# Bayesian Kernelized Tensor Factorization as Surrogate for Bayesian Optimization

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Bayesian optimization (BO) mainly uses Gaussian processes (GP) with a stationary and separable kernel function (e.g., the squared-exponential kernel with automatic relevance determination [SE-ARD]) as the surrogate model. However, such localized kernel specifications are deficient in learning complex functions that are non-stationary, non-separable and multi-modal. In this paper, we propose using Bayesian Kernelized Tensor Factorization (BKTF) as a new surrogate model for Bayesian optimization (BO) in a $D$-dimensional grid with both continuous and categorical variables. Our key idea is to approximate the underlying $D$-dimensional solid with a fully Bayesian low-rank tensor CP decomposition, in which we place GP priors on the latent basis functions for each dimension to encode local consistency and smoothness. With this formulation, the information from each sample can be shared not only with neighbors but also across dimensions, thus fostering a more global search strategy. Although BKTF no longer has an analytical posterior, we efficiently approximate the posterior distribution through Markov chain Monte Carlo (MCMC). We conduct numerical experiments on several test functions with continuous variables and two machine learning hyperparameter tuning problems with mixed variables. The results show that BKTF offers a flexible and highly effective approach to characterizing and optimizing complex functions, especially in cases where the initial sample size and budget are severely limited.

## 1 Introduction

For many applications in science and engineering, such as emulation-based studies, experiment design, and automated machine learning, the goal is to optimize a complex black-box function $f(\boldsymbol{x})$ in a $D$-dimensional space, for which we have limited prior knowledge. The main challenge in such optimization problems is that we aim to efficiently find the global optima, while the objective function $f$ is often gradient-free, multimodal and computationally expensive to evaluate. Bayesian optimization (BO) offers a powerful statistical approach to these problems, particularly when the observation budgets are limited [1, 2, 3]. A typical BO framework consists of two components—a surrogate model and an acquisition function (AF)—to balance exploitation and exploration. The surrogate is a probabilistic model that allows us to estimate $f(\boldsymbol{x})$ with uncertainty at a new location $\boldsymbol{x}$, and the AF is used to determine which location to query next.

Gaussian process (GP) regression is the most widely used surrogate for BO [3, 4], thanks to its appealing properties in providing analytical derivations and uncertainty quantification (UQ). The choice of kernel/covariance function is a critical decision in GP models; for multidimensional BO problems, perhaps the most popular kernel is the ARD (automatic relevance determination)—Squared-Exponential (SE) or Matérn kernel [4]. Although this specification has certain numerical advantages and can help automatically learn the importance of input variables, a key limitation is

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

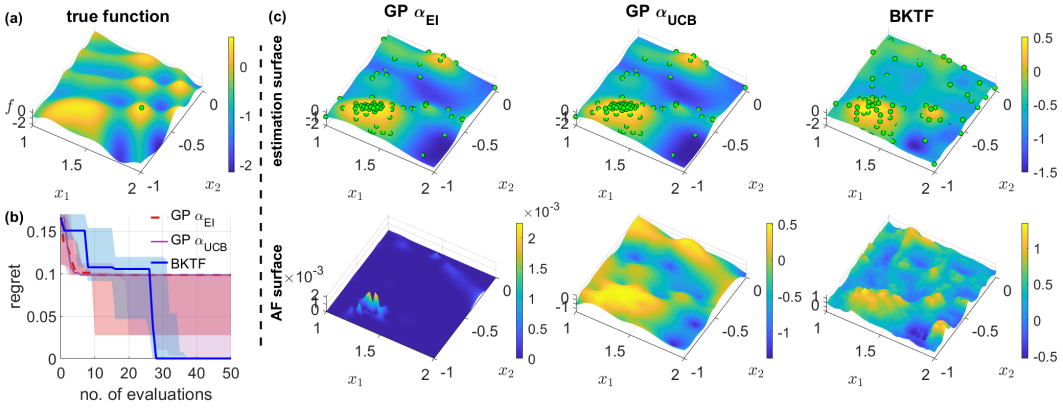



Figure 1: BO for a 2D function: (a) True function surface, where the global maximum is marked; (b) Comparison between BO models using GP surrogates (with two AFs) and BKTF with 30 random initial observations, averaged over 20 replications; (c) Specific results of one run, including the final estimated mean surface for $f$, in which green dots denote the locations of the selected candidates, and the corresponding AF surface.

that it implies/assumes that the underlying stochastic process is stationary and separable, and the value of the covariance function between two random points quickly goes to zero with the increase of input dimensionality. These assumptions can be problematic for complex real-world processes with long-range dependencies, because estimating the underlying function with a simple ARD kernel would require a large number of observations. A potential solution to address this issue is to use more flexible kernel structures. The additive kernel, for example, is designed to characterize a more "global" structure by restricting variable interactions [5]. However, in practice using additive kernels requires strong prior knowledge to determine the proper interactions and involves a large number of kernel hyperparameters to learn [6]. Another emerging solution is to use deep GP [7], such as in [8, 9]; however, learning deep GP often becomes a more challenging task due to the inference of latent layers. In addition, these GP related surrogates can be more deficient to tune when taking into account both continuous and categorical inputs.

In this paper, we propose using *Bayesian Kernelized Tensor Factorization* (BKTF) as a flexible and adaptive surrogate model for BO in a $D$-dimensional Cartesian product space (i.e., grid) when $D$ is relatively small (say $D \leq 10$). BKTF is initially developed for modeling multidimensional spatiotemporal data with UQ, for tasks such as spatiotemporal kriging/cokriging [10, 11]. This paper adapts BKTF to the BO setting, and our key idea is to characterize the multivariate objective function $f(\boldsymbol{x}) = f(x_1, \ldots, x_D)$ for a specific BO problem using the low-rank tensor CANDECOMP/PARAFAC (CP) factorization with random basis functions. Unlike other basis-function models that rely on known/deterministic basis functions [12], BKTF uses a hierarchical Bayesian framework to achieve high-quality UQ in a more flexible way—GP priors are used to model the basis functions, and hyperpriors are used to model kernel hyperparameters in particular for the lengthscale that characterizes the scale of variation. In addition, BKTF also provides a natural solution for categorical variables, for which we can simply introduce an inverse-Wishart prior on the covariance matrix of the basis functions.

Figure 1 shows the comparison between BKTF and GP surrogates when optimizing a bivariate function ($D = 2$) that is nonstationary, nonseparable, and multimodal. The details of this function and the BO experiments are provided in Appendix C. This $2D$ case clearly shows that GP surrogate is limited by the local kernel and becomes ineffective in finding the global solution, while BKTF offers superior flexibility and adaptability to characterize the multidimensional process from limited data. Unlike GP-based surrogate models, BKTF no longer has an analytical posterior; however, efficient inference and acquisition can be achieved through Markov chain Monte Carlo (MCMC) in an element-wise learning way, in which we update basis functions and kernel hyperparameters using Gibbs sampling and slice sampling, respectively [11]. For optimization, we first use MCMC samples to approximate the posterior distribution of the entire tensor and then naturally define the upper confidence bound (UCB) of the posterior as AF. This process is feasible for many real-world applications that can be studied in a discretized tensor product space, such as experimental design.

We conduct extensive experiments on both standard optimization and ML hyperparameter tuning tasks. Our results show that BKTF achieves a fast global search for optimizing complex objective functions with limited initial data and budget.

## 2 Bayesian optimization

Let $f : \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_D \to \mathbb{R}$ be a black-box function that could be nonconvex, derivative-free, and expensive to evaluate. BO aims to address the global optimization problem:

$$\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}). \tag{1}$$

BO solves this problem by first building a probabilistic model for $f(\boldsymbol{x})$ (i.e., surrogate model) based on initial observations and then using the predictive distribution to decide where in $\mathcal{X}$ to evaluate/query next. The overall goal of BO is to find the global optimum of the objective function using as few evaluations as possible. Most BO models rely on a GP prior for $f(\boldsymbol{x})$ to achieve prediction and UQ:

$$f(\boldsymbol{x}) = f(x_1, \ldots, x_D) \sim \mathcal{GP}\left(m\left(\boldsymbol{x}\right), k\left(\boldsymbol{x}, \boldsymbol{x}'\right)\right), \text{ with } x_d \in \mathcal{X}_d,\ d = 1, \ldots, D, \tag{2}$$

where $k\left(\cdot,\cdot\right)$ is a valid kernel/covariance function and $m\left(\cdot\right)$ is a mean function that can be generally assumed to be 0. Given a finite set of observation points $\{\boldsymbol{x}_i\}_{i=1}^n$ with $\boldsymbol{x}_i = \left(x_1^i, \ldots, x_D^i\right)^\top$, the vector of function values $\boldsymbol{f} = \left(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)\right)^\top$ has a multivariate Gaussian distribution $\boldsymbol{f} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{K}\right)$, where $\boldsymbol{K}$ is the $n \times n$ covariance matrix. For a set of observed data $\mathcal{D}_n = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ with i.i.d. Gaussian noise, i.e., $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \tau^{-1})$, GP gives an analytical posterior distribution of $f(\boldsymbol{x})$ at an unobserved point $\boldsymbol{x}^*$:

$$f(\boldsymbol{x}^*) \mid\ \mathcal{D}_n \sim \mathcal{N}\Big(\boldsymbol{k}_{\boldsymbol{x}^*\boldsymbol{X}}\left(\boldsymbol{K} + \tau^{-1}\boldsymbol{I}_n\right)^{-1}\boldsymbol{y},\ k(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}_{\boldsymbol{x}^*\boldsymbol{X}}\left(\boldsymbol{K} + \tau^{-1}\boldsymbol{I}_n\right)^{-1}\boldsymbol{k}_{\boldsymbol{x}^*\boldsymbol{X}}^\top\Big), \tag{3}$$

where $\boldsymbol{k}_{\boldsymbol{x}^*\boldsymbol{X}} = [k(\boldsymbol{x}^*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}^*, \boldsymbol{x}_n)]^\top$ and $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$.

Based on the posterior distributions of $f$, one can compute an AF, denoted by $\alpha : \mathcal{X} \to \mathbb{R}$, for a new candidate $\boldsymbol{x}^*$ and evaluate how promising $\boldsymbol{x}^*$ is. In BO, the next query point is often determined by maximizing a selected/predefined AF, i.e., $\boldsymbol{x}_{n+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha\left(\boldsymbol{x} \mid \mathcal{D}_n\right)$. Most AFs are based on the predictive mean and variance; for example, a commonly used AF is the **expected improvement** (EI) [1]:

$$\alpha_{\text{EI}}\left(\boldsymbol{x} \mid \mathcal{D}_n\right) = \sigma(\boldsymbol{x})\varphi\left(\frac{\Delta(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right) + |\Delta(\boldsymbol{x})|\,\Phi\left(\frac{\Delta(\boldsymbol{x})}{\sigma(\boldsymbol{x})}\right), \tag{4}$$

**Algorithm 1:** Basic BO process.

**Input:** Initial dataset $\mathcal{D}_0$ and a trained surrogate model; total budget $N$.

**for** $n = 1, \ldots, N$ **do**

- Approximate the posterior distribution of $f$ using the surrogate model based on $\mathcal{D}_{n-1}$;
- Find next evaluation point $\boldsymbol{x}_n$ by optimizing the AF;
- Augment data $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\boldsymbol{x}_n, y_n\}$, update the surrogate model.

where $\Delta(\boldsymbol{x}) = \mu(\boldsymbol{x}) - f_n^\star$ is the expected difference between the proposed point $\boldsymbol{x}$ and the current best solution, $f_n^\star = \max_{\boldsymbol{x}\in\{\boldsymbol{x}_i\}_{i=1}^n} f(\boldsymbol{x})$ denotes the best function value obtained so far; $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are the predictive mean and predictive standard deviation at $\boldsymbol{x}$, respectively; and $\varphi(\cdot)$ and $\Phi(\cdot)$ denote the probability density function (PDF) and the cumulative distribution function (CDF) of standard normal, respectively. Another widely applied AF for maximization problems is the **upper confidence bound** (UCB) [13]:

$$\alpha_{\text{UCB}}\left(\boldsymbol{x} \mid \mathcal{D}_n, \beta\right) = \mu(\boldsymbol{x}) + \beta\sigma(\boldsymbol{x}), \tag{5}$$

where $\beta$ is a tunable parameter that balances exploration and exploitation. The general BO procedure can be summarized as Algorithm 1.

## 3 BKTF for Bayesian optimization

### 3.1 Bayesian hierarchical model specification

Before introducing BKTF, we first construct a $D$-dimensional grid space corresponding to the search space $\mathcal{X}$, where $\mathcal{X}_d$ could be continuous, integer-valued, or categorical. We define it on $D$ sets $\{S_1, \ldots, S_D\}$ and denote the whole grid by $\prod_{d=1}^D S_d$: $S_1 \times \ldots \times S_D =$

$\{(s_1, \ldots, s_D) \mid \forall d \in \{1, \ldots, D\}, s_d \in S_d\}$. For dimensions with integer-valued and categorical input, we consider $S_d$ the set of corresponding discrete values. For dimensions with continuous input, the coordinate set $S_d$ is formed by $m_d$ interpolation points $c_i^d$ that are distributed over the bounded interval $\mathcal{X}_d = [a_d, b_d]$, i.e., $S_d = \left\{c_i^d\right\}_{i=1}^{m_d}$ with $c_i^d \in \mathcal{X}_d$. The size of $S_d$ becomes $|S_d| = m_d$, and size of the entire grid space is $\prod_{d=1}^{D} |S_d|$. Note that we do not restrict $S_d$ to be uniformly distributed.

We assume the underlying function $f$ as a stochastic process that is zero-centered. We randomly sample an initial dataset including $n_0$ input-output data pairs $\mathcal{D}_0 = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n_0}$, where $\{\boldsymbol{x}_i\}_{i=1}^{n_0}$ are located in $\prod_{d=1}^{D} S_d$, and this yields an incomplete $D$-dimensional tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{|S_1| \times \cdots \times |S_D|}$ with $n_0$ observed points. BKTF approximates the entire data tensor $\boldsymbol{\mathcal{Y}}$ by a kernelized tensor CP decomposition:

$$\boldsymbol{\mathcal{Y}} = \sum_{r=1}^{R} \lambda_r \cdot \boldsymbol{g}_1^r \circ \boldsymbol{g}_2^r \circ \cdots \circ \boldsymbol{g}_D^r + \boldsymbol{\mathcal{E}}, \tag{6}$$

where $R$ is a pre-specified tensor CP rank, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_R)^\top$ denote weight coefficients that capture the magnitude/importance of each rank in the factorization, $\boldsymbol{g}_d^r = [g_d^r(s_d) : s_d \in S_d] \in \mathbb{R}^{|S_d|}$ denotes the $r$th latent factor for the $d$th dimension, entries in $\boldsymbol{\mathcal{E}}$ are i.i.d. white noises following $\mathcal{N}(0, \tau^{-1})$. It should be particularly noted that both the coefficients $\{\lambda_r\}_{r=1}^{R}$ and the latent basis functions $\{g_1^r, \ldots, g_D^r\}_{r=1}^{R}$ are random variables. The function approximation for $\boldsymbol{x} = (x_1, \ldots, x_D)^\top$ is:

$$f(\boldsymbol{x}) = \sum_{r=1}^{R} \lambda_r g_1^r(x_1) \cdots g_D^r(x_D) = \sum_{r=1}^{R} \lambda_r \prod_{d=1}^{D} g_d^r(x_d). \tag{7}$$

For priors, we assume $\lambda_r \sim \mathcal{N}(0, 1)$ for $r = 1, \ldots, R$ and use a GP prior on the latent factors for dimension $d$ with continuous input:

$$g_d^r(x_d) \mid l_d^r \sim \mathcal{GP}\left(0, k_d^r(x_d, x_d'; l_d^r)\right), \text{ for } r = 1, \ldots, R, \tag{8}$$

where $k_d^r$ is a valid kernel function. In this paper, we choose a Matérn 3/2 kernel $k_d^r(x_d, x_d'; l_d^r) = \sigma^2 \left(1 + \frac{\sqrt{3}|x_d - x_d'|}{l_d^r}\right) \exp\left(-\frac{\sqrt{3}|x_d - x_d'|}{l_d^r}\right)$. We fix the kernel variance of $k_d^r$ as $\sigma^2 = 1$, and only learn the lengthscale hyperparameters $l_d^r$, since the variances of the model can be captured by $\boldsymbol{\lambda}$. One can also exclude $\boldsymbol{\lambda}$ but introduce variance $\sigma^2$ as a kernel hyperparameter on one of the basis functions; however, learning kernel hyperparameters is computationally more expensive than learning $\boldsymbol{\lambda}$. For simplicity, we can also assume the lengthscale parameters to be identical, i.e., $l_d^1 = \ldots = l_d^R = l_d$, for each dimension $d$. The prior distribution for the corresponding latent factor $\boldsymbol{g}_d^r$ is then a Gaussian distribution: $\boldsymbol{g}_d^r \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_d^r)$, where $\boldsymbol{K}_d^r$ is the $|S_d| \times |S_d|$ covariance matrix computed from $k_d^r$. We place Gaussian hyperpriors on the log-transformed kernel hyperparameters to ensure positive values, i.e., $\log(l_d^r) \sim \mathcal{N}\left(\mu_l, \tau_l^{-1}\right)$. For categorical input, we assume that the corresponding latent basis functions $\boldsymbol{g}_d^r \mid \boldsymbol{\Lambda}_d \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}_d^{-1}\right)$ for $r = 1, \ldots, R$, where the precision matrix $\boldsymbol{\Lambda}_d$ follows a Wishart prior with an identity scale matrix and $|S_d|$ degrees of freedom, i.e., $\boldsymbol{\Lambda}_d \sim \mathcal{W}\left(\boldsymbol{I}_{|S_d|}, |S_d|\right)$. For noise precision $\tau$, we assume a conjugate Gamma prior $\tau \sim \text{Gamma}(a_0, b_0)$. For dimensions with integer variables, we could model the covariance of the basis functions either using a kernel matrix or with an inverse-Wishart prior, depending on specific situations.

For observations, we assume each $y_i$ in the initial dataset $\mathcal{D}_0$ follows a Gaussian distribution:

$$y_i \mid \{g_d^r(x_d^i)\}, \{\lambda_r\}, \tau \sim \mathcal{N}\left(f(\boldsymbol{x}_i), \tau^{-1}\right). \tag{9}$$

### 3.2 BKTF as a two-layer deep GP

Here we show the representation of BKTF as a two-layer deep GP. The **first** layer characterizes the generation of latent functions $\{g_d^r\}_{r=1}^{R}$ for dimension $d$. For the **second** layer, if we consider $\{g_1^r, \ldots, g_D^r\}_{r=1}^{R}$ as parameters and rewrite the functional decomposition in Eq. (7) as a linear function $f(\boldsymbol{x}; \{\lambda_r\}) = \sum_{r=1}^{R} \lambda_r \prod_{d=1}^{D} g_d^r(x_d)$ with $\lambda_r \overset{\text{iid}}{\sim} \mathcal{N}(0, 1)$, we can marginalize $\{\lambda_r\}$ and obtain a fully symmetric multilinear kernel/covariance function for any two data points $\boldsymbol{x} = (x_1, \ldots, x_D)^\top$ and $\boldsymbol{x}' = (x_1', \ldots, x_D')^\top$:

$$k\left(\boldsymbol{x}, \boldsymbol{x}'; \{g_1^r, \ldots, g_D^r\}_{r=1}^{R}\right) = \sum_{r=1}^{R} \prod_{d=1}^{D} g_d^r(x_d) g_d^r(x_d'). \tag{10}$$

As can be seen, the second layer has a multilinear product kernel function parameterized by $\{g_1^r, \ldots, g_D^r\}_{r=1}^R$. There are some properties to highlight: (i) the kernel is **nonstationary** since the value of $g_d^r(\cdot)$ is location specific, and (ii) the kernel is **nonseparable** when $R > 1$. Therefore, this specification is very different from traditional GP surrogates, such as:

$$\begin{cases} \text{GP ARD:} & k(\boldsymbol{x}, \boldsymbol{x}') = \prod_{d=1}^D k_d(x_d, x_d'), \\ & \text{kernel function is stationary and separable} \\ \text{Additive GP (2nd order):} & k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^D k_d^1(x_d, x_d') + \sum_{d=1}^{D-1} \sum_{e=d+1}^D k_d^2(x_d, x_d') k_e^2(x_e, x_e'), \\ & \text{kerenl is stationary and nonseparable} \end{cases}$$

where all kernel functions are stationary with different hyperparameters (e.g., length scale and variance). Compared to the GP-based kernel specification, the multilinear kernel in Eq. (10) has a much larger set of hyperparameters and becomes more flexible and adaptive for the data. From a GP perspective, learning the hyperparameter in the kernel function in Eq. (10) will be computationally expensive; however, we can achieve efficient Bayesian inference of $\{\lambda_r, g_1^r, \ldots, g_D^r\}_{r=1}^R$ under a kernelized tensor factorization framework.

### 3.3 Model inference

Unlike GP, BKTF no longer enjoys an analytical posterior distribution. Based on the aforementioned prior and hyperprior settings, we adapt the MCMC updating procedure in Ref. [10, 11] to an efficient Gibbs sampling algorithm for model inference. This allows us to accommodate observations that are not located in the grid space $\prod_{d=1}^D S_d$. The detailed derivation of the sampling algorithm is given in Appendix A. In terms of computational cost, we note that updating $\boldsymbol{g}_d^r$ and kernel hyperparameters requires $\min\left\{\mathcal{O}(n^3), \mathcal{O}(|S_d|^3)\right\}$ in time. Sparse GP (such as [14]) could be introduced when $n, |S_d|$ become large. See Appendix A, F for detailed discussion/comparison about computation complexity.

### 3.4 Prediction and AF computation

In each step of function evaluation, we run the MCMC sampling process $K$ iterations for model inference, where the first $K_0$ samples are taken as burn-in and the last $K - K_0$ samples are used for posterior approximation. The predictive distribution for any entry $f^*$ in the defined grid space conditioned on the observed dataset $\mathcal{D}_n$ can be obtained by the Monte Carlo approximation $p(f^* \mid \mathcal{D}_n, \boldsymbol{\theta}_0) \approx \frac{1}{K-K_0} \times \sum_{k=K_0+1}^K p\left(f^* \,\middle|\, (\boldsymbol{g}_d^r)^{(k)}, \boldsymbol{\lambda}^{(k)}, \tau^{(k)}\right)$, where $\boldsymbol{\theta}_0 = \{\mu_l, \tau_l, a_0, b_0\}$ is the set of all parameters used in hyperpriors. Although a direct analytical predictive distribution does not exist in BKTF, we can use MCMC samples to obtain the mean and variance of all points on the grid, thus offering an enumeration-based approach to define AF.

We define a Bayesian variant of the UCB as the AF by approximating the predictive mean and variance (or uncertainty) in ordinary GP-based UCB with the values calculated from MCMC sampling. For every MCMC sample after burn-in, i.e., $k > K_0$, we can estimate an output tensor $\tilde{\boldsymbol{\mathcal{F}}}^{(k)}$ over the entire grid space using the latent factors $(\boldsymbol{g}_d^r)^{(k)}$ and the weight vector $\boldsymbol{\lambda}^{(k)}$: $\tilde{\boldsymbol{\mathcal{F}}}^{(k)} = \sum_{r=1}^R \lambda_r^{(k)} (\boldsymbol{g}_1^r)^{(k)} \circ (\boldsymbol{g}_2^r)^{(k)} \circ \cdots \circ (\boldsymbol{g}_D^r)^{(k)}$. We can then compute the corresponding mean and variance tensors of the $(K - K_0)$ samples $\{\tilde{\boldsymbol{\mathcal{F}}}^{(k)}\}_{k=K_0+1}^K$, and denote the two tensors by $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$, respectively. The approximated predictive distribution at each point $\boldsymbol{x}$ in the space becomes $\tilde{f}(\boldsymbol{x}) \sim \mathcal{N}(u(\boldsymbol{x}), v(\boldsymbol{x}))$. Following the definition of UCB in Eq. (5), we define Bayesian UCB (B-UCB) at location $\boldsymbol{x}$ as $\alpha_{\text{B-UCB}}(\boldsymbol{x} \mid \mathcal{D}, \beta, \boldsymbol{g}_d^r, \boldsymbol{\lambda}) = u(\boldsymbol{x}) + \beta\sqrt{v(\boldsymbol{x})}$. The next search/query point can be determined via $\boldsymbol{x}_{\text{next}} = \arg\max_{\boldsymbol{x} \in \{\prod_{d=1}^D S_d - \mathcal{D}_{n-1}\}} \alpha_{\text{B-UCB}}(\boldsymbol{x})$.

We summarize the implementation procedure of BKTF for BO in Appendix B (see Algorithm 2). Given the sequential nature of BO, when a new data point arrives at step $n$, we can start the MCMC with the last iteration of the Markov chains at step $n-1$ to accelerate model convergence. The main computational and storage cost of BKTF is to interpolate and save the tensors $\tilde{\boldsymbol{\mathcal{F}}} \in \mathbb{R}^{|S_1| \times \cdots \times |S_D|}$ over $(K - K_0)$ iterations for Bayesian AF estimation. This could be prohibitive when the MCMC sample size $K$ or the dimensionality $D$ becomes large. To avoid saving the tensors, in practice, we can simply use the maximum values of each entry over the $(K - K_0)$ iterations through iterative pairwise comparison. The number of samples after burn-in then implies the value of $\beta$ in $\alpha_{\text{B-UCB}}$. We adopt this simple AF in our numerical experiments.

A critical challenge in BKTF is that tensor size grows exponentially with the number of dimensions. To decrease the computational burden of enumeration-based AF, we also implement BKTF with random discretization—randomly selecting candidate points instead of reconstructing the whole space, denoted as **BKTFrandom**. BKTFrandom can be applied to functions with higher dimensions (e.g., $D > 10$). We discuss the comparison between BKTF and BKTFrandom in Experiments on test functions, see Section 5.1.

# 4 Related work

The key of BO is to effectively characterize the posterior distribution of the objective function from a limited number of observations. The most relevant work to our study is the *Bayesian Kernelized Factorization* (BKF) framework, which has been mainly used for modeling large-scale and multidimensional spatiotemporal data with UQ. The key idea is to parameterize the multidimensional stochastic processes using a factorization model, in which specific priors are used to encode spatial and temporal dependencies. Signature examples of BKF include spatial dynamic factor model (SDFM) [15], variational Gaussian process factor analysis (VGFA) [16], and Bayesian kernelized matrix/tensor factorization (BKMF/BKTF) [10, 11, 17]. A common solution in these models is to use GP prior to modeling the factor matrices, thus encoding spatial and temporal dependencies. In addition, for categorical dimensions, BKTF uses an inverse-Wishart prior to modeling the covariance matrix for the latent factors. A key difference among these methods is how inference is performed. SDFM and BKMF/BKTF are fully Bayesian hierarchical models and they rely on MCMC for model inference, where the factors can be updated via Gibbs sampling with conjugate priors; for learning the posterior distributions of kernel hyperparameters, SDFM uses the Metropolis-Hastings sampling, while BKMF/BKTF uses the more efficient slice sampling. On the other hand, VGFA uses variational inference to learn factor matrices, while kernel hyperparameters are learned through maximum a posteriori (MAP) estimation without UQ. Overall, BKTF has shown superior performance in modeling multidimensional spatiotemporal processes with high-quality UQ for 2D/3D spaces [11, 17].

The proposed BKTF surrogate models the objective function—as a single realization of a random process—using low-rank tensor factorization with random basis functions. This basis function-based specification is closely related to multidimensional Karhunen-Loève (KL) expansion [18] for stochastic (spatial, temporal, and spatiotemporal) processes. Empirical analysis of KL expansion is also known as proper orthogonal decomposition (POD). With a known kernel/covariance function, truncated KL expansion allows us to approximate the underlying random process using a set of eigenvalues and eigenfunctions derived from the kernel function. Numerical KL expansion is often referred to as the Garlekin method, and in practice, the basis functions are often chosen as prespecified and deterministic functions [12, 19], such as Fourier basis, wavelet basis, orthogonal polynomials, B-splines, empirical orthogonal functions, radial basis functions (RBF), and Wendland functions (i.e., compactly supported RBF) (see, e.g., [20], [21], [22], [23]). However, the quality of UQ will be undermined as the randomness is fully attributed to the coefficients $\{\lambda_r\}$; in addition, these methods also require a large number of basis functions (or a large rank) to fit complex stochastic processes. Different from methods with fixed/known basis functions, BKTF uses a Bayesian hierarchical modeling framework to better capture the randomness and uncertainty in the data, in which GP priors are used to model the latent factors (i.e., basis functions are also random processes) on different dimensions, and hyperpriors are introduced on the kernel hyperparameters. Therefore, BKTF becomes a fully Bayesian version of multidimensional KL expansion for stochastic processes with unknown covariance from partially observed data, however, without imposing any orthogonal constraint on the basis functions. Following the analysis in section 3.2, BKTF is also a special case of a two-layer deep Gaussian process [24, 7], where the first layer produces latent factors for each dimension, and the second layer has a multilinear kernel parameterized by all latent factors.

# 5 Experiments

## 5.1 Optimization for benchmark test functions

We test the proposed BKTF model for BO on seven benchmark functions that are used for global optimization problems [25], with dimension $D$ ranging from 2 to 10. All selected standard functions are multimodal; detailed descriptions are given in Appendix D. In fact, we can visually see that the standard Damavandi/Schaffer/Griewank functions in Figure 7 (see Appendix D) indeed have a

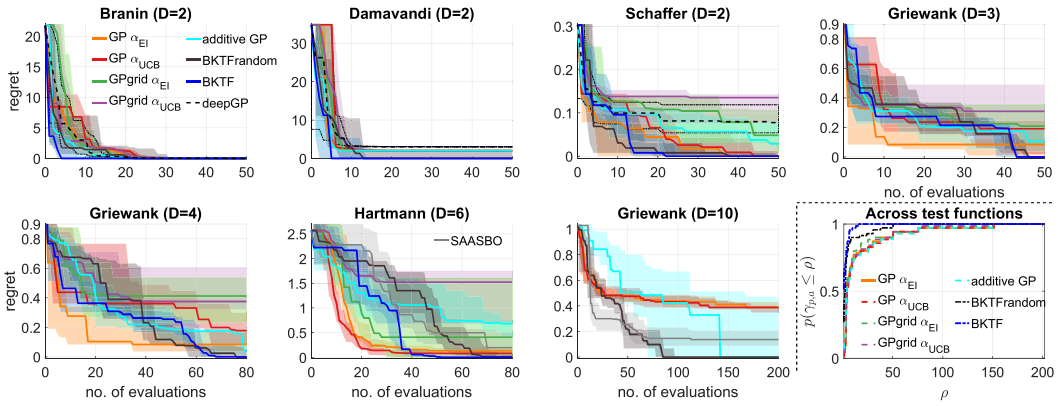



Figure 2: Optimization on benchmark test functions, where medians with 25% and 75% quartiles of 10 runs are compared.

low-rank structure. For each function, we assume the initial dataset $\mathcal{D}_0$ contains $n_0 = D$ observed data pairs, and we set the total number of query points to $N = 80$ for 4D Griewank and 6D Hartmann function, $N = 200$ for 10D Griewank, and $N = 50$ for others. We rescale the input search range to $[0, 1]$ for all dimensions and normalize the output data using z-score normalization.

**Model configuration** When applying BKTF to continuous test functions, we introduce $m_d$ interpolation points for the $d$th dimension of the input space. The values of $m_d$ used for each benchmark function are predefined and given in Table 1 (see Appendix D). Setting the resolution grid will require certain prior knowledge (e.g., smoothness of the function); and it also depends on the available computational resources and the number of entries in the tensor, which grows exponentially with $m_d$. In practice, we find that setting $m_d = 10 \sim 100$ is sufficient for most problems. We set the CP rank $R = 2$, and for each BO function evaluation run 400 MCMC iterations for model inference where the first 200 iterations are taken as burn-in. We use Matérn 3/2 kernel as the covariance function for all the test functions.

Note that for the 10D Griewank function, the grid-based models do not work, and only models built in continuous space and BKTFrandom can be performed. For BKTFrandom, in each evaluation we randomly select 20k points in the search space as candidates and choose the one with the best AF as the next evaluation location.

**Baselines** We compare BKTF with the following BO methods that use GP as the surrogate model: (1) GP $\alpha_{\text{EI}}$ and (2) GPgrid $\alpha_{\text{EI}}$: GP as the surrogate model and EI as the AF, in continuous space $\prod_{d=1}^{D} \mathcal{X}_d$ and Cartesian grid space $\prod_{d=1}^{D} S_d$, respectively; (3) GP $\alpha_{\text{UCB}}$ and (4) GPgrid $\alpha_{\text{UCB}}$: GP as the surrogate model with UCB as the AF where $\beta = 2$, in $\prod_{d=1}^{D} \mathcal{X}_d$ and $\prod_{d=1}^{D} S_d$, respectively; (5) additive GP: the sum of two 1st-order additive kernels per dimension as the surrogate with EI as the AF, in continuous space. This baseline has the same number of latent functions as BKTF ($R = 2$) but in a sum-based manner; (6) deepGP: a two-layer fully-Bayesian deep GP with EI as the AF, implemented with the *deepgp* package[1].

Similar as the setting for BKTF, we use Matérn 3/2 kernel in all GP models. Given the computational cost, we only compare deepGP on 2D functions [9]. For AF optimization in GP $\alpha_{\text{EI}}$ and GP $\alpha_{\text{UCB}}$, we first use the DIRECT algorithm [26] and then apply the Nelder-Mead algorithm [27] to further search if there exist better solutions. We also compare with SAASBO (Sparse Axis-Aligned Subspace) [28] with Hamiltonian Monte Carlo sampling, implemented using BoTorch [29], on the 6D Hartmann and 10D Griewank functions.

**Results** To compare the optimization performance of different models on the benchmark functions, we define the absolute error between the global optimum $f^\star$ and the current estimated global optimum $\hat{f}^\star$, i.e., $\left| f^\star - \hat{f}^\star \right|$, as the *regret*, and examine how *regret* varies with the number of function

[1] `https://CRAN.R-project.org/package=deepgp`

evaluations. We run the optimization 10 times for every test function with a different set of initial observations. The results are summarized in Figure 2. We see that for the 2D functions Branin and Schaffer, BKTF clearly finds the global optima much faster than GP surrogate-based baselines. For Damavandi function, where the global minimum ($f(\boldsymbol{x}^\star) = 0$) is located in a small sharp area while the local optimum ($f(\boldsymbol{x}) = 2$) is located at a large smooth area (see Figure 7 in Appendix D), GP-based models are trapped around the local optima in most cases ( i.e., *regret* $= 2$) and cannot jump out. In contrast, BKTF explores the global characteristics of the objective function over the entire search space and reaches the global optimum within 10 iterations of function evaluations. For higher dimensional Griewank and Hartmann functions, BKTF successfully arrives at the global optima under the given observation budgets, while GP-based comparison methods are prone to be stuck around local optima. We compare the *regret* at the last iteration in Table 2 (Appendix E.2), along with the average cost of evaluations. The enumeration-based GP surrogates, i.e., GPgrid $\alpha_{\text{EI}}$ and GPgrid $\alpha_{\text{UCB}}$, perform a little better than direct GP-based search, i.e., GP $\alpha_{\text{EI}}$ and GP $\alpha_{\text{UCB}}$ on Damavandi function, but worse on others. This means that the discretization, to some extent, offers possibilities for searching all the alternative points in the space, since in each function evaluation, every sample in the space is equally compared solely based on the predictive distribution. Additive GP is comparable with $R = 2$ BKTF; while the results demonstrate that BKTF can be much more flexible than additive GP. As for BKTFrandom, we see that it can alleviate the curse of dimensionality and be applied for higher-dimensional problems that may not be performed with a grid but at the cost of more evaluation budgets, particularly it costs more iterations for lower-dimensional functions compared with BKTF.

Overall, BKTF reaches the global optimum for every test function and shows superior performance for complex objective functions with a faster convergence rate. To intuitively compare the overall performance of different models across multiple experiments/functions, we further estimate performance profiles (PPs) [30] (see Appendix E.1), and compute the area under the curve (AUC) for quantitative analysis (see bottom right of Figure 2 and Table 2 in Appendix E.2). Our results show that BKTF obtains the best performance across all functions.

**Interpretable latent space.** The sampled latent functions are interpretable and imply the underlying correlations of the objective function. We illustrate the learned periodic (global) patterns in Appendix E.3. **Effects of hyperpriors.** Since we build a fully Bayesian model, the hyperparameters of the covariance functions can be updated automatically from the data likelihood and hyperprior. Note that in optimization scenarios where the observations are scarce, the prediction performance of BKTF highly depends on the hyperprior settings, i.e., $\boldsymbol{\theta}_0 = \{\mu_l, \tau_l, a_0, b_0\}$. We discuss the effects of hyperpriors in Appendix E.4. **Effects of rank.** The only predefined model parameter is the model rank, all other model parameters and hyperparameters are sampled with MCMC. We discuss the effects of rank on the 2D nonstationary nonseparable function defined in Introduction (see Figure 1) in Appendix E.5. We see that BKTF is robust to rank specification and can find the global solution efficiently with rank set as 2, 4, 6 and 8.

### 5.2 Hyperparameter tuning for machine learning

In this section, we evaluate the performance of BKTF for automatic machine learning (ML) tasks. We compare different models to optimize the hyperparameters of two ML algorithms—random forest (RF) and neural network (NN)—on classification for the MNIST database of handwritten digits[2] and regression for the Boston housing dataset[3]. The tuning tasks involve both integer-valued and categorical parameters, and the details are given in the Appendix G. We treat those integer-valued dimensions as continuous and use a Matérn 3/2 kernel for the basis functions. Given the size of the hyperparameter space, we perform BKTFrandom for classification and BKTF for regression. We assume that the number of initial observations $\mathcal{D}_0$ equals the number of tuning hyperparameters. The total budget $N$ is 20 for the classification task and 50 for the regression. We implement RF algorithms using the scikit-learn package and construct NN models by Keras with 2 hidden layers. All other model hyperparameters are set as the default values.

**Model configuration** We treat all the integer hyperparameters as samples from a continuous space and then generate the corresponding Cartesian product space $\prod_{d=1}^{D} S_d$. One can interpret

[2] `http://yann.lecun.com/exdb/mnist/`

[3] `https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html`

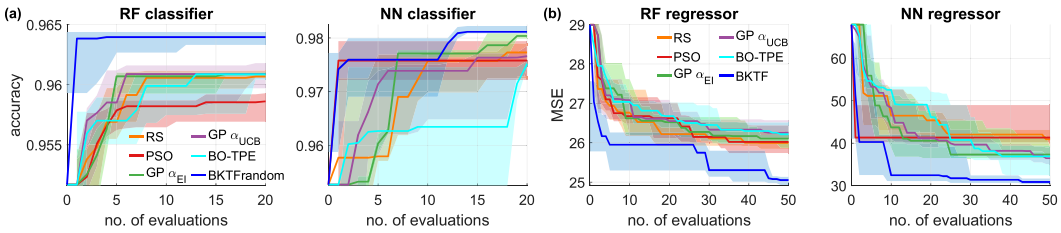



Figure 3: Comparison of hyperparameter tuning for ML tasks: (a) classification; (b) regression.

the candidate values for each hyperparameter as the interpolation points in the corresponding input dimension. The size of the spanned space $\prod S_d$ is $91 \times 46 \times 64 \times 10 \times 11 \times 2$ and $91 \times 46 \times 13 \times 10$ for RF classifier and RF regressor, respectively; $91 \times 49 \times 31 \times 18 \times 3 \times 2$ and $91 \times 49 \times 31$ for NN classifier and NN regressor respectively. We set the tensor rank $R = 4$ for BKTF, set $K = 400$ and $K_0 = 200$ for MCMC inference, and use the Matérn 3/2 kernel for capturing correlations.

**Baselines** In addition to GP surrogate-based GP $\alpha_{\text{EI}}$ and GP $\alpha_{\text{UCB}}$, we also compare with a baseline method: random search (RS), and two non-GP models: particle swarm optimization (PSO) [31] and Tree-structured Parzen Estimator (BO-TPE) [32], which are common methods for hyperparameter tuning. We exclude grid-based GP models as sampling the entire grid becomes infeasible.

**Results** We compare the accuracy for MNIST classification and MSE (mean squared error) for Boston housing regression both in terms of the number of function evaluations and still run the optimization processes ten times with different initial datasets $\mathcal{D}_0$. The results obtained by different BO models are given in Figure 3, and the final classification accuracy and regression MSE are compared in Table 5 (see Appendix H). For BKTF, we see from Figure 3 that the width between the two quartiles of accuracy and error decreases as more iterations are evaluated, and the median curves present better convergence rates compared to the baselines. Table 5 also shows that the proposed BKTF surrogate achieves the best final mean accuracy and regression error with small standard deviations. The results above demonstrate the advantage of BKTF as a surrogate.

## 6 Conclusion

This paper proposes to use Bayesian Kernelized Tensor Factorization (BKTF) as a new surrogate model for Bayesian optimization with mixed variables (both discrete/categorical and continuous) when the dimensionality is relatively small (e.g., say $D < 10$). Compared with traditional GP surrogates, the BKTF surrogate is more flexible and adaptive to data thanks to the Bayesian hierarchical specification, which provides high-quality UQ for BO tasks. The tensor factorization model behind BKTF offers an effective solution to capture global/long-range correlations and cross-dimension correlations. The inference of BKTF is achieved through MCMC, which provides a natural solution for acquisition. Experiments on both test function optimization and ML hyperparameter tuning confirm the superiority of BKTF as a surrogate for BO. Moreover, the tensor factorization framework makes it straightforward to adapt BKTF to handle multivariate and functional output (see e.g., [33, 34]), by directly treating the output coordinates as part of the input. A limitation of BKTF is that we restrict BO to a grid search space in order to leverage tensor factorization; however, we believe that designing a compatible grid space based on prior knowledge is not a challenging task.

There are several directions to be explored to make BKTF more scalable. Scalable GP solutions, such as sparse GP [14] and Gaussian Markov Random Field (GMRF) [35], can be introduced to reduce the inference cost when $|S_d|$ becomes large. The current MCMC-based acquisition method requires explicit reconstruction of the whole tensor, which quickly becomes infeasible when $D$ becomes large (e.g., $D > 20$). A natural question is whether it is possible to achieve efficient acquisition directly using the basis functions and the corresponding weights without explicitly constructing the tensors. This problem corresponds to finding/locating the maximum entry of a tensor given its low-rank decomposition (see e.g., [36]).

This work aims to advance the field of probabilistic machine learning, particularly Bayesian optimization. Regardless of the model limitations, it has the potential of misuse for ML algorithms.

# References


[1] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

[2] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[3] Robert B Gramacy. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. Chapman and Hall/CRC, 2020.

[4] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[5] David K Duvenaud, Hannes Nickisch, and Carl Rasmussen. Additive Gaussian processes. *Advances in neural information processing systems*, 24, 2011.

[6] Mickael Binois and Nathan Wycoff. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26, 2022.

[7] Andreas Damianou and Neil D Lawrence. Deep Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 207–215, 2013.

[8] Ali Hebbal, Loïc Brevault, Mathieu Balesdent, El-Ghazali Talbi, and Nouredine Melab. Bayesian optimization using deep gaussian processes with applications to aerospace system design. *Optimization and Engineering*, 22:321–361, 2021.

[9] Annie Sauer, Robert B Gramacy, and David Higdon. Active learning for deep gaussian process surrogates. *Technometrics*, 65(1):4–18, 2023.

[10] Mengying Lei, Aurelie Labbe, Yuankai Wu, and Lijun Sun. Bayesian kernelized matrix factorization for spatiotemporal traffic data imputation and kriging. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):18962–18974, 2022.

[11] Mengying Lei, Aurelie Labbe, and Lijun Sun. Bayesian complementary kernelized learning for multidimensional spatiotemporal data. *arXiv preprint arXiv:2208.09978*, 2022.

[12] Noel Cressie, Matthew Sainsbury-Dale, and Andrew Zammit-Mangion. Basis-function models in spatial statistics. *Annual Review of Statistics and Its Application*, 9:373–400, 2022.

[13] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[14] Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[15] Hedibert Freitas Lopes, Esther Salazar, and Dani Gamerman. Spatial dynamic factor analysis. *Bayesian Analysis*, 3(4):759–792, 2008.

[16] Jaakko Luttinen and Alexander Ilin. Variational Gaussian-process factor analysis for modeling spatio-temporal data. *Advances in Neural Information Processing Systems*, 22:1177–1185, 2009.

[17] Mengying Lei, Aurelie Labbe, and Lijun Sun. Scalable spatiotemporally varying coefficient modeling with bayesian kernelized tensor regression. *arXiv preprint arXiv:2109.00046*, 2021.

[18] Limin Wang. *Karhunen-Loeve expansions and their applications*. London School of Economics and Political Science (United Kingdom), 2008.

[19] Holger Wendland. *Scattered Data Approximation*, volume 17. Cambridge university press, 2004.

[20] Rommel G Regis and Christine A Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4):497–509, 2007.

[21] Gregory Beylkin, Jochen Garcke, and Martin J Mohlenkamp. Multivariate regression and machine learning with sums of separable functions. *SIAM Journal on Scientific Computing*, 31(3):1840–1857, 2009.

[22] Christopher K Wikle and Noel Cressie. A dimension-reduced approach to space-time kalman filtering. *Biometrika*, 86(4):815–829, 1999.

[23] Mathilde Chevreuil, Régis Lebrun, Anthony Nouy, and Prashant Rai. A least-squares method for sparse low rank approximation of multivariate functions. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):897–921, 2015.

[24] Alexandra M Schmidt and Anthony O'Hagan. Bayesian inference for non-stationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758, 2003.

[25] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimization problems. *arXiv preprint arXiv:1308.4008*, 2013.

[26] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79(1):157–181, 1993.

[27] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[28] David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*, pages 493–503. PMLR, 2021.

[29] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[30] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[31] Jun Tang, Gang Liu, and Qingtao Pan. A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. *IEEE/CAA Journal of Automatica Sinica*, 8(10):1627–1643, 2021.

[32] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[33] Dave Higdon, James Gattiker, Brian Williams, and Maria Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

[34] Shandian Zhe, Wei Xing, and Robert M Kirby. Scalable high-order gaussian process regression. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2611–2620. PMLR, 2019.

[35] Havard Rue and Leonhard Held. *Gaussian Markov random fields: theory and applications*. Chapman and Hall/CRC, 2005.

[36] Anastasiia Batsheva, Andrei Chertkov, Gleb Ryzhakov, and Ivan Oseledets. Protes: probabilistic optimization with tensor sampling. *Advances in Neural Information Processing Systems*, 36:808–823, 2023.

# Appendix

# Contents (Appendix)



# A Model inference

Assume an observation dataset $\mathcal{D}_n = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, we exploit an efficient element-wise Gibbs sampling algorithm for model inference.

## A.1 Sampling latent functions

Given the Gaussian prior and Gaussian likelihood of the latent factors $\boldsymbol{g}_d^r$, their posterior distributions are still Gaussian. Let $y_r^i = y_i - \sum_{\substack{h=1 \\ h \neq r}}^R \lambda_h \prod_{d=1}^D g_d^h\left(x_d^i\right)$ and $\boldsymbol{y}_r = [y_r^1, \ldots, y_r^n]^\top \in \mathbb{R}^n$ for $r = 1, \ldots, R$. Every $\boldsymbol{y}_r$ generates a $D$-dimensional tensor $\boldsymbol{\mathcal{Y}}_r \in \mathbb{R}^{|S_1| \times \cdots \times |S_D|}$ in the Cartesian product space $\prod_{d=1}^D S_d$. We define a binary tensor $\boldsymbol{\mathcal{O}}$ with the same size of $\boldsymbol{\mathcal{Y}}$ indicating the locations of the observation data, where $o\left(\boldsymbol{x}_i\right) = 1$ for $i \in [1, n]$, and other values are zero. The posterior of

$\boldsymbol{g}_d^r$ is given by

$$p\left(\boldsymbol{g}_d^r \mid -\right) = \mathcal{N}\left(\boldsymbol{g}_d^r \;\middle|\; \left[\boldsymbol{\mu}_d^r\right]^*, \left(\left[\boldsymbol{\Lambda}_d^r\right]^*\right)^{-1}\right), \tag{11}$$

where

$$\left[\boldsymbol{\mu}_d^r\right]^* = \left(\left[\boldsymbol{\Lambda}_d^r\right]^*\right)^{-1} \underbrace{\left(\tau\left(\boldsymbol{Y}_{r(d)} \circledast \boldsymbol{O}_{(d)}\right)\left(\lambda_r \bigotimes_{\substack{h=D \\ h\neq d}}^{1} \boldsymbol{g}_h^r\right)\right)}_{\boldsymbol{a}\in\mathbb{R}^{|S_d|}}, \tag{12}$$

$$\left[\boldsymbol{\Lambda}_d^r\right]^* = \tau \operatorname{diag}\underbrace{\left(\boldsymbol{O}_{(d)}\left(\lambda_r \bigotimes_{\substack{h=D \\ h\neq d}}^{1} \boldsymbol{g}_h^r\right)^2\right)}_{\boldsymbol{b}\in\mathbb{R}^{|S_d|}} + \left(\boldsymbol{K}_d^r\right)^{-1}. \tag{13}$$

$\boldsymbol{Y}_{r(d)}$ and $\boldsymbol{O}_{(d)}$ are mode-$d$ unfoldings of $\boldsymbol{\mathcal{Y}}_r$ and $\boldsymbol{\mathcal{O}}$, respectively, with the size of $|S_d| \times \left(\prod_{\substack{h=1 \\ h\neq d}}^{D} |S_h|\right)$. Note that the vector term $\boldsymbol{a}$ in $\left[\boldsymbol{\mu}_d^r\right]^*$ and $\boldsymbol{b}$ in $\left[\boldsymbol{\Lambda}_d^r\right]^*$, which are only relevant to the $n$ observations and corresponding function values, can be computed element-wise instead of using matrix multiplication and Kronecker product. The point-wise computation can dramatically reduce the computational cost, especially for a relatively large $D$, since in such case the number of observations in BO can be much smaller compared to the number of samples in the entire grid space, i.e., $n \ll \prod_{d=1}^{D} |S_d|$.

### A.2 Sampling kernel hyperparameters

We update the kernel lengthscale hyperparameters $l_d^r$ from their marginal posteriors by integrating out the latent factors. Let $\left[\boldsymbol{y}_r\right]_d = \boldsymbol{O}_d \operatorname{vec}\left(\boldsymbol{Y}_{r(d)}\right) \in \mathbb{R}^n$, where $\boldsymbol{O}_d \in \mathbb{R}^{n\times\left(\prod_{d=1}^{D}|S_d|\right)}$ is a binary matrix obtained by removing the rows corresponding to zeros in $\operatorname{vec}\left(\boldsymbol{O}_{(d)}\right)$ from $\boldsymbol{I}_{\prod_{d=1}^{D}|S_d|}$. When sampling the posteriors for kernel hyperparameters under a given $d$ and $r$, their marginal likelihoods only relate to $\left[\boldsymbol{y}_r\right]_d$. The log marginal likelihood of $l_d^r$, for example, is:

$$\begin{aligned}
\log p\left(\left[\boldsymbol{y}_r\right]_d \mid l_d^r\right) &\propto -\frac{1}{2}\left(\left[\boldsymbol{y}_r\right]_d\right)^\top \boldsymbol{\Sigma}^{-1}_{\left[\boldsymbol{y}_r\right]_d|l_d^r}\left[\boldsymbol{y}_r\right]_d - \frac{1}{2}\log\det\left(\boldsymbol{\Sigma}_{\left[\boldsymbol{y}_r\right]_d|l_d^r}\right) \\
&\propto \frac{\tau^2}{2}\underbrace{\left(\left[\boldsymbol{y}_r\right]_d\right)^\top \boldsymbol{H}\left(\left[\boldsymbol{\Lambda}_d^r\right]^*\right)^{-1}\boldsymbol{H}^\top\left[\boldsymbol{y}_r\right]_d}_{c} - \frac{1}{2}\log\det\left(\left[\boldsymbol{\Lambda}_d^r\right]^*\right) - \frac{1}{2}\log\det\left(\boldsymbol{K}_d^r\right),
\end{aligned} \tag{14}$$

where $\boldsymbol{H} = \boldsymbol{O}_d\left(\lambda_r \bigotimes_{\substack{h=D \\ h\neq d}}^{1} \boldsymbol{g}_h^r \otimes \boldsymbol{I}_{(|S_d|)}\right) \in \mathbb{R}^{n\times|S_d|}$ and $\boldsymbol{\Sigma}_{\left[\boldsymbol{y}_r\right]_d|l_d^r} = \boldsymbol{H}\boldsymbol{K}_d^r\boldsymbol{H}^\top + \tau^{-1}\boldsymbol{I}_n$. The term $c$ in Eq. (14) is a scalar that can be computed with $\boldsymbol{u}^\top\boldsymbol{u}$, where $\boldsymbol{u} = \left(\boldsymbol{L}_d^r\right)^{-1}\boldsymbol{a}$ is a vector of length $|S_d|$; $\boldsymbol{L}_d^r = \operatorname{chol}\left(\left[\boldsymbol{\Lambda}_d^r\right]^*\right)$ is the Cholesky factor matrix of $\left[\boldsymbol{\Lambda}_d^r\right]^*$. This means that the complicated term $c$ can also be calculated element-wise, and it leads to a fast learning process. With the marginal likelihood and predefined log-normal hyperpriors, we can get the marginal posteriors of the kernel hyperparameters straightforwardly; and we update them by using the slice sampling algorithm presented in [17].

### A.3 Sampling $\boldsymbol{\Lambda}_d$ for latent functions on categorical inputs

For latent factors in dimensions with categorical variables/inputs, we sample the precision hyperparameter $\boldsymbol{\Lambda}_d$ in its prior distribution from a Wishart distribution:

$$\boldsymbol{\Lambda}_d \mid - \sim \mathcal{W}\left(\left(\boldsymbol{G}_d\boldsymbol{G}_d^\top + \boldsymbol{I}_{|S_d|}\right)^{-1}, |S_d| + R\right). \tag{15}$$

### A.4 Sampling weight vector

Every observed data point has the following distribution:

$$y_i \sim \mathcal{N}\left(\sum_{r=1}^{R}\lambda_r\prod_{d=1}^{D}g_d^r\left(x_d^i\right)=\left[\prod_{d=1}^{D}g_d^1\left(x_d^i\right),\ldots,\prod_{d=1}^{D}g_d^R\left(x_d^i\right)\right]\boldsymbol{\lambda},\tau^{-1}\right),\ i=1,\ldots,n. \quad (16)$$

Let $\boldsymbol{g}\left(\boldsymbol{x}_i\right)=\left(\prod_{d=1}^{D}g_d^1\left(x_d^i\right),\ldots,\prod_{d=1}^{D}g_d^R\left(x_d^i\right)\right)^\top\in\mathbb{R}^R$ and $\boldsymbol{y}=\left(y_1,\ldots,y_n\right)^\top\in\mathbb{R}^n$; we then have $\boldsymbol{y}\sim\mathcal{N}\left(\tilde{\boldsymbol{G}}^\top\boldsymbol{\lambda},\tau^{-1}\boldsymbol{I}_n\right)$, where $\tilde{\boldsymbol{G}}=[\boldsymbol{g}(\boldsymbol{x}_1),\ldots,\boldsymbol{g}(\boldsymbol{x}_n)]\in\mathbb{R}^{R\times n}$. The posterior of $\boldsymbol{\lambda}$ follows a Gaussian distribution

$$p\left(\boldsymbol{\lambda}\mid-\right)\sim\mathcal{N}\left(\boldsymbol{\mu}_\lambda^*,\left(\boldsymbol{\Lambda}_\lambda^*\right)^{-1}\right), \quad (17)$$

where

$$\boldsymbol{\mu}_\lambda^*=\tau\left(\boldsymbol{\Lambda}_\lambda^*\right)^{-1}\tilde{\boldsymbol{G}}\boldsymbol{y}, \quad (18)$$

$$\boldsymbol{\Lambda}_\lambda^*=\tau\tilde{\boldsymbol{G}}\tilde{\boldsymbol{G}}^\top+\boldsymbol{I}_R. \quad (19)$$

### A.5 Sampling model noise precision

For precision $\tau$, we have a Gamma posterior

$$p\left(\tau\mid-\right)=\text{Gamma}\left(\tau\mid a^*,b^*\right), \quad (20)$$

where

$$a^*=a_0+\frac{1}{2}n, \quad (21)$$

$$b^*=b_0+\frac{1}{2}\sum_{i=1}^{n}\left(y_i-\sum_{r=1}^{R}\lambda_r\prod_{d=1}^{D}g_d^r\left(x_d^i\right)\right)^2. \quad (22)$$

# B Algorithm of BKTF for BO

**Algorithm 2:** BKTF for BO

**Input:** Initial dataset $\mathcal{D}_0$.
**for** $n = 1$ **to** $N$ **do**
  **for** $k = 1$ **to** $K$ **do**
    **for** $r = 1$ **to** $R$ **do**
      **for** $d = 1$ **to** $D$ **do**
        Draw kernel lengthscale hyperparameter $l_d^r$ or precision hyperparameter $\boldsymbol{\Lambda}_d$;
        Draw latent factors $(\boldsymbol{g}_d^r)^{(k)}$;
      **end for**
    **end for**
    Draw model noise precision $\tau^{(k)}$;
    Draw weight vector $\boldsymbol{\lambda}^{(k)}$;
    **if** $k > K_0$ **then**
      Compute and collect $\tilde{\boldsymbol{\mathcal{F}}}^{(k)}$.
    **end if**
  **end for**
  Compute mean $\boldsymbol{\mathcal{U}}$ and variance $\boldsymbol{\mathcal{V}}$ of $\{\tilde{\boldsymbol{\mathcal{F}}}^{(k)}\}$;
  Compute $\alpha_{\text{B-UCB}}\left(\boldsymbol{x} \mid \mathcal{D}_{n-1}, \beta\right)$ as a tensor;
  Find next $\boldsymbol{x}_n = \arg\max_{\boldsymbol{x}} \alpha_{\text{B-UCB}}\left(\boldsymbol{x} \mid \mathcal{D}_{n-1}, \beta\right)$;
  Augment the data $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\boldsymbol{x}_n, y_n\}$.
**end for**

# C Optimization for the 2D nonstationary and nonseparable function

## C.1 Data generation

The function in Figure 1(a) of the paper (see Section 1 Introduction) is a modification of the case study used in [11]. It is a $40 \times 40$ 2D process generated in a $[1, 2] \times [-1, 0]$ square, with

$$\boldsymbol{Y}(x_1, x_2) = \left(\cos\left(4\left[f_1(x_1) + f_2(x_2)\right]\right) + \sin\left(4\left[f_1(x_2) - f_2(x_1)\right]\right) - 1\right) \times \exp\left(-(x_1 - 0.5)^2 + \frac{(x_2 - 1)^2}{5}\right), \quad (23)$$

where $f(x_1) = x_1\left(\sin 2x_1 + 2\right)$, $f(x_2) = 0.2x_2\sqrt{99(x_2 + 1) + 4}$, $x_1 \in [1, 2]$, $x_2 \in [-1, 0]$. This is a nonstationary, nonseparable, and multimodel function, with the global maximum $f(\boldsymbol{x}^\star) = 0.6028$ at $\boldsymbol{x}^\star = (1.75, -0.55)$.

## C.2 Experimental setting

We randomly select $n_0 = 30$ data points as the initial data and run 50 iterations (i.e., budget) of evaluation for optimization. To compare different surrogates, we run the optimization for 20 replications with different initial datasets. For the proposed BKTF surrogate, we place Matérn 3/2 kernel functions on the latent factors, set the rank $R = 4$, and run 1000 MCMC samples for model inference where the first 600 samples are burn-in. For comparison of BO methods, we consider typical GP surrogate with both EI and UCB ($\beta = 2$) as the AF, denoted by GP $\alpha_{\text{EI}}$ and GP $\alpha_{\text{UCB}}$ respectively, and use the same Matérn 3/2 kernel for GP surrogate.

## C.3 Results

Figure 1(b) in Section 1 Introduction of the paper shows the medians along with the 25% and 75% quantiles of the optimization results from 20 runs. We see that GP $\alpha_{\text{EI}}$ and GP $\alpha_{\text{UCB}}$ cannot find the global optimum in most cases, and they easily get stuck in the lower left flat area which contains easily find local optima. Figure 1(c) illustrates the estimation surface of the function and the estimated AF surface from one run. It is clear that BKTF can capture global correlations with limited data. The

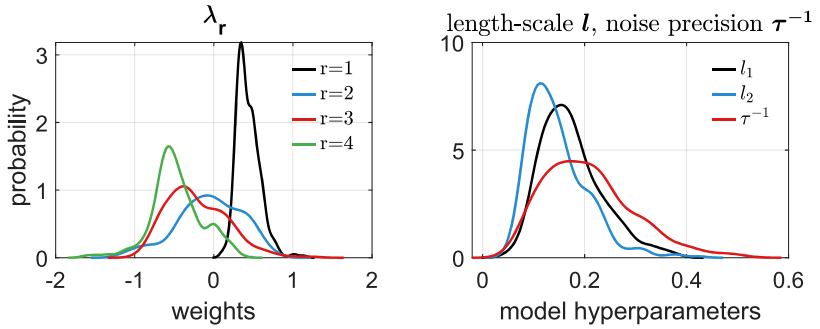



Figure 4: Optimization on the 2D function (Eq. 23): Posterior distributions of model hyperparameters learned by BKTF.

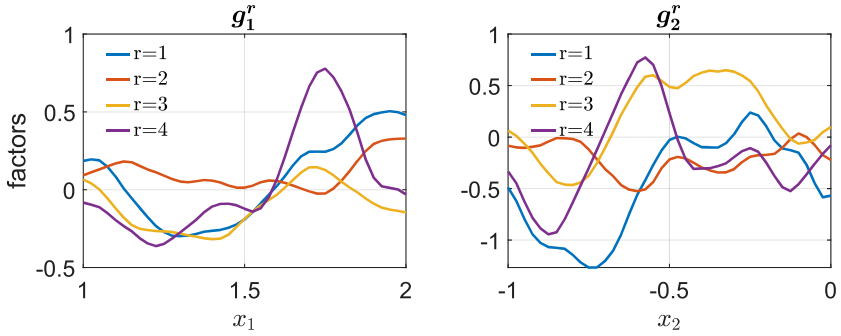
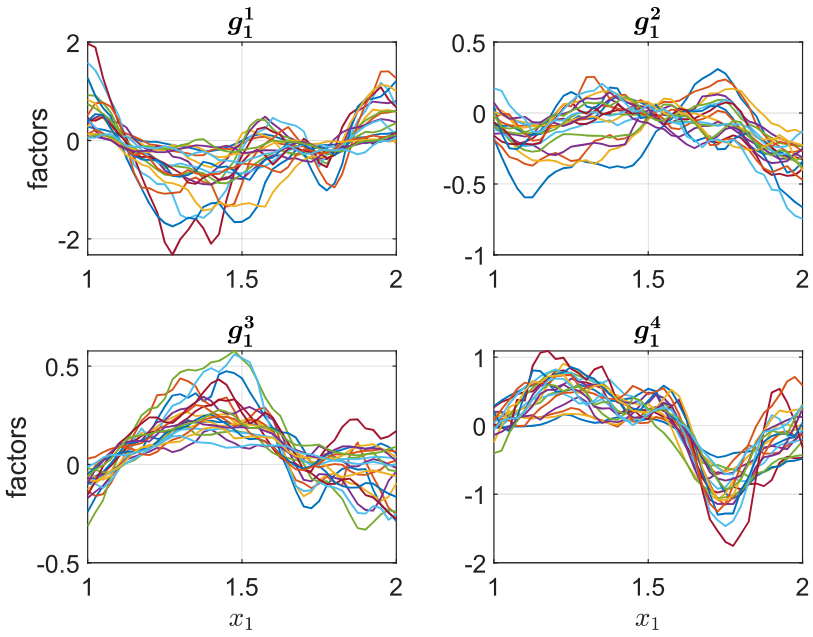


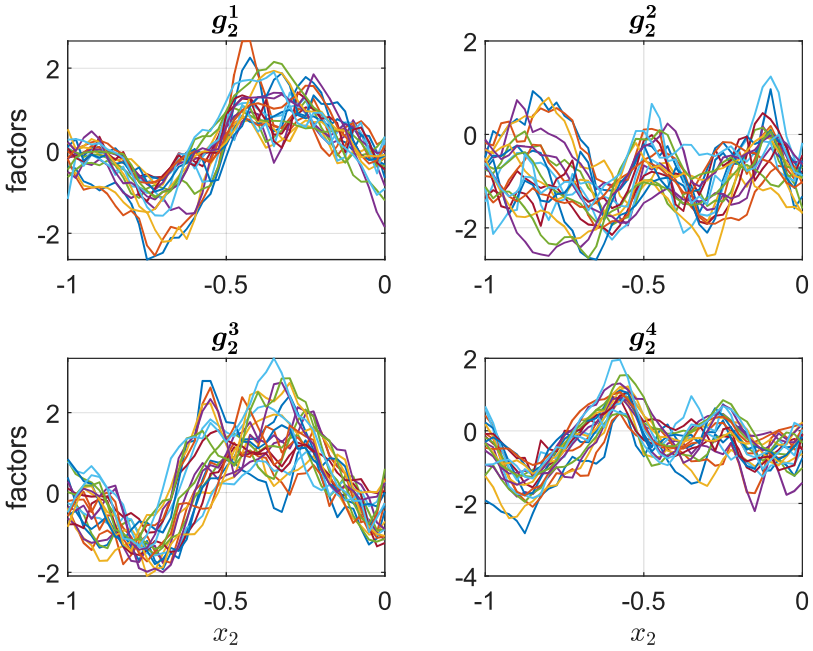

Figure 5: Optimization on the 2D function (Eq. 23): Mean of latent factors sampled by BKTF.

search points contain areas of almost every peak in the true function, and the peaks of its AF surface also reflect the peak area in $f$.

Figure 4 shows the approximated posterior distributions of model parameters in BKTF. Figure 5 and Figure 6 illustrate the posterior mean and the last 20 MCMC samples of the latent factors, respectively. Samples of the latent factors in panels (a) and (b) of Figure 6 explain the uncertainty learned by BKTF for this optimization problem.




(a) MCMC samples for latent factors ($d = 1$).




(b) MCMC samples for latent factors ($d = 2$).

Figure 6: Optimization on the 2D function (Eq. 23): The last 20 MCMC samples of the latent factors in the two dimensions learned by BKTF.

## D Benchmark test functions

We summarize the characteristics of the benchmark functions tested in Table 1. Figure 7 shows the functions with 2-dimensional inputs together with the 2D Griewank function. The functional expressions and more detailed features of these test functions are provided in the following.

Table 1: Summary of the studied benchmark functions.

| Function | $D$ | Search space | $m_d$ | Characteristics |
|---|---|---|---|---|
| Branin | 2 | $[-5, 10] \times [0, 15]$ | 14 | 3 global minima, flat |
| Damavandi | 2 | $[0, 14]^2$ | 71 | multimodal, global minimum located in small area |
| Schaffer | 2 | $[-10, 10]^2$ | 11 | multimodal, global optimum located close to local minima |
| Griewank | 3 | $[-10, 10]^3$ | 11 | multimodal, many widespread and regularly distributed local optima |
| | 4 | $[-10, 10]^4$ | 11 | |
| | 10 | $[-10, 10]^{10}$ | - | |
| Hartmann | 6 | $[0, 1]^6$ | 12 | multimodal, multi-input |

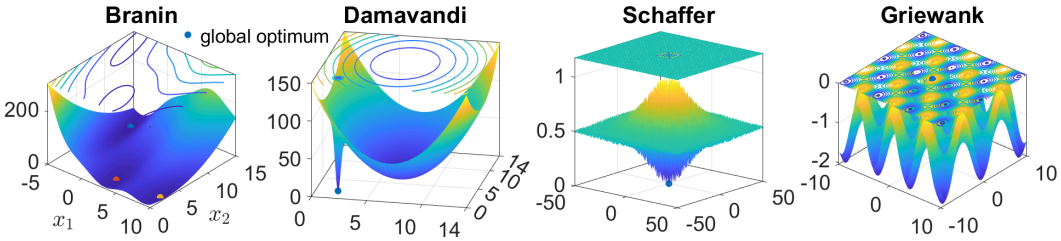



Figure 7: Examples of the tested benchmark functions.

**(1) Branin function** $(D = 2)$

$$f(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10, \tag{24}$$

where $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. It is a smooth but multimodal function with global minima $f(\boldsymbol{x}^*) = 0.3978873$ at three input points $\boldsymbol{x}^* = (-\pi, 12.275), (\pi, 2.275), (3\pi, 2.425)$.

**(2) Damavandi function** $(D = 2)$

$$f(x_1, x_2) = \left[1 - \left|\frac{\sin[\pi(x_1 - 2)]\sin[\pi(x_2 - 2)]}{\pi^2(x_1 - 2)(x_2 - 2)}\right|^5\right]\left[2 + (x_1 - 7)^2 + 2(x_2 - 7)^2\right], \tag{25}$$

where $x_d \in [0, 14]$. This is a multimodal function with the global minimum $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (2, 2)$.

**(3) Schaffer function** $(D = 2)$

$$f(x_1, x_2) = 0.5 + \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{\left[1 + 0.001\left(x_1^2 + x_2^2\right)\right]^2}, \tag{26}$$

where $x_d \in [-10, 10]$. The global minimum value is $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (0, 0)$. One characteristic of this function is that the global minimum is located very close to the local minima.

**(4) Griewank function** $(D = 3, 4, 10)$

$$f(\boldsymbol{x}) = 1 + \sum_{d=1}^{D}\frac{x_d^2}{4000} - \prod_{d=1}^{D}\cos\left(\frac{x_d}{\sqrt{d}}\right), \tag{27}$$

where $x_d \in [-10, 10]$. This is a multimodal function with global minimum $f(\boldsymbol{x}^*) = 0$ at $\boldsymbol{x}^* = (0, 0)$.

**(5) Hartmann function** $(D = 6)$ A nonseparable function with multidimensional inputs and multiple local minima.

$$f(\boldsymbol{x}) = -\sum_{j=1}^{4} c_j \exp\left(-\sum_{d=1}^{6} a_{jd}(x_d - b_{jd})^2\right), \tag{28}$$

where

$$\boldsymbol{A} = [a_{jd}] = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \ \boldsymbol{c} = [c_j] = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}, \\ \boldsymbol{B} = [b_{jd}] = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5586 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2833 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}, \tag{29}$$

$x_d \in [0, 1]$. The 6-dimensional case has 6 local minima, and the global minimum is $f(\boldsymbol{x}^*) = -3.32237$ at $\boldsymbol{x}^* = (0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.657301)$.

Note that all these minimization problems can be easily transformed as a maximization optimization problem, i.e., $\max -f(\boldsymbol{x})$.

# E Supplementary results on benchmark test functions

## E.1 Performance profiles

When computing the performance profiles (PPs), i.e., Dolan-Moré curves [30], we consider the number of function evaluations to find the global optimum as the performance measure. Specifically, let $t_{p,a}$ denote the number of evaluations used by method/solver $a$ to reach the global solution in experiment $p$ (a lower value is better). The value is equal to $N_p + 100$ if the method cannot find the global optimum with $N_p$ being the given observation budget for experiment $p$. The performance ratio

$$\gamma_{p,a} = \frac{t_{p,a}}{\min\{t_{p,a} : a \in \mathcal{A}\}}, \tag{30}$$

where $\mathcal{A}$ represents the set that includes all comparing models, and the performance profile for each method is the distribution of $p\left(\gamma_{p,a} \leq \rho\right)$ in terms of a factor $\rho$. We set $\rho = 1 : N_{\text{max}} + 1$, where $N_{\text{max}} = \max N_p$ is the largest observation budget assumed for the compared experiments. We define the problem set $\{\mathcal{P} \mid \forall p \in \mathcal{P}\}$ as the 10 runs for every function and draw the performance profiles of each model, also set $\mathcal{P}$ as the 70 experiments in the 7 tested functions and estimate the overall performance profiles.

We show the obtained PPs across test functions (i.e., overall PPs) in Figure 2 (bottom right) for illustration (see Section 5.1 of the paper); the results of all the tested functions are shown below in Figure 8. Note we only compute PPs for the methods that compared on all test functions, i.e., deepGP and SAASBO are not considered.

Table 2 gives the AUC (area under the curve) values of these curves, where the AUC of the overall performance profiles is taken as the metric to compare the overall performances. As can be seen, BKTF obtains the best performance across all the considered functions. In addition, for most of the test functions, the AUC of grid-based baseline models is comparable with those of continuous GP-based models, suggesting that discretization of the continuous space is feasible to simplify the optimization problem.

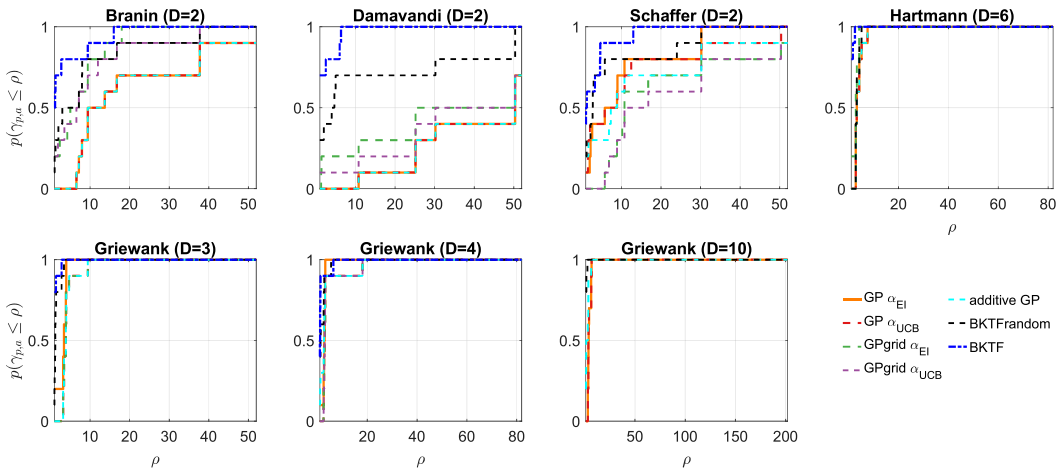



Figure 8: Performance profiles on the standard test functions.

## E.2 Quantitative comparison

We compare the last step *regret*, average costs of evaluations, and AUC of PPs of different methods on benchmark test functions in Table 2.

Table 2: Optimization on test functions: *regret* when $n = N$ (mean $\pm$ std.) / Average costs of evaluations / AUC of PPs.

| $f(\boldsymbol{x})$ ($D$) | | GP $\alpha_{\text{EI}}$ | GP $\alpha_{\text{UCB}}$ | GPgrid $\alpha_{\text{EI}}$ | GPgrid $\alpha_{\text{UCB}}$ | additive GP | BKTF random | BKTF | deepGP | SAAS BO |
|---|---|---|---|---|---|---|---|---|---|---|
| B (2) | *regret* | 0.01 $\pm$ 0.01 | 0.01 $\pm$ 0.01 | 0.31 $\pm$ 0.62 | 0.24 $\pm$ 0.64 | 0.05 $\pm$ 0.09 | **0.00** $\pm$ 0.00 | **0.00** $\pm$ 0.00 | 0.02 $\pm$ 0.02 | - |
| | Cost | $\approx 44$ | $\approx 42$ | $\approx 23$ | $\approx 36$ | $\approx 100$ | $\approx 47$ | $\approx$ **4** | $\approx 47$ | - |
| | AUC | 32.14 | 32.14 | 44.86 | 42.29 | 2.14 | 43.40 | **48.44** | - | - |
| D (2) | *regret* | 2.00 $\pm$ 0.00 | 2.00 $\pm$ 0.00 | 1.60 $\pm$ 0.80 | 2.00 $\pm$ 0.00 | 2.00 $\pm$ 0.00 | 0.60 $\pm$ 0.92 | **0.00** $\pm$ 0.00 | 3.00 $\pm$ 0.007 | - |
| | Cost | - | - | - | - | - | $\approx 48$ | $\approx$ **5** | - | - |
| | AUC | 12.17 | 12.17 | 19.95 | 17.10 | 12.17 | 37.05 | **49.82** | - | - |
| S (2) | *regret* | 0.02 $\pm$ 0.02 | 0.02 $\pm$ 0.02 | 0.10 $\pm$ 0.15 | 0.09 $\pm$ 0.07 | 0.03 $\pm$ 0.03 | **0.00** $\pm$ 0.00 | **0.00** $\pm$ 0.00 | 0.08 $\pm$ 0.04 | - |
| | Cost | $\approx 36$ | $\approx 44$ | $> 50$ | $> 50$ | $\approx 43$ | $\approx 54$ | $\approx$ **22** | $> 50$ | - |
| | AUC | 41.73 | 39.51 | 31.74 | 29.74 | 37.00 | 44.60 | **48.80** | - | - |
| G (3) | *regret* | 0.14 $\pm$ 0.14 | 0.25 $\pm$ 0.10 | 0.23 $\pm$ 0.13 | 0.22 $\pm$ 0.12 | 0.10 $\pm$ 0.09 | **0.00** $\pm$ 0.00 | **0.00** $\pm$ 0.00 | - | - |
| | Cost | $> 100$ | $> 100$ | $> 100$ | $> 100$ | $> 100$ | $\approx 47$ | $\approx$ **43** | - | - |
| | AUC | 48.90 | 47.69 | 47.69 | 47.69 | 47.69 | 50.44 | **50.78** | - | - |
| G (4) | *regret* | 0.10 $\pm$ 0.07 | 0.19 $\pm$ 0.12 | 0.38 $\pm$ 0.19 | 0.27 $\pm$ 0.17 | 0.13 $\pm$ 0.11 | **0.00** $\pm$ 0.00 | **0.00** $\pm$ 0.00 | - | - |
| | Cost | $> 100$ | $> 100$ | $> 100$ | $> 100$ | $> 100$ | $\approx 87$ | $\approx$ **68** | - | - |
| | AUC | 79.32 | 77.61 | 77.61 | 77.61 | 78.17 | 80.01 | **80.40** | - | - |
| H (6) | *regret* | 0.12 $\pm$ 0.07 | 0.07 $\pm$ 0.07 | 0.70 $\pm$ 0.70 | 0.79 $\pm$ 0.61 | 0.48 $\pm$ 0.17 | 1e-5 $\pm$ 1e-5 | **0.00** $\pm$ 0.00 | - | 0.19 $\pm$ 0.48 |
| | Cost | $> 100$ | $> 100$ | $> 100$ | $> 100$ | $> 100$ | $\approx 154$ | $\approx$ **60** | - | $> 100$ |
| | AUC | 78.11 | 78.11 | 79.18 | 78.11 | 78.11 | 78.55 | **80.78** | - | - |
| G (10) | *regret* | 0.36 $\pm$ 0.07 | 0.38 $\pm$ 0.10 | - $\pm$ - | - $\pm$ - | 0.25 $\pm$ 0.30 | **0.00** $\pm$ 0.00 | - | - | 0.14 $\pm$ 0.10 |
| | Cost | $> 200$ | $> 200$ | - | - | $> 150$ | $\approx$ **124** | - | - | $> 200$ |
| | AUC | 197.55 | 197.55 | - | - | 199.38 | **200.77** | - | - | - |
| Overall | AUC | 186.49 | 185.75 | 188.80 | 187.59 | 185.40 | 196.40 | **199.51** | - | - |

Best results are highlighted in bold fonts. B: Branin; D: Damavandi; S: Schaffer; G: Griewank; H: Hartmann.

### E.3 Interpretation of results

The basis functions learned by BKTF are interpretable. For example, Figure 9 shows the latent factors ($r = 1$) obtained at the last iteration of function evaluation in one run on 3D Griewank function. We see that BKTF can learn the periodicity (global structure) of the function benefited from the low-rank modeling. On the other hand, a stationary and separable GP cannot, other than using a specific kernel function such as the periodic kernel, which however requires strong prior knowledge to set the periodicity kernel hyperparameter.

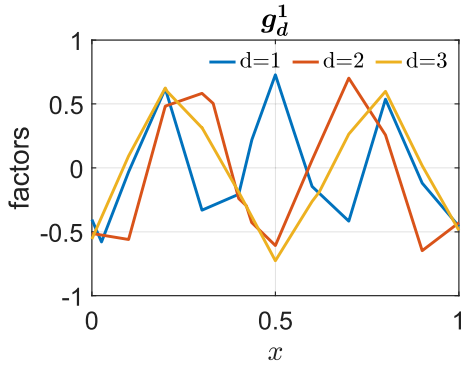



Figure 9: Examples of latent factors learned by BKTF on 3D Griewank function.

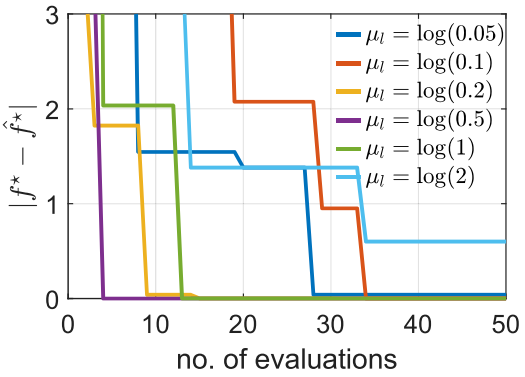



Figure 10: Effects of hyperpriors on Branin function: Optimization with different hyperpriors.

### E.4 Effects of hyperpriors

We compare the optimization performance on Branin function with different hyperprior settings in Figure 10 as an example to illustrate the effects of hyperpriors. Specifically, we compare the optimization results under several hyperprior assumptions of $\mu_l$, when $\tau_l^{-1}$ is set as 0.5. As can be seen, BKTF is not able to reach the global minimum with too small or too large mean assumptions (comparable to $[0, 1]$) on the kernel lengthscales $l$, for example in the cases where $\mu_l = \{\log(0.05), \log(2)\}$. In contrast, it finds the global optimum after 4 iterations of function evaluations when $\mu_l = \log(0.5)$, see the purple line. These imply the importance of hyperprior selection. The reason is that in the first several evaluations, since the observations are rare, the prior basically determines the exploration-exploitation balance and guides the search process.

Figure 11 shows the approximated posterior distributions for kernel hyperparameters and model noise variance when $\tau_l^{-1} = 0.5, \mu_l = \log(0.5)$. We see that for the re-scaled input space and normalized function output, the sampled length scales are around half of the input domain. Such settings are reasonable to capture the correlations between the observations and are also interpretable.

The effects of hyper-priors on other functions are similar, and we choose an appropriate setting relevant to the input range. The hyper-prior on $\tau$ impacts the uncertainty of the latent factors, for example a large model noise assumption allows more variances in the factors. The role of $\{a_0, b_0\}$ becomes more important when the objective function is complex that BKTF cannot well describe the function with limited observations. Generally, we select the priors that make the noise variances not quite large, such as the results of $\tau^{-1}$ shown in Figure 4 and Figure 11. An example of the uncertainty provided by BKTF is explained in Appendix C (see Figure 6).

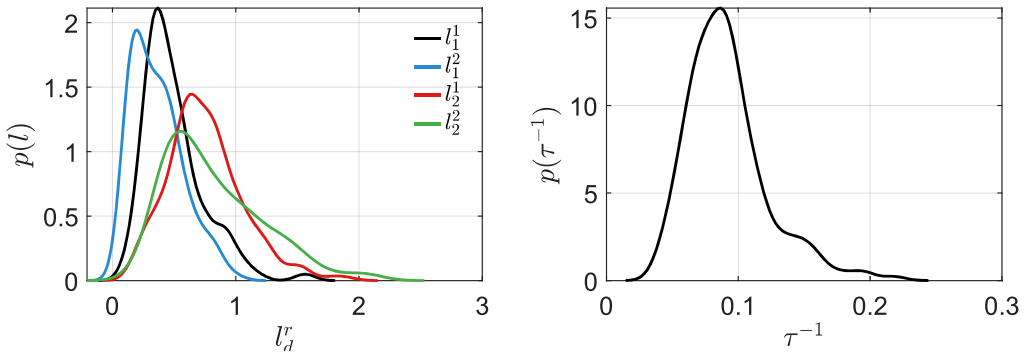



Figure 11: Effects of hyperpriors on Branin function: Posterior probability distributions of length-scales and model noise variance when $\tau_l^{-1} = 0.5, \mu_l = \log(0.5)$.

### E.5 Effects of rank

Under a fully Bayesian treatment, kernel hyperparameters of BKTF are automatically sampled by MCMC with proper priors; the only selected parameter is the model rank. We test the effects of rank specification for the proposed BKTF surrogate on the 2D nonstationary nonseparable function defined in Section 1 Introduction (see Figure 1 and Appendix C). We use the same experiment settings as in Appendix C, i.e., 30 initial observations and 50 budget.

The results are given in Figure 12 and 13, where we compare the optimization performance of BKTF with rank $R = \{2, 4, 6, 8\}$ and two GP-based surrogate models: GP $\alpha_{\text{EI}}$ and GP $\alpha_{\text{UCB}}$. To clearly illustrate the results, we only show the comparison on one run. In Figure 12, we compare the regret from different models, and in Figure 13 we compute and compare the mean CRPS (continuous ranked probability score) on the unobserved points.

CRPS is a widely applied metric for evaluating the performance of UQ for probabilistic models. With Gaussian likelihoods, CRPS can be defined as:

$$\text{CRPS} = -\frac{1}{n'}\sum_{i=1}^{n'}\sigma_i\left[\frac{1}{\sqrt{\pi}} - 2\psi\left(\frac{f_i-\hat{y}_i}{\sigma_i}\right) - \frac{f_i-\hat{y}_i}{\sigma_i}\left(2\Phi\left(\frac{f_i-\hat{y}_i}{\sigma_i}\right)-1\right)\right], \tag{31}$$

where $n'$ is the number of unknown points in the defined space, i.e., $n' = \prod_{d=1}^{D} m_d - n$, $\hat{y}_i$ and $\sigma_i$ are the approximated posterior mean and std. for the $i$th data point, respectively, $f_i$ denotes the true value for the $i$th point, and $\psi(\cdot)$ and $\Phi(\cdot)$ are the PDF (probability density function) and CDF (cumulative distribution function) of standard normal, respectively.

We see that BKTF successfully finds the global optimum with rank from 2 to 8, and obtains better (lower) CRPS values than GP baseline surrogates during the search processes. These results indicate that the proposed fully Bayesian framework is robust to the rank setting and can avoid overfitting.

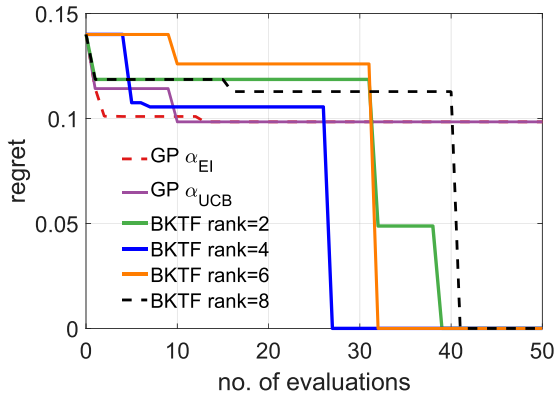



Figure 12: Effects of rank specification on the test function defined in Introduction (see Appendix C, Eq. 23): Comparison of optimization performance.

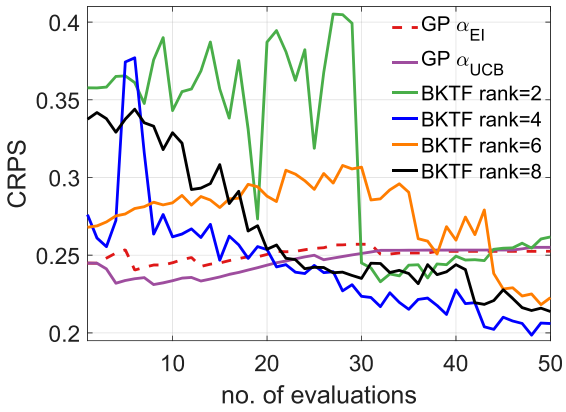



Figure 13: Effects of rank specification on the test function defined in Introduction (see Appendix C, Eq. 23): Comparison of CRPS.

## F Comparison of computational complexity

We compare the computational complexity of BKTF, BKTFrandom and the baseline methods applied on test functions in Table 3, where $n$ is the number of observations, $m_d$ is the number of interpolation points for the $d$th dimension, and $n_{\mathrm{random}}$ denotes the number of candidates we selected in BKTFrandom, which is 20k in the conducted experiments.

As can be seen, theoretically the proposed model BKTF/BKTFrandom has the lowest computational complexity.

Table 3: Comparison of model complexity.

| Model | Complexity |
|---|---|
| GP $\alpha_{\text{EI}}$ | $\mathcal{O}\left(n^3\right)$ |
| GP $\alpha_{\text{UCB}}$ | $\mathcal{O}\left(n^3\right)$ |
| GPgrid $\alpha_{\text{EI}}$ | $\mathcal{O}\left(\left(\prod_{d=1}^{D} m_d\right)^3\right)$ |
| GPgrid $\alpha_{\text{UCB}}$ | $\mathcal{O}\left(\left(\prod_{d=1}^{D} m_d\right)^3\right)$ |
| additive GP | $\mathcal{O}\left(n^3\right)$ |
| BKTF | $\min\left\{\mathcal{O}\left(n^3\right), \mathcal{O}\left(\sum_{d=1}^{D} m_d^3\right)\right\}$ |
| BKTFrandom | $\mathcal{O}\left(n_{\text{random}}^3\right)$ |
| deepGP | $\mathcal{O}\left(\left(\prod_{d=1}^{D} m_d\right)^3\right)$ |

# G Hyperparameter tuning for machine learning

Table 4 lists all the hyperparameters in the tuning tasks.

Table 4: Hyperparameters of the tested ML algorithms.

| Dataset | Algorithm | Hyperparameters | Type | Search space |
|---|---|---|---|---|
| MNIST | RF classifier | no. of estimators | integer | [10, 100] |
| | | max depth | integer | [5, 50] |
| | | max features | integer | [1, 64] |
| | | min samples split | integer | [2, 11] |
| | | min samples leaf | integer | [1,11] |
| | | criterion | categorical | gini, entropy |
| | NN classifier | neurons | integer | [10, 100] |
| | | batch size | integer | [16, 64] |
| | | epochs | integer | [20, 50] |
| | | patience | integer | [3, 20] |
| | | optimizer | categorical | adam, rmsprop, sgd |
| | | activation | categorical | relu, tanh |
| Boston housing | RF regressor | no. of estimators | integer | [10, 100] |
| | | max depth | integer | [5, 50] |
| | | max features | integer | [1, 13] |
| | | min samples split | integer | [2, 11] |
| | NN regressor | neurons | integer | [10, 100] |
| | | batch size | integer | [16, 64] |
| | | epochs | integer | [20, 50] |

# H Supplementary results on ML hyperparameter tuning

We summarize the final accuracy obtained with different BO methods for MNIST classification and the final MSE for the regression task in Table 5.

Table 5: Final accuracy for (a) MNIST classification and MSE for (b) Boston housing regression.

| | (a) MNIST | | (b) Boston housing | |
|---|---|---|---|---|
| | RF classifier | NN classifier | RF regressor | NN regressor |
| RS | $96.09 \pm 0.05$ | $97.59 \pm 0.37$ | $26.10 \pm 0.45$ | $40.43 \pm 3.91$ |
| PSO | $95.84 \pm 0.14$ | $97.54 \pm 0.51$ | $26.07 \pm 0.44$ | $44.40 \pm 6.29$ |
| GP $\alpha_{\mathrm{EI}}$ | $96.10 \pm 0.07$ | $98.04 \pm 0.05$ | $26.19 \pm 0.45$ | $38.46 \pm 3.31$ |
| GP $\alpha_{\mathrm{UCB}}$ | $96.06 \pm 0.05$ | $97.46 \pm 0.64$ | $26.34 \pm 0.35$ | $36.78 \pm 1.91$ |
| BO-TPE | $96.10 \pm 0.06$ | $97.52 \pm 0.06$ | $26.27 \pm 0.31$ | $36.40 \pm 4.72$ |
| BKTF | - | - | $\mathbf{25.03} \pm 0.18$ | $\mathbf{30.84} \pm 1.13$ |
| BKTFrandom | $\mathbf{96.38} \pm 0.04$ | $\mathbf{98.16} \pm 0.09$ | - | - |

The values are presented as mean $\pm$ std. Best results are highlighted in bold fonts.

# NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please see Abstract and Section Introduction for more details.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Section 6 Conclusion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We illustrate methodology and technical details of the proposed model in Section 3 BKTF for Bayesian optimization and Appendix A Model inference.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We illustrate detailed information for experiment implementation in Section 5 Experiments and Appendices C-G.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the code for the 2D test function in supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We specify the experimental setting and implementation details in Section 5 Experiments and Appendices C-G.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We repeat the experiments certain times and report the mean with std. results in Figure 1, 2, 3, and Table 2, 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All the experiments can be performed with a 16-core 2.40 GHz CPU and 32 GB RAM.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper conform with the NeurIPS code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This paper presents work whose goal is to advance the field of probabilistic Machine Learning, particularly Bayesian Optimization. We discussed such impacts in Section 6 Conclusion.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: This work has the potential of misuse for machine learning algorithms. However the current model has certain limitations on applying for high-dimensional problems, thus such risks are low. We mentioned such risks in the last paragraph in Section 6 Conclusion.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We include the URLs for the datasets we used in this work.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We submit partial of the code in supplementary material and select a license when submitting the paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.