search in more complex, potentially multi-objective tasks (Ip et al., 2024). A standard approach of exploiting

gradients (or higher-order derivatives) in BO with GPs is by defining a joint kernel that utilizes both function and gradient observations (Eriksson et al., 2018; Wu et al., 2017a). Alternatively, to mitigate challenges due to noisy function and derivative observations, gradients can be directly modeled using a probabilistic model and utilized to enforce optimality conditions in the AF (Penubothula et al., 2021; Makrygiorgos et al., 2023a,b).

Nonetheless, GPs suffer from several limitations in this context, including slow inference times that scale cubically with the number of observations and input dimension  $D$ . In particular, the computational complexity of GP inference is  $\mathcal{O}((D+1)^3n^3)$  and  $\mathcal{O}((D+1)n^3)$  for joint and independent kernels, respectively (Wu et al., 2017b). This challenge has motivated probabilistic modeling using neural networks (NNs), while combining them with ideas from non-parametric kernel methods such as GPs. Deep kernel learning (Wilson et al., 2016) is one such approach that improves the inference time of standard GPs from  $\mathcal{O}(n^3)$  to  $\mathcal{O}(n)$  by utilizing a NN to obtain a feature representation that is passed to a GP layer, enabling probabilistic inference. Another class of models is neural processes that combines the flexibility of NNs with the probabilistic properties of stochastic processes (Garnelo et al., 2018). Using an encoder-decoder framework with latent variables, neural processes provide a distribution over functions, capturing uncertainty in the latent representation.

In this work, we focus on Bayesian neural networks (BNNs), a class of probabilistic NNs that treat network

---

<sup>\*</sup> These authors contributed equally to this work.

The work was supported by the U.S. National Science Foundation under grants 2112754 and 2130734.

parameters as random variables; hence, allowing for uncertainty quantification in predictions. Dropout (Gal and Ghahramani, 2016), which randomly inactivates weights during NN training, is one of the simplest mechanisms for uncertainty quantification of NNs. Another popular method relies on variational inference, which approximates the posterior over weights using the so-called Bayes by Backprop method by minimizing the divergence from the true posterior (Blundell et al., 2015; Bao et al., 2023). Unlike the aforementioned methods that typically lead to conservative and suboptimal uncertainty estimates, Hamiltonian Monte Carlo (HMC) is a sampling-based uncertainty quantification approach, where Hamiltonian dynamics are used to guide exploration of the posterior to propose new samples (Neal, 2012). However, HMC does not scale favorably for handling a large number of uncertainties. In (Chen et al., 2014), HMC is modified to address this limitation by using stochastic gradients, known as stochastic gradient HMC (SGHMC), and adding noise to maintain a consistent posterior distribution, making it practical for BNN training. Although SGHMC can be effective for probabilistic surrogate modeling in BO, custom design of the loss function is crucial for obtaining models that describe the general function shape in the  $D$ -dimensional input space. Similar to conditioning GPs on gradient observations that provide extra local information about the function, augmenting the loss function in BNN training with “physically-relevant” terms that can provide the same type of information is essential, especially in low-data problems where BO is usually used.

We propose a new approach, termed gradient-informed neural network BO (GINNBO), that employs BNNs to learn scalable surrogates using both function and gradient observations. BNNs are  $\mathcal{O}((D+1)n)$  in computational complexity, which makes them far more efficient than GPs, especially when gradients are involved in surrogate modeling. We introduce a weighted derivative-informed regularization term in the loss function, guiding the BNN training towards solutions that align the learned function with its gradient. This is achieved by incorporating a soft constraint negative log-likelihood (NLL) term based on gradient observations alongside the NLL term for function observations, enriching the loss function with local gradient information. As this only modifies the loss for BNN training, the proposed method is agnostic to the choice of AF. We demonstrate that incorporating gradients via independent surrogates is insufficient. Instead, we explicitly enforce the relationship between the function and its partial derivatives in the input search space via automatic differentiation. This is achieved via a joint gradient-informed loss that involves both the objective function and its partial derivatives.

## 2. PRELIMINARIES

### 2.1 Problem Statement

We aim to minimize an expensive objective function  $f(\mathbf{x})$

$$\mathbf{x}^* \in \arg \min_{\mathbf{x}} f(\mathbf{x}), \quad (1)$$

where  $\mathbf{x} \in \mathbb{R}^D$  are the decision variables and  $D$  denotes the dimensionality of  $\mathbf{x}$ . We assume noisy observations of  $f$  or  $\nabla f$  are available, i.e.,

$$(y, \nabla y)^\top = (f(\mathbf{x}), \nabla f(\mathbf{x}))^\top + \boldsymbol{\epsilon}, \quad (2)$$

where  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 I)$ .

### 2.2 Bayesian Neural Networks

BO uses a surrogate model of the objective  $f(\mathbf{x})$ . Here, we use BNNs due to favorable scalability, i.e.,  $\mathcal{O}((D+1)n)$ . BNNs extend traditional NNs by incorporating Bayesian inference, allowing the network to estimate the posterior distribution of its outputs with respect to its learnable parameters. We define a BNN surrogate of function  $\phi(\mathbf{x})$  as

$$p(\phi(\mathbf{x})|\mathbf{x}, \theta) = \mathcal{N}(\hat{\phi}(\mathbf{x}, \theta_\mu), \theta_{\sigma^2}), \quad (3)$$

where  $\hat{\phi}(\mathbf{x}, \theta_\mu) : \mathbb{R}^D \rightarrow \mathbb{R}$  is a parametric NN model evaluated for parameters  $\theta_\mu$  and  $\mathcal{N}$  denotes a normal distribution. As such, we denote the learnable parameters of a BNN model by  $\theta = [\theta_\mu, \theta_{\sigma^2}]^\top$ .

Given a training dataset  $\mathcal{D} = \{(\mathbf{x}_i, y_i, \nabla y_i)\}_{i=1}^N$  with  $N$  being the number of samples, a BNN learns the posterior distribution of parameters  $\theta$ , i.e.,  $p(\theta|\mathcal{D})$ , rather than just finding a maximum likelihood estimate of  $\theta$ . Uncertainty in BNN predictions can then be derived by marginalizing over this posterior as

$$p(\phi(\mathbf{x}), \mathcal{D}) = \int p(\phi(\mathbf{x})|\mathbf{x}, \theta)p(\theta|\mathcal{D}) d\theta. \quad (4)$$

Furthermore, given the posterior parameter distribution  $p(\theta|\mathcal{D})$ , we can approximate the output posterior  $p(\phi(\mathbf{x}), \mathcal{D})$  in (4) in a sample-based manner to predict the mean  $\mu(\cdot)$  and variance  $\sigma^2(\cdot)$  of a BNN model as

$$\mu(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M \phi(\mathbf{x}; \theta_m^i), \quad (5a)$$

$$\sigma^2(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M (\phi(\mathbf{x}; \theta^i) - \mu(\mathbf{x}))^2 + \theta_{\sigma^2}, \quad (5b)$$

where  $M$  denotes the number of samples  $\theta^i \sim p(\theta|\mathcal{D})$  drawn from the posterior of  $\theta$ .

### 2.3 Bayesian Optimization

In BO, the next input query is designed by optimizing an AF, which realizes a tradeoff between *exploration* and *exploitation* of the input design space. Exploration seeks to explore regions that lack information about  $f(\mathbf{x})$ , towards searching for potentially better input designs. On the other hand, exploitation attempts to find better input designs in the vicinity of previously located desirable ones. Here, we use two AFs: lower confidence bound (LCB) and log expected improvement (LogEI). LCB is a commonly used AF in BO defined as (Srinivas et al., 2009)

$$\text{LCB}(\mathbf{x}) = \mu(\mathbf{x}) - \beta\sigma(\mathbf{x}), \quad (6)$$

where  $\beta$  is a hyperparameter that balances the tradeoff between exploration and exploitation. LogEI is based on the popular expected improvement AF (Jones et al., 1998), where the logarithm yields approximately the same solutions while alleviating issues with numerical optimization due to vanishing gradients. LogEI is defined as (Ament et al., 2023)

$$\text{LogEI}(\mathbf{x}) = \log \left| h \left( \frac{\mu(\mathbf{x}) - y^*}{\sigma(\mathbf{x})} \right) \right| + \log |\sigma(\mathbf{x})|, \quad (7)$$

where  $y^*$  is the best  $y$  found thus far,  $h(\cdot) = \phi_G(\cdot) + (\cdot)\Phi_G(\cdot)$ , and  $\phi_G$  and  $\Phi_G$  denote the probability distribution function and cumulative distribution function of a normal distribution, respectively.

Putting this together, a BO algorithm iteratively performs surrogate model training, AF optimization, and objective function query until the experimental or computational budget for querying the system is exhausted.

### 3. GRADIENT-INFORMED BAYESIAN NEURAL NETWORKS FOR BAYESIAN OPTIMIZATION

We now present the gradient-informed loss used for training a BNN surrogate, along with the proposed gradient-informed neural network BO (GINNBO) algorithm.

#### 3.1 Gradient-Informed Loss Function

Here, we present a loss function that embeds gradient information in constructing a BNN surrogate. Automatic differentiation allows for efficient computation of gradients. The gradients of a BNN output  $\phi(\mathbf{x}; \theta^i)$  with respect to inputs  $\mathbf{x}$  can be calculated by propagating the gradients backwards through the BNN using the chain rule

$$\frac{\partial \phi(\mathbf{x}; \theta^i)}{\partial \mathbf{x}} = \frac{\partial \phi(\mathbf{x}; \theta^i)}{\partial z} \times \frac{\partial z}{\partial \mathbf{x}}, \quad (8)$$

where  $z$  denotes the activation functions within the network. Let  $\nabla_{\mathbf{x}} \phi(\mathbf{x}; \theta^i)$  denote the vector of all partial derivatives. Similarly to the sample-based inference of the mean and variance of the BNN output in (5), we can approximate the mean and variance of gradients as

$$\mu_{\nabla}(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M \nabla_{\mathbf{x}} \phi(\mathbf{x}; \theta^i), \quad (9a)$$

$$\sigma_{\nabla}^2(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M (\nabla_{\mathbf{x}} \phi(\mathbf{x}; \theta^i) - \mu_{\nabla}(\mathbf{x}))^2 + \theta_{\sigma^2}. \quad (9b)$$

We define the total loss for training a BNN surrogate in terms of two negative log-likelihood (NLL) loss functions. The first is the NLL for fitting the BNN output,  $\mathcal{L}_f$ , defined as

$$\mathcal{L}_f = \frac{1}{N} \sum_{i=1}^N \left( \frac{(y_i - \mu(\mathbf{x}_i))^2}{2\sigma^2(\mathbf{x}_i)} + \frac{1}{2} \log(2\pi\sigma^2(\mathbf{x}_i)) \right). \quad (10)$$

$\mathcal{L}_f$  evaluates the likelihood of the predicted mean  $\mu(\mathbf{x}_i)$  and variance  $\sigma^2(\mathbf{x}_i)$  for each data point  $\mathbf{x}_i$ , penalizing predictions that deviate from the observation  $y_i$  based on the assumption that the observation noise is Gaussian. As such,  $\mathcal{L}_f$  accounts for both the predicted values and the associated uncertainty of predictions.

The second NLL loss is defined in terms of gradient predictions. The gradient loss  $\mathcal{L}_{\nabla f}$  takes the form of

$$\mathcal{L}_{\nabla f} = \frac{1}{N} \sum_{i=1}^N \left( \frac{\|\nabla y_i - \mu_{\nabla}(\mathbf{x}_i)\|^2}{2\sigma_{\nabla}^2(\mathbf{x}_i)} + \frac{1}{2} \log(2\pi\sigma_{\nabla}^2(\mathbf{x}_i)) \right), \quad (11)$$

where an  $L_2$ -norm is used since gradients comprise a vector. The gradient loss  $\mathcal{L}_{\nabla f}$  can be viewed as a regularization term that applies the same NLL formulation as in (10) to the partial gradients of the predicted mean.

Specifically, (11) penalizes discrepancies between the true gradients  $\nabla y_i$  and the BNN predicted gradients with mean  $\mu_{\nabla}(\mathbf{x}_i)$  and variance  $\sigma_{\nabla}^2(\mathbf{x}_i)$ .

By leveraging  $\mathcal{L}_{\nabla f}$ , we ensure that the BNN surrogate would better align with the shape and slope of the target function, yielding a more accurate description of the unknown objective. Accordingly, we define the total loss function as

$$\mathcal{L} = \mathcal{L}_f + \lambda_{\nabla} \mathcal{L}_{\nabla f} + \mathcal{L}_{\text{prior}}, \quad (12)$$

where  $\lambda_{\nabla}$  is a hyperparameter and  $\mathcal{L}_{\text{prior}}$  is as a regularization term (e.g., Ridge regularization). The total loss (12) makes the surrogate more sensitive to directional information in the input space.  $\lambda_{\nabla}$  scales the importance of the NLL of gradient observations in relation to function observations.  $\lambda_{\nabla}$  should be selected such that the NLL losses  $\mathcal{L}_f$  and  $\mathcal{L}_{\nabla f}$  have the same order of magnitude, while it can be adjusted for noisy observations. In practice,  $\lambda_{\nabla}$  can be set empirically via cross-validation.

#### 3.2 Stochastic Gradient Descent-Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) is a Markov Chain MC approach that uses the loss as a *potential energy function*  $U(\theta)$ , framing the problem of learning  $\theta$  as a physical problem (Neal, 2012). To this end, HMC views the unknown parameters  $\theta$  as “position” and introduces auxiliary variables  $\mathbf{r}$ , corresponding to “momentum” of a dynamical system. Accordingly, HMC looks to establish the joint distribution of  $\theta$  and  $\mathbf{r}$ , i.e.,

$$p(\theta, \mathbf{r} \mid \mathcal{D}) \propto \exp \left( -U(\theta) - \frac{1}{2} \mathbf{r}^T \mathbf{M}^{-1} \mathbf{r} \right), \quad (13)$$

where  $\mathbf{M}$  is a mass matrix modulating system dynamics. By simulating this Hamiltonian dynamical system, HMC generates samples according to (13), and the posterior distribution of  $\theta$  is obtained by marginalizing over  $\mathbf{r}$ .

In stochastic gradient HMC (SGHMC), the HMC approach is adapted for large-scale datasets by incorporating mini-batch gradients to alleviate the need for full-batch gradient computations and, thus, allow for efficient sampling (Chen et al., 2014). The updates of position  $\theta$  and momentum  $\mathbf{r}$  discretize Hamiltonian dynamics, while adding friction, via a matrix  $\mathbf{C}$ , and noise terms to counterbalance the variance introduced by stochastic gradients. This ensures that the sampler remains stable and converges to the desired target distribution even in the presence of noisy updates. To improve robustness in the sampling process, Springenberg et al. (2016) propose several modifications to the standard SGHMC. These include adaptive choices for the mass matrix  $\mathbf{M}$ , friction term  $\mathbf{C}$ , and step size  $\delta$ , making the method more robust to varying scales of parameters and noise levels in gradient estimates. The modified mass matrix  $\hat{\mathbf{M}}$  is adapted based on an estimate of the gradient variance  $\hat{V}_{\theta}$ , obtained via an exponential moving average of the gradient’s element-wise variance during a “burn-in” phase. The friction term  $\mathbf{C}$  is selected as  $\mathbf{C} = c\mathbf{I}$ , where  $c$  is tuned relative to the step size  $\delta$  and the variance estimate  $\hat{V}_{\theta}$  to ensure that the system maintains stability under stochastic gradients. With these modifications, the update rules of the adaptive SGHMC approach used here become

---

**Algorithm 1** Gradient-Informed Neural Network Bayesian Optimization (GINNBO)

---

```

1: Initialize dataset by obtaining  $k$  initial observations
    $\mathcal{D} = \{(\mathbf{x}_i, y_i, \nabla y_i)\}_{i=1}^k$ 
2: Initialize hyperparameters for BNN  $\phi(x; \theta^i)$ ; number of
   layers  $N_l$ , number of nodes per layer  $N_{\text{nodes}}^l$ , activation
   function  $z$ , and gradient regularization weight  $\lambda_{\nabla}$ 
3: Initialize SGHMC parameters: total steps  $N_{\text{MC}}$ ,
   “Burn-in” steps  $N_{\text{burn}}$ , learning rate  $\eta$ , step size  $\delta$ , and
   sampling interval  $w$ 
4: while computational budget not exceeded do
5:   Train BNN:
6:   for SGHMC step  $s = 1$  to  $N_{\text{MC}}$  do
7:     for mini-batch  $(\mathbf{x}_{\text{batch}}, y_{\text{batch}}, \nabla y_{\text{batch}})$  in  $\mathcal{D}$  do
8:       Calculate  $\mathcal{L}_f$  and  $\mathcal{L}_{\nabla f}$  for batch
9:       Compute total loss  $\mathcal{L} = \mathcal{L}_f + \lambda_{\nabla} \mathcal{L}_{\nabla f} + \mathcal{L}_{\text{prior}}$ 
10:      Backpropagate  $\mathcal{L}$  with SGHMC
11:    end for
12:    if  $s > N_{\text{burn}}$  and  $(s - N_{\text{burn}}) \bmod w = 0$  then
13:      Store current BNN weights sample
14:    end if
15:  end for
16:  Optimize Acquisition Function (AF):
17:   $\mathbf{x}' \leftarrow \arg \min_{\mathbf{x}} \text{AF}(\mathbf{x})$ 
18:  New observation  $(y', \nabla y')^\top = (f(\mathbf{x}'), \nabla f(\mathbf{x}'))^\top + \epsilon$ 
19:  Update dataset  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}', y', \nabla y')\}$ 
20: end while

```

---

$$\theta \leftarrow \theta + \mathbf{v}, \quad (14a)$$

$$\begin{aligned} \mathbf{v} \leftarrow & \mathbf{v} - \delta^2 \hat{V}_\theta^{-1/2} \nabla \tilde{U}(\theta) - \delta \hat{V}_\theta^{-1/2} \mathbf{C} \mathbf{v} \\ & + \mathcal{N}\left(0, 2\delta^3 \hat{V}_\theta^{-1/2} \mathbf{C} \hat{V}_\theta^{-1/2} - \delta^4 \mathbf{I}\right), \end{aligned} \quad (14b)$$

where  $\mathbf{v} = \delta \hat{V}_\theta^{-1/2} \mathbf{r}$  is an auxiliary variable and  $\nabla \tilde{U}(\theta)$  denotes the stochastic gradient of the potential energy  $U(\theta)$  computed based on a mini-batch of data. Here,  $\mathbf{v}$  stabilizes the updates by adapting to the variance structure of the parameter space, whereas the noise term  $\mathcal{N}\left(0, 2\delta^3 \hat{V}_\theta^{-1/2} \mathbf{C} \hat{V}_\theta^{-1/2} - \delta^4 \mathbf{I}\right)$  is dynamically scaled to counterbalance the variance introduced by mini-batch gradients to ensure the sampling robustness. As such, these adaptive updates balance exploration and stability in training the BNN surrogate within each BO iteration by aligning the influence of each parameter’s noise with its gradient variance. The proposed GINNBO approach is summarized in Algorithm 1.

### 3.3 Illustrative Example of Effect of Gradient Information

To demonstrate the effectiveness of the gradient-informed total loss (12) on the predictive capability of a BNN surrogate, we consider a 1D function  $f(x)$ , taken from (Paulson et al., 2022). This function exhibits two local minima. In Fig. 1, we present predictions of three different BNN models: BNN trained with the proposed loss (12) that captures the function and its derivative jointly (Fig. 1a); BNN trained with only function observations (Fig. 1b); and BNN trained to predict both the function and its derivative, i.e.,  $\hat{\phi}(x, \theta_\mu) : \mathbb{R}^D \rightarrow \mathbb{R}^{D+1}$  (Fig. 1c). In the latter case, the loss is (12); however, the posterior of gradients in (9a) and (9b) are evaluated using the  $i^{\text{th}}$  BNN

output  $\phi_i(x, \theta_\mu)$ ,  $i = 1, \dots, D + 1$ . In Fig. 1b, when only function observations are used as training data, the BNN predicts a function with a parabolic shape, completely missing the existence of the two local minima. In this case, the gradient of the BNN output, computed via automatic differentiation, deviates significantly from the true derivative. In Fig. 1c, SGHMC converges to inference mean and variance that minimize the NLL loss independently for each output, without enforcing an explicit connection between them. Therefore, although both the function and derivative predictions interpolate between the training samples, they fail to capture the true shape of the function. More importantly, the function and gradient predictions do not match the real values.

In contrast, when derivative observations are incorporated into the total loss (12), while the gradients are computed via automatic differentiation and not as independent outputs as in Fig. 1c, the BNN surrogate captures the true function almost perfectly (Fig. 1a). This is because the BNN learns not only the function at the training samples, but also the directionality of the function. Notably, in all cases, SGHMC offers a reliable characterization of uncertainty, exhibiting very low uncertainty around observed samples and reasonable bounds in unexplored regions. This reliable uncertainty estimation may not be achieved with other methods, such as variational inference.

## 4. CASE STUDIES

We demonstrate the performance of the proposed GINNBO approach on three benchmark problems of increasing dimensions and complexity, namely the McCormick (2D) (Adorio and Diliman, 2005), Rosenbrock (4D) (Picheny et al., 2013), and Hartmann (6D) (Dixon, 1978) functions. The performance of GINNBO with the total loss (12) (with  $\lambda_{\nabla} = 1$ ) is compared with the baseline case without the gradient NLL loss, i.e.,  $\lambda_{\nabla} = 0$  in (12). The AFs used are LCB (6) and LogEI (7). Since the analytical expressions for the three benchmark problems are known, their exact derivatives can be computed. Here, we consider the case of noiseless gradient observations. In all cases, the initial BNN surrogate is learned using  $2D$  samples of  $\mathbf{x}$  selected randomly. For all BNN surrogates, we use  $N_l = 5$  hidden layers with  $N_{\text{nodes}}^l = 80$  nodes and **tanh** activation function. For the SGHMC algorithm, we use approximately  $N_{\text{MC}} = 6000$  steps with a burn-in phase of  $N_{\text{burn}} = 2000$ . In the implementation of the AF optimization, L-BFGS with restarts is used.

We define  $\mathcal{R}_t$  as the instantaneous regret at iteration  $t$ , i.e.,  $\mathcal{R}_t = f(\mathbf{x}^*) - \mathcal{F}(x_t)$ , where  $\mathcal{F}(x_t)$  is the cumulative best observation (incumbent best) at iteration  $t$ , as our performance metric. Fig. 2 shows the convergence plots of GINNBO and the baseline algorithm with  $\lambda_{\nabla} = 0$  for the three benchmark problems. All regret profiles are based on the mean  $\pm 1$  standard deviation of  $\mathcal{R}_t$  over 10 random seeds. Fig. 2 suggests that GINNBO of Algorithm 1 outperforms the baseline BO algorithm for both LCB and LogEI AFs across all three benchmarks. It is evident that GINNBO is agnostic to the choice of AF. In all cases, GINNBO with  $\lambda_{\nabla} = 1$  in the total loss function reaches much smaller values of the instantaneous regret  $\mathcal{R}_t$  at earlier iteration indices  $t$ . When gradient information

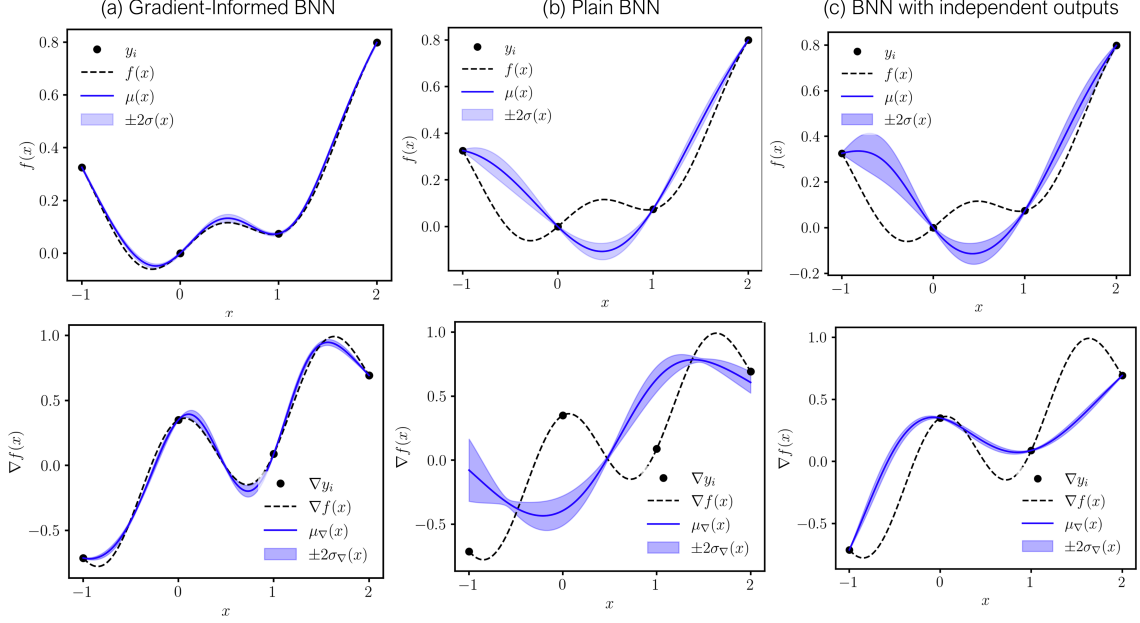


Fig. 1. Effect of gradient information on predictive capability of a BNN surrogate. (a) BNN trained with function and gradient observations using loss (12). (b) BNN trained with only function observations. (c) BNN trained with function and gradient observations, but treated as independent outputs of the BNN. The blue line represents the predicted mean, while the blue shaded region indicates the uncertainty ( $\pm 2$  standard deviations) around the prediction. In cases (a) and (b), derivative is computed via automatic differentiation of the BNN output.

is available, the BNN surrogate is more likely to locate regions that may contain the global minimum, but can also provide more reliable predictions of the unknown objective (as also seen in Fig. 1), in turn enabling less redundant exploration. The extent to which GINNBO outperforms the baseline algorithm varies with the problem dimensionality  $D$ . The improved performance is more pronounced in higher dimensions. Even in lower dimensions, it becomes more challenging to learn an accurate BNN surrogate in the limit of sample size. Gradient information allows for effectively scaling the available information per each training sample,  $D$ -fold; hence, enabling more accurate models to be trained. As can be seen, data scarcity in the case of no derivative observations can make the BNN-based search comparable (or worse) to that of random search.

Furthermore, we investigate the computational complexity of GINNBO with the LCB AF in relation to dimension  $D$ . Note that GINNBO is agnostic to AF used. As shown in Table 1, the computational times for training the BNN surrogate are nearly identical across all benchmark problem under identical SGHMC settings. This is expected since BNNs are scalable due to their NN architecture, which makes them an attractive option for higher-dimensional problems where traditional BO with GPs becomes compu-

tationally inefficient. Here, at each BO iteration, a BNN surrogate is re-trained from scratch based on the available dataset  $\mathcal{D}_t$ . It should be noted that transfer learning techniques can further accelerate the time needed to train each model.

## 5. CONCLUSION

This paper presented a gradient-informed Bayesian neural network BO (GINNBO) approach, as a scalable alternative for BO when gradient information is available. GINNBO uses a gradient-based negative log-likelihood term in the loss function, acting as a regularization term, to learn a more accurate model of the blackbox objective function in BO. Numerical experiments on several benchmark problems demonstrate that GINNBO consistently outperforms the case in which the BNN surrogate is only trained using function observations. We also demonstrate that the computational complexity of the BNN is independent of the dimensionality of the input space, making it an attractive alternative for gradient-informed BO compared to GP-based methods. Future work will investigate the effectiveness of GINNBO on real-life applications in comparison with alternative gradient-informed GP-based BO methods. We will also investigate the effect of BNN and BO hyperparameters on GINNBO performance.

## REFERENCES

- Adorio, E.P. and Diliman, U. (2005). MVF-multivariate test functions library in C for unconstrained global optimization. *Quezon City, Metro Manila, Philippines*.  
Ament, S., Daulton, S., Eriksson, D., Balandat, M., and Bakshy, E. (2023). Unexpected improvements to expected improvement for Bayesian optimization. *Ad-*

Benchmark problem	$D$	Mean $\pm$ Std. Dev. (s)
Mccormick	2	23.09 $\pm$ 1.26
Rosenbrock	4	23.36 $\pm$ 0.52
Hartmann	6	23.71 $\pm$ 0.29

Table 1. Computational times for training a BNN surrogate with the gradient-informed loss (12) for the McCormick (2D), Rosenbrock (4D), and Hartmann (6D) problems.

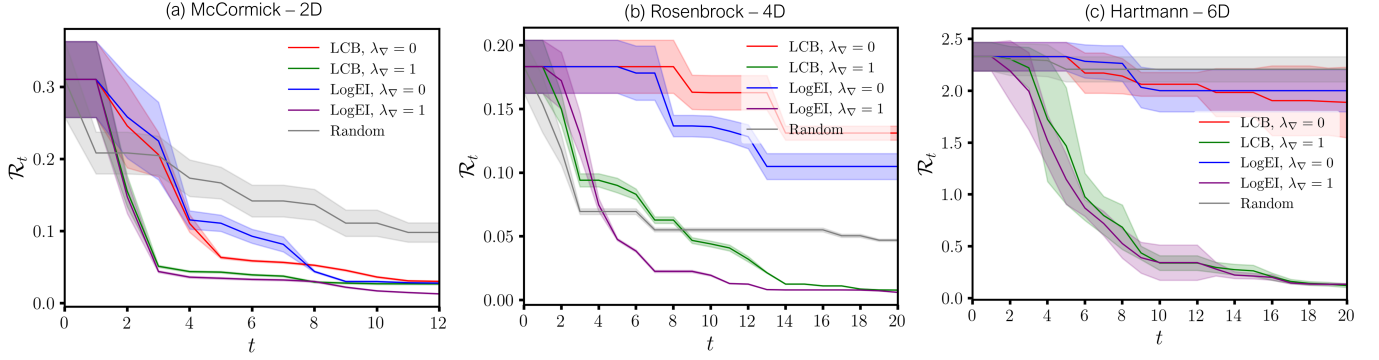


Fig. 2. Instantaneous regret plots for (a) McCormick (2D), (b) Rosenbrock (4D), and (c) Hartmann (6D) benchmark problems. Comparisons are demonstrated for the base case (with no gradient NLL term in the loss function (12),  $\lambda_{\nabla} = 0$ ) and the case of the gradient NLL term with  $\lambda_{\nabla} = 1$  for the LCB and LogEI acquisition functions. Each experiment is performed with 10 random seeds and the mean  $\pm 1$  standard deviation are displayed.

- vances in Neural Information Processing Systems*, 36, 20577–20612.
- Bao, Y., Chan, K.J., Mesbah, A., and Velni, J.M. (2023). Learning-based adaptive-scenario-tree model predictive control with improved probabilistic safety using robust Bayesian neural networks. *International Journal of Robust and Nonlinear Control*, 33(5), 3312–3333.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *Proceedings of the International Conference on Machine Learning*, 1613–1622. Lille, France.
- Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the International Conference on Machine Learning*, volume 32, 1683–1691. Beijing, China.
- Dixon, L.C.W. (1978). The global optimization problem: an introduction. *Towards Global Optimization* 2, 1–15.
- Eriksson, D., Dong, K., Lee, E., Bindel, D., and Wilson, A.G. (2018). Scaling Gaussian process regression with derivatives. *Advances in Neural Information Processing Systems*, 31.
- Frazier, P.I. (2018). A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.
- Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the International Conference on Machine Learning*, volume 48, 1050–1059.
- Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D.J., Eslami, S., and Teh, Y.W. (2018). Neural processes. *arXiv preprint arXiv:1807.01622*.
- Ip, J.H.S., Chakrabarty, A., Mesbah, A., and Romeres, D. (2024). Preference-based multi-objective Bayesian optimization with gradients. In *NeurIPS Workshop on Bayesian Decision-making and Uncertainty*.
- Jones, D.R., Schonlau, M., and Welch, W.J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13, 455–492.
- Makrygiorgos, G., Paulson, J.A., and Mesbah, A. (2023a). Gradient-enhanced Bayesian optimization via acquisition ensembles with application to reinforcement learning. *IFAC-PapersOnLine*, 56(2), 638–643.
- Makrygiorgos, G., Paulson, J.A., and Mesbah, A. (2023b). No-regret Bayesian optimization with gradients using local optimality-based constraints: Application to closed-loop policy search. In *Proceedings of the 62<sup>nd</sup> IEEE Conference on Decision and Control*, 20–25.
- Moćkus, J. (1974). On Bayesian methods for seeking the extremum. In *Proceedings of the Optimization Techniques IFIP Technical Conference*, 400–404. Novosibirsk.
- Neal, R.M. (2012). MCMC using Hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*.
- Paulson, J.A., Makrygiorgos, G., and Mesbah, A. (2022). Adversarially robust Bayesian optimization for efficient auto-tuning of generic control structures under uncertainty. *AIChE Journal*, 68(6), e17591.
- Penubothula, S., Kamanchi, C., Bhatnagar, S., et al. (2021). Novel first order Bayesian optimization with an application to reinforcement learning. *Applied Intelligence*, 51(3), 1565–1579.
- Picheny, V., Wagner, T., and Ginsbourger, D. (2013). A benchmark of Kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48, 607–626.
- Shekhar, S. and Javidi, T. (2021). Significance of gradient information in Bayesian optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 130, 2836–2844.
- Springenberg, J.T., Klein, A., Falkner, S., and Hutter, F. (2016). Bayesian optimization with robust Bayesian neural networks. *Advances in Neural Information Processing Systems*, 29.
- Srinivas, N., Krause, A., Kakade, S.M., and Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.
- Williams, C.K. and Rasmussen, C.E. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge.
- Wilson, A.G., Hu, Z., Salakhutdinov, R., and Xing, E.P. (2016). Deep kernel learning. In *Artificial Intelligence and Statistics*, volume 51, 370–378.
- Wu, A., Aoi, M.C., and Pillow, J.W. (2017a). Exploiting gradients and Hessians in Bayesian optimization and Bayesian quadrature. *arXiv preprint arXiv:1704.00060*.
- Wu, J., Poloczek, M., Wilson, A.G., and Frazier, P. (2017b). Bayesian optimization with gradients. *Advances in Neural Information Processing Systems*, 30.