MALIBO: META-LEARNING FOR LIKELIHOOD-FREE BAYESIAN OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian Optimization (BO) is a popular method to optimize expensive black-box functions. Typically, BO only uses observations from the current task. Recently proposed methods try to warm-start BO by exploiting knowledge from related tasks, yet suffer from scalability issues and sensitivity to heterogeneous scale across multiple datasets. We propose a novel approach to solve these problems by combining a meta-learning technique and a likelihood-free acquisition function. The meta-learning model simultaneously learns the underlying (task-agnostic) data distribution and a latent feature representation for individual tasks. The likelihood-free BO technique has less stringent assumptions about the problems and works with any classification algorithm, making it computation efficient and robust to different scales across tasks. Finally, gradient boosting is used as a residual model on top to adapt to distribution drifts between new and prior tasks, which might otherwise weaken the usefulness of the meta-learned features. Experiments show that the meta-model learns an effective prior for warm-starting optimization algorithms, while being cheap to evaluate and invariant to changes of scale across different datasets.

1 INTRODUCTION

Bayesian Optimization (BO) is a widely used method to optimize expensive black-box functions (Shahriari et al., 2016), and has been successfully applied in different fields, including automated machine learning (ML) (Hutter et al., 2019). Given small amounts of data, traditional BO using a Gaussian Process (GP) surrogate model and an acquisition function can quickly optimize a black-box function. However, most BO techniques start from scratch for each problem, instead of leveraging information from previous runs for similar tasks to further improve data efficiency.

Different approaches have been proposed in the context of transfer learning (Weiss et al., 2016) and meta-learning (Vanschoren, 2018). Prior knowledge can be used to build informed surrogate models (Schilling et al., 2016; Wistuba et al., 2018; Feurer et al., 2018b; Perrone et al., 2018), to restrict the search space (Perrone et al., 2019), or to warm-start the optimization with configurations that generally score well (Feurer et al., 2014; Salinas et al., 2020)

However, these approaches have three important issues:

(i) GPs scale poorly due to their cubical computational complexity (Rasmussen, 2004).

(ii) The standard BO framework requires a surrogate model with well-calibrated and tractable predictive uncertainty, making it inapplicable to high-dimensional problems that require many observations (Tiao et al., 2021; Song et al., 2022).

(iii) Regression models, including GPs, struggle with different scales and noise levels across tasks, which hurts warm-starting and optimization efficiency (Feurer et al., 2018a).

We seek a meta-learning BO approach without the above limitations, that can effectively transfer knowledge from related tasks and scales to large datasets. We propose a new BO algorithm, inspired by the idea of likelihood-free BO (Bergstra et al., 2011; Tiao et al., 2021; Song et al., 2022), that replaces the surrogate model with a meta-learned classifier which directly balances exploration and exploitation without modeling the objective function, thus elegantly avoiding both the scalability and the scale sensitivity issues simultaneously.

This paper makes the following contributions:

(i) A novel probabilistic meta-learning model that uses Bayesian logistic regression and a probabilistic approach to learn feature representations from prior tasks.

(ii) A scalable BO technique with good anytime performance that combines a meta-learning classifier and a likelihood-free acquisition function.

(iii) Robust adaptation to new tasks by combining the meta-learned classifier with gradient boosting to correct prediction errors.

After discussing related work in Sect.2 and covering the necessary background in Sect. 3, we explain our approach in Sect. 4. We evaluate our method in Sect. 5 and draw conclusions in Sect. 6.

2 Related Work

2.1 Meta-learning Bayesian Optimization

Various methods have aimed to improve the data-efficiency of Bayesian optimization (BO) by leveraging the information of previous observations from similar tasks. They leverage meta-learning (Vanschoren, 2018) or transfer-learning (Weiss et al., 2016) depending on the context, and have been proven effective in various applications (Andrychowicz et al., 2016; Finn et al., 2017). We refer to Vanschoren (2018) for an in-depth overview.

One line of work adapts the initial design to warm-start BO, either by reducing the search space (Perrone et al., 2019) or reusing good configurations from similar tasks, where similarity can be based on hand crafted features (Feurer et al., 2014) or learned with Neural Networks (NNs) (Kim et al., 2017). Alternative approaches estimate the usefulness of a given configuration on both the current and prior tasks, which can be done using heuristics (Wistuba et al., 2015) or learning-based (Volpp et al., 2020) methods. Other approaches use transfer learning to modify the probabilistic surrogate model, for instance using a multi-task GP (Swersky et al., 2013), an additive GP model (Golovin et al., 2017; Marco et al., 2017), or weighted combinations of independent GPs for different tasks (Schilling et al., 2016; Wistuba et al., 2018; Feurer et al., 2018a).

Several methods try to *simultaneously* learn the initial design and modify the surrogate model. Springenberg et al. (2016) apply task-specific embeddings for BO and use a Bayesian NN as the surrogate model, which are computationally expensive and hard to train. Perrone et al. (2018) propose Adaptive Bayesian Linear Regression (ABLR), which uses a NN to learn a shared feature representation across tasks with task-specific BLR layers, allowing better scalability and adaptability. However, both these methods are sensitive to changes in the scale and noise level across datasets. To tackle this, Salinas et al. (2020) propose Gaussian Copula Process Plus Prior (GC3P), which transforms the data response values via the empirical CDF, and fits a NN across all prior tasks. This NN is used to warm-start the optimization and predict the mean for a GP on the target task. Despite its robustness, the use of a GP surrogate still limits its applicability on high-dimensional and large tasks.

Our meta-learning method is more closely related to *Bayesian optimization with NNs and embedding reasoning* (BaNNER, Berkenkamp et al. (2021)). It uses a meta-learning model based on a NN to learn a latent representation and a task-specific BLR layer, similar to ABLR. However, the model output is divided into a task-independent mean and a task-specific residual prediction learned by a BLR layer. In this paper, we introduce a classifier variant of this meta-learning model and combine it with a likelihood-free acquisition function.

2.2 Likelihood-free Bayesian Optimization

Bayesian optimization does not require explicit modeling of the likelihood of the observed values (Garnett, 2022). Tree-Parzen Estimators (TPE, Bergstra et al. (2011)) turn BO into a density ratio estimation problem (Sugiyama et al., 2012), and use the density ratio over 'good' and 'bad' configurations as an acquisition function without a probabilistic surrogate model. Tiao et al. (2021) estimate the density ratio through class probability estimation (Qin, 1998), which is equivalent to modeling the acquisition function with a binary classifier. Likelihood-Free BO (LFBO, Song et al. (2022)) improves upon this by weighting the observations with the magnitude of improvement.

Likelihood-free BO approaches address two drawbacks of traditional GP-based BO methods, namely the computationally expensive inference and the lack of flexibility due to the strong assumptions associated with kernel methods. Rather than modeling the objective function, likelihood-free BO methods use deterministic binary classifiers to separate good and bad configurations, which results in scale-invariant models. This allows the use of any binary classification method (Tiao et al., 2021; Song et al., 2022). We leverage this flexibility to create a new meta-learning classifier, yielding a scalable method that is robust to heterogeneous scales across datasets.

3 Problem Statement and Background

3.1 Bayesian Optimization

Consider optimizing a black-box function $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ over a set of input $\boldsymbol{x} \in \mathcal{X}$. At each step n, we can choose a function input $\boldsymbol{x}$ and obtain a noisy observation $y_n = f(\boldsymbol{x}_n) + \epsilon_n$, with noise ϵ_n. The goal of Bayesian Optimization (BO) is to find the optimum $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$ efficiently by using a set of input-output observations pairs collected up to step N, $\mathcal{D}_N = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$. Given a probabilistic surrogate model $\mathcal{M}$, BO algorithms use its posterior prediction $p(y|\boldsymbol{x}, \mathcal{D}_N)$ to compute the acquisition function $\alpha(\boldsymbol{x}; \mathcal{D}_N)$, which balances exploration and exploitation. The next candidate is obtained by maximizing the selected acquisition function: $\boldsymbol{x}_{N+1} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \alpha(\boldsymbol{x}; \mathcal{D}_N)$.

The acquisition function is typically defined as an expected utility function, where a utility function $U(y; \tau)$ with threshold τ heuristically balances exploration and exploitation. For example, the prevalent Expected Improvement (EI, Močkus (1975)) acquisition function has $U(y; \tau) := \max(\tau - y, 0)$ while the Probability of Improvement (PI, Kushner (1964)) has $U(y; \tau) := \mathbb{1}(\tau - y > 0)$. The expected utility over the posterior belief from the surrogate model $p(y|\boldsymbol{x}, \mathcal{D}_N)$ reads

$$\alpha^U(\boldsymbol{x}; \mathcal{D}_N, \tau) = \mathbb{E}_{y \sim p(y|\boldsymbol{x}, \mathcal{D}_N)}[U(y; \tau)] = \int U(y; \tau) p(y|\boldsymbol{x}, \mathcal{D}_N) dy \tag{1}$$

where the threshold is normally chosen to be the lowest observed function value, i.e. $\tau = \min y_n$.

3.2 Likelihood-free Acquisition Functions

Likelihood-free acquisition functions intuitively model the belief of a candidate being promising. Instead of modeling the outcomes explicitly via a surrogate model $p(y|\boldsymbol{x}, \mathcal{D}_N)$, Tree Parzen Estimators (TPE) (Bergstra et al., 2011) model two independent density distributions over the best and worst inputs, namely $\ell(\boldsymbol{x}) = p(\boldsymbol{x}|y \leq \tau, \mathcal{D}_N)$ and $g(\boldsymbol{x}) = p(\boldsymbol{x}|y > \tau, \mathcal{D}_N)$. The density ratio (DR) serves as the acquisition function α: $\alpha^{\text{DR}}(\boldsymbol{x}; \mathcal{D}_N, \tau) = \ell(\boldsymbol{x})/g(\boldsymbol{x})$. As discussed by Tiao et al. (2021), this approach suffers from several drawbacks due to solving the more challenging problem of density estimation as an intermediate step to the *easier* problem of estimating the ratio directly.

To fix these issues, Tiao et al. (2021) propose BORE, where the density-ratio is directly estimated by using class probability estimation. Theoretically, BORE can work with any classifier with probabilistic outputs, which enables us to use more robust and expressive algorithms. Let $\gamma = p(y \leq \tau|\mathcal{D}_N)$ be the proportion of observations with top performing y values, and k the binary class label $\mathbb{1}(y \leq \tau)$. BORE estimates the class label k for a given input $\boldsymbol{x}$ by a classifier $C_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$. They show that the prediction of $C_{\boldsymbol{\theta}}$ is proportional to the density ratio, $\alpha^{\text{DR}}(\boldsymbol{x}; \mathcal{D}_N, \tau) \propto C_{\boldsymbol{\theta}}(\boldsymbol{x})$, if one minimizes a proper scoring rule (Gneiting & Raftery, 2007) w.r.t. $\boldsymbol{\theta}$ to recover the true class-posterior probabilities:

$$\mathcal{L}^{\text{BORE}} = -\frac{1}{N} \sum_{n=1}^{N} \left(k_n \log C_{\boldsymbol{\theta}}(\boldsymbol{x}_n) + (1 - k_n) \log(1 - C_{\boldsymbol{\theta}}(\boldsymbol{x}_n)) \right) \tag{2}$$

Although Tiao et al. (2021) argue that BORE resembles EI, assigning the same label to all observations with $y < \tau$ regardless of the magnitude of improvement over the threshold conforms to the definition of PI rather than EI, and might lead to conservative optimization with little global exploration (Garnett, 2022; Song et al., 2022). In practice, EI has proven to have stronger performance than PI. Song et al. (2022) show how to convert the expected utility with EI into an optimization problem using a variational representation. The solution is reformulated as a classification problem

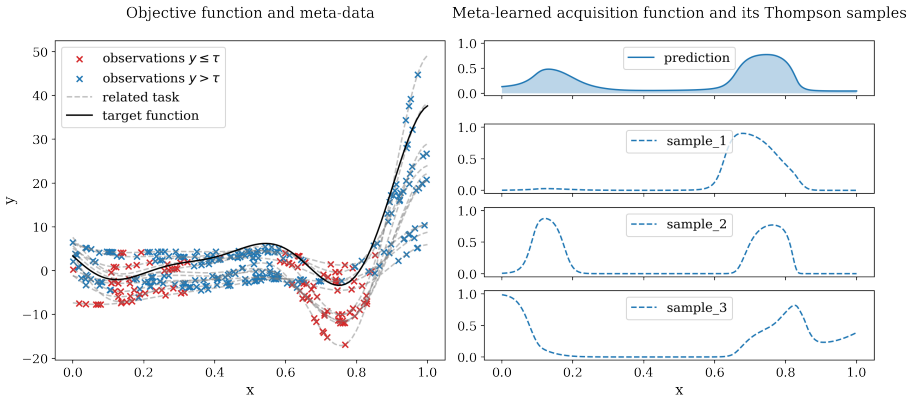


Figure 1: Illustration of meta-learning the acquisition function. Left: Observations (crosses) from 10 related tasks and the target task. The top performing observations ($\gamma = 1/3$) in each task are shown in red, otherwise in blue. Right: The top panel shows the approximated predictive distribution (see Eq. 7) while the others show Thompson samples. MALIBO successfully identifies the promising areas, and the Thompson samples show variability in the meta-learned acquisition function.

with the following objective for EI:

$$\begin{aligned}\mathcal{L}^{\text{LFBO}} &= -\mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y|\mathcal{D}_N)}[\max(y-\tau,0)\log C_{\boldsymbol{\theta}}(\boldsymbol{x}) + \log(1-C_{\boldsymbol{\theta}}(\boldsymbol{x}))] \\ &= -\gamma\,\mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y|y\leq\tau,\mathcal{D}_N)}[(y-\tau)\log C_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y|\mathcal{D}_N)}[\log(1-C_{\boldsymbol{\theta}}(\boldsymbol{x}))]\end{aligned} \tag{3}$$

The resulting method, dubbed Likelihood-Free Bayesian Optimization (LFBO), can be seen as a weighted classification problem with noisy targets for only class $k = 1$, where the EI utility function $\max(y - \tau)$ weights the importance for observations below τ by their improvement. The minimizer of Equation 3 is shown to be equivalent to the EI acquisition function (Song et al., 2022).

4 Methodology

In this section we introduce our MetA-learning for LIkelihood-free BO method, dubbed MALIBO. It incorporates a probabilistic meta-learning approach derived from BaNNER into the LFBO framework. In particular, we show how to convert the regression model in BaNNER into a meta-learning classifier that warm-starts BO and can be integrated into gradient boosting (Friedman, 2001), yielding good anytime performance, more flexibility in selecting the model, and better adaptation to new tasks. We first explain the different components and then summarize our algorithm in Algorithm 1.

4.1 Meta-learning

The goal of meta-learning for optimization is to extract information from previous tasks and use this to accelerate the optimization of a new one. As in all methods discussed in Section 2, we only consider the case of identical input spaces $\mathcal{X}$ across all tasks, which simplifies the learning problem. Our goal is to learn a probabilistic model for the likelihood-free acquisition function from the meta-data, as illustrated in Figure 1 for a one-dimensional function.

Following the approach from Berkenkamp et al. (2021) and Perrone et al. (2018), our meta-learning classifier uses a deterministic, task-agnostic model to map the input space into a feature space, $h(\boldsymbol{x}) = \boldsymbol{\phi}$ of a predefined dimensionality. In this feature space, we seek to find a simple model that is able to classify the data on all prior tasks using a task specific adaptation. We propose to use Bayesian logistic regression with a minor modification: instead of using a linear model directly on $\boldsymbol{\phi}$, we separate out a task-agnostic mean function $m(\boldsymbol{\phi})$ and a task-specific linear model $\boldsymbol{\phi} \cdot \boldsymbol{z}_t$, where $\boldsymbol{z}_t$ is a numerical representation of the prior task t in the learned feature space. The sum of those two serves as the input to a sigmoid function σ, converting it into class probabilities. An overview of the method is demonstrated in Figure 2.

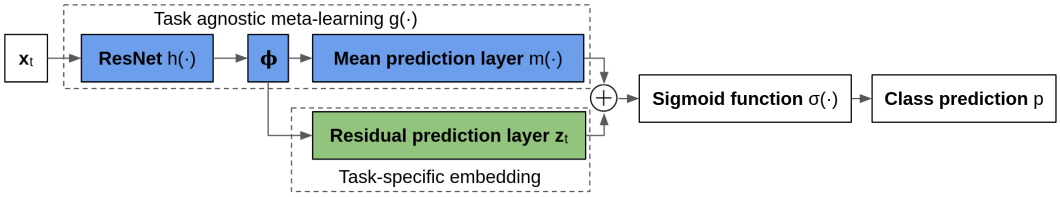


Figure 2: Schematic representation of our meta-learning classifier. A ResNet (He et al., 2016) maps the input x_t to a latent feature representation $\boldsymbol{\phi}$. From this, a global mean prediction $m(\boldsymbol{\phi})$ and task-specific embedding $\boldsymbol{z}_t$ are combined and translated into a class prediction via a sigmoid function.

During training, the meta-learning model $g(\cdot) = m(h(\boldsymbol{x}))$ learns a global prior for the classification problem, and the task-specific embedding vectors $\boldsymbol{z}_t$ represent the necessary adaptation. For better inference of the new task's $\boldsymbol{z}$ values, their distribution is regularized during training. We follow the technique used by Berkenkamp et al. (2021) and Saseendran et al. (2021) to bring $\boldsymbol{z}_t$ close to the prior distribution $p(\mathcal{Z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, using a modified Kolmogorov-Smirnov test and the covariance to calculate the disparity of two distributions. The loss used for training on the meta-data reads:

$$\mathcal{L}^{\text{meta}} = \min_{g(\cdot), \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T} \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}^{\text{LFBO}}(k_t; g(\boldsymbol{x}_t), \boldsymbol{z}_t) + \lambda \mathcal{R}(\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T\}; p(\mathcal{Z}))\,, \tag{4}$$

where we average the contribution of all T prior-tasks, $\mathcal{R}$ is a regularization term outlined by Berkenkamp et al. (2021) weighted by λ.

4.2 Task adaptation via Bayesian logistic regression

After training the meta-model, we need to adapt our predictions to each new task. Hence, a value for $\boldsymbol{z}$ must be estimated that yields good classification results. While this could in principle be done by greedily optimizing Equation 3 using $C(\boldsymbol{x}) = \sigma(g(\boldsymbol{x}) + \boldsymbol{z} \cdot h(\boldsymbol{x}))$, we follow a Bayesian approach, Bayesian logistic regression in particular, to cover the uncertainty in the task-specific $\boldsymbol{z}$. Especially with few observations on the new task, using a single $\boldsymbol{z}$ vector can introduce a strong bias.

Besides stronger exploration in the early phases of optimization, the Bayesian approach allows us to employ Thompson sampling (Thompson, 1933) for $\boldsymbol{z}$, as shown in Figure 1. We believe this to be a valuable strategy for parallelization which we do not explore here. Kandasamy et al. (2018) showed that this bypasses the sequential scheme of traditional BO, without introducing the common computational burden of more sophisticated methods. See Garnett (2022) for more details.

Bayesian logistic regression is a simple yet powerful classification model with Bayesian treatment. Although exact Bayesian inference for logistic regression is intractable, the Laplacian and probit approximation can be applied to approximate the weights' posterior and the predictive distribution respectively while remaining reliable and scalable (Bishop & Nasrabadi, 2006; Murphy, 2012). The Laplacian method approximates the posterior by fitting a Gaussian distribution around the maximum-a-posteriori (MAP) estimate of the weights distribution, and matching the second order derivative near the optimum.

In the first step, we obtain the MAP estimate by maximizing the posterior of our classifier C parameterized by $\boldsymbol{z}$. To be consistent with the regularization used during meta-training, we use a zero mean isotropic Gaussian as the prior weight distribution $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{m}_0, \boldsymbol{\Sigma}_0)$, with mean $\mathbf{m}_0 = \mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}_0 = \mathbf{I}$. Given observations $\mathcal{D}_N$, the posterior distribution over $\boldsymbol{z}$ is proportional to $p(\boldsymbol{z}|\mathcal{D}_N) \propto p(\boldsymbol{z})p(\mathcal{D}_N|\boldsymbol{z})$. We can maximize the log posterior likelihood via optimizing the following objective w.r.t. $\boldsymbol{z}$:

$$\mathcal{L}^{\text{MALIBO}} = \frac{1}{2}(\boldsymbol{z} - \mathbf{m}_0)^T \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{z} - \mathbf{m}_0) + \sum_{n=1}^{N} \left(k_n (y - \tau) \log C(\boldsymbol{x}_n) + \log(1 - C(\boldsymbol{x}_n))\right)\,, \tag{5}$$

where our class probability prediction is given by the output of a sigmoid function: $\hat{k} = \sigma(\boldsymbol{z}^T \boldsymbol{\phi} + m(\boldsymbol{\phi}))$, $m(\boldsymbol{\phi})$ is the mean predictions with given input features $\boldsymbol{\phi}$ and $\boldsymbol{z}^T \boldsymbol{\phi}$ is the residual prediction[1]. The MAP estimate of the weights is therefore $\boldsymbol{z}_{\text{MAP}} = \arg\min_{\boldsymbol{z} \in \mathcal{Z}} \mathcal{L}^{\text{MALIBO}}$.

[1]This is similar to meta-training, the only difference is we only learn $\boldsymbol{z}$ with task agnostic model $g(\cdot)$ fixed.

Algorithm 1: MALIBO: Meta-learning for likelihood-free Bayesian optimization

Meta-learning:
- **Input:** Meta-datasets $\mathcal{D}_t^{\text{meta}}$ for tasks $t = 1, \dots, T$, $\gamma = p(y \leq \tau | \mathcal{D}^{\text{meta}})$
- Generate binary class labels for meta-data using γ: $\mathbb{1}(y \leq \Phi^{-1}(\gamma))$
- Learn the meta-learning model $C(\cdot; \mathcal{D}^{\text{meta}})$ by optimizing $\mathcal{L}^{\text{meta}}$ (Equation 4)

Bayesian optimization with Meta-learning:
- **Input:** Fixed meta-learned model $g(\cdot)$
- Obtain the first best guess $\boldsymbol{x}_0$ from the meta-learned model $\mathcal{D} = \{(\boldsymbol{x}_0, f(\boldsymbol{x}_0) + \epsilon)\}$
- For the gradient boosting variant, use $C(\cdot) \leftarrow GB(C(\cdot))$ as the classifier
- **while** *Has budget* **do**
  - Estimate $\boldsymbol{z}_{\text{MAP}}$ by optimizing $\mathcal{L}^{\text{MALIBO}}$ (Equation 5) w.r.t. $\boldsymbol{z}$
  - Update precision matrix $\boldsymbol{\Sigma}_n^{-1}$ (Equation 6)
  - **if** *Thompson sampling* **then**
    - Sample $\hat{\boldsymbol{z}} \sim \text{MVN}(\boldsymbol{z}_{\text{MAP}}, \boldsymbol{\Sigma})$
    - $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} C(\boldsymbol{x}; \hat{\boldsymbol{z}}, g(\boldsymbol{x}), \mathcal{D})$
  - **else**
    - $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} C(\boldsymbol{x}; g(\boldsymbol{x}), \mathcal{D})$ with probit approximation (Equation 13)
  - $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}^*, f(\boldsymbol{x}^*) + \epsilon)\}$

For the Laplace approximation, the negative Hessian of the log posterior

$$\boldsymbol{\Sigma}_N^{-1} = -\nabla\nabla \ln p(\boldsymbol{z}|\mathcal{D}_N) = \boldsymbol{\Sigma}_0^{-1} + \sum_{n=1}^{N} \hat{k}_n (1 - \hat{k}_n) \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T \tag{6}$$

is used as the inverse covariance matrix of the approximated posterior $q(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{z}_{\text{MAP}}, \boldsymbol{\Sigma}_N)$.

To finally compute the approximate predictive distribution, we need to marginalize w.r.t. $p(\boldsymbol{z}|\mathcal{D}_N)$:

$$C(\boldsymbol{x}; g(\boldsymbol{x}), \mathcal{D}_N) = \int p(k = 1 | g(\boldsymbol{x}), \boldsymbol{z}) p(\boldsymbol{z}|\mathcal{D}_N) d\boldsymbol{z} \simeq \int \sigma(\boldsymbol{z}^T \boldsymbol{\phi} + m(\boldsymbol{\phi})) q(\boldsymbol{z}) d\boldsymbol{z} \tag{7}$$

Although no closed form solution for the integral in Equation 7 is available due to the logistic activation, we can evaluate the predictions with probit approximation, which makes use of the similarity of the sigmoid function and the probit function. See Appendix A for further details.

4.3 Gradient boosting as a residual prediction model

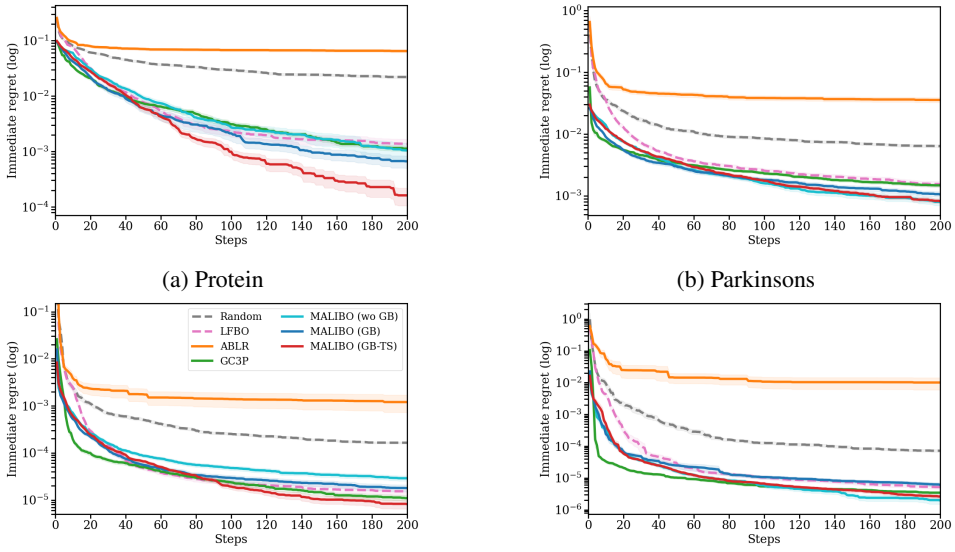
While the optimization is warm-started by learning the feature representation from the meta-data, the model will gain more knowledge about the new task as more data is acquired, and thus the importance of the meta model will diminish. In addition, the distribution of the (new) data obtained from optimization can differ from that of the prior meta-data, especially in the later phases of optimization. While we do update the Bayesian logistic regression model with every new observation, we want to ensure that our method can always flexibly adapt to the new task, even if, for instance, the meta-learned representation does not work optimally for the new task.

To this end, we propose to combine our method with gradient boosting (Friedman, 2001). In every iteration, besides updating the Bayesian logistic model, we train new gradient boosting models (regression trees) with collected $(\mathbf{x}, y)$ pairs on top to correct the remaining errors of the ensemble. Therefore, gradient boosting will correct any mistakes of our initial model and finetune it to the new task, especially later in the optimization process.

5 Experiments

In this section, we describe the experiments conducted to empirically evaluate our method. For the choice of problems, we focus on automated machine learning (AutoML), i. e. hyperparameter optimization (HPO) and neural architecture search (NAS). Besides that, we evaluate our methods on synthetic functions to show the robustness against multiplicative noise.

We compare our method against state-of-the-art baselines across all problems, namely: random search (Bergstra & Bengio, 2012), gradient boosting variant of LFBO (Song et al., 2022),



(a) Protein (b) Parkinsons

(c) Slice (d) Naval

Figure 3: Immediate regret for different BO algorithms on the HPOBench neural network tuning problems ($D = 9$) for 4 datasets.

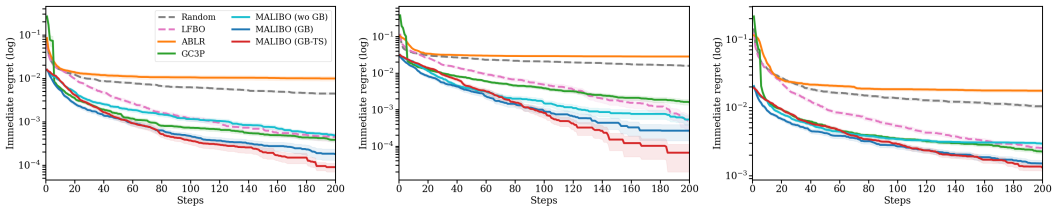
ABLR (Perrone et al., 2018) and GC3P (Salinas et al., 2020). In addition, we consider the performance of our method without gradient boosting (wo GB), with gradient boosting only (GB) and with both gradient boosting and Thompson sampling (GB-TS).

We use *immediate regret* to quantitatively measure performance of different methods, which by definition is the absolute error between the global minimum and the lowest function value obtained so far. For all benchmarks, we report the results by mean and standard error across 100 random runs.

We implemented ABLR and MALIBO in PyTorch (Paszke et al., 2019), and used scikit-learn (Pedregosa et al., 2011) for gradient boosting. For GC3P, we use the benchmarking functionality from its open source implementation[2]. GC3P samples five from a meta-learned NN model before building a task-specific Coppola process while LFBO randomly samples 10 observaitons for gathering global information from the target task. Thanks to the meta-leared acquisition function, MALIBO starts the optimization at the point with highest acquisition function value. For all likelihood-free BO methods required hyperparameter γ, i. e. LFBO, MALIBO and its variants, we set $\gamma = 1/3$, following Tiao et al. (2021) and Song et al. (2022).

Neural network tuning (HPOBench) This benchmarks represents a joint NAS and HPO problem: optimizing the mean squared errors of a two-layer feed-forward regression network on four poular UCI datasets (Dua & Graff, 2017). The search space of the problem is 9 dimensional and is available as a tabular benchmark (Klein & Hutter, 2019; Eggensperger et al., 2021) with a total of 62,208 unique configurations. Additional details about the benchmark are included in Appendix D.1. For meta-learning methods, we randomly sampled 512 configurations from each of three other datasets as meta-data. In Figure 3, we see across all benchmarks all MALIBO variants show the best warm-starting performance among all methods. Meanwhile, both gradient boosting MALIBO guarantees the best end performance in all benchmarks, especially in protein and parkinsons experiments. The Thompson sampling variant helps the model to explore the space first, resulting slightly slower convergence speed at the beginning but better end results. GC3P is a very competitive method often being the best method after some initial steps. Throughout the 4 datasets, ABLR performs very poorly, presumably because of the small number of prior-tasks and the fact that the meta-training seems have driven the shared features into a local optima that does not fit the respective target task.

[2]https://github.com/geoalgo/A-Quantile-based-Approach-for-Hyperparameter-Transfer-Learning

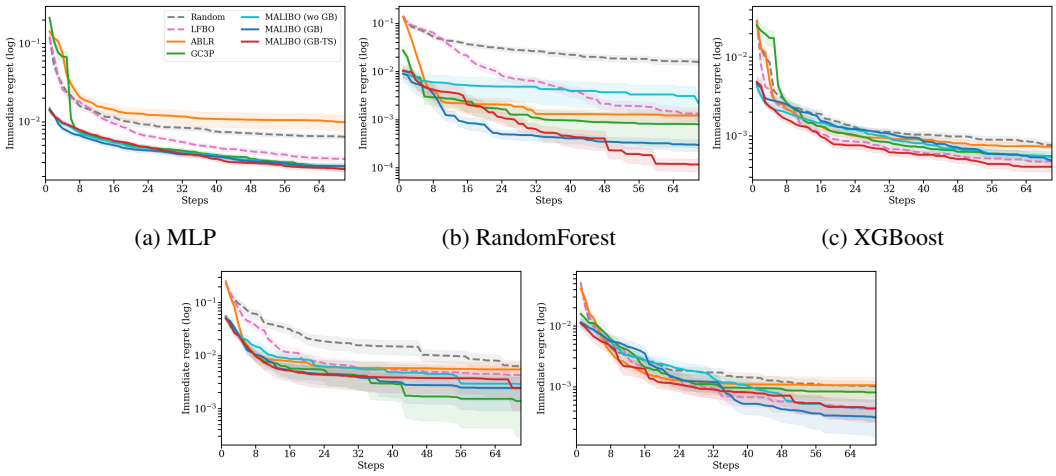


(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet-16

Figure 4: Immediate regret for different BO algorithms on the NASBench201 neural architecture search problems ($D = 6$) on 3 datasets.

Neural architecture search (NASBench201) NASBench201 (Dong & Yang, 2020) considers the problem of designing a neural cell with 6 discrete parameters and a total search space size of $15,625$ architectures, evaluated on three datasets: CIFAR-10, CIFAR-100 (Krizhevsky, 2009) and ImageNet-16 (Chrabaszcz et al., 2017). We refer to Appendix D.3 for more details. We randomly sampled 512 configurations from each of two other datasets for meta-training. From the results shown in Figure 4, we see that our methods start again with the lowest average regrets, and the gradient boosting variants of MALIBO outperforms all baselines across all datasets with rapid convergence. Although the one with Thompson sampling shows slightly slower convergence rate, it improves the end performance even further. GC3P remains a strong competitor, but seems to converge prematurely. ABLR again suffers from few prior-tasks.

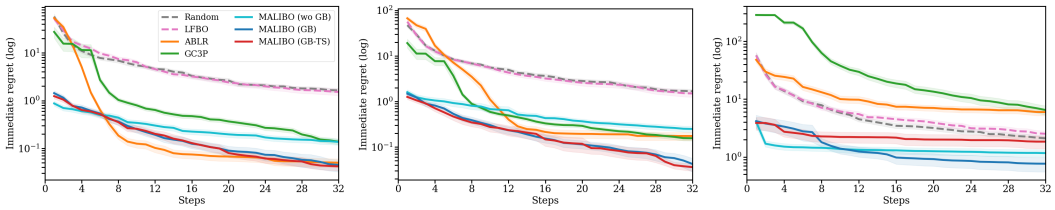
ML algorithms tuning (MLBench) This ML benchmark suite provided by Eggensperger et al. (2021) includes HPO tasks including 5 algorithms (SVM, LogReg, XGBoost, RandomForest and MLP) evaluated on 20 OpenML tasks (Vanschoren et al., 2014), except for MLP with only 8 tasks. The search spaces dimensions range from 2 (SVM) to 5 (MLP). We refer to Appendix D.2 for more details. For meta-data, we randomly sampled 128 configurations from each task except the current target one (which changed across the repetitions). As seen in Figure 5, all MALIBO variants continue to demonstrate strong warm-starting performance, especially for RandomForest and MLP. Besides, they show similar performance on all benchmarks, except for RandomForest, where gradient boosting variants are better. For these larger number of prior-tasks, ABLR finally shows its potential and is competitive on some benchmarks, but fails to converge on most of them. The warm-starting of GC3P fluctuates from being on-par with MALIBO (SVM, LogReg) to being essentially random guesses (MLP, XGBoost). The method remains very strong throughout all benchmarks.



(a) MLP (b) RandomForest (c) XGBoost

(d) SVM (e) LogReg

Figure 5: Immediate regret for different BO algorithms on the HPOBench hyperparamter tuning problems for 5 different machine learning algorithms.



(a) Branin ($\epsilon = 0.0$) (b) Branin ($\epsilon = 0.1$) (c) Branin ($\epsilon = 1.0$)

Figure 6: Immediate regret for different BO algorithms on Branin function ensembles ($D = 2$) with different levels of multiplicative noise.

Robustness against different noise level To test the robustness of our method against different scale and noise level in the data, we use synthetic function ensembles introduced by Berkenkamp et al. (2021) with multiplicative noise. We focus on the Branin function ensmeble (Dixon, 1978), which is a two-dimensional problem with 3 local minima. Their location and the global minimum varies across different functions in the ensemble. See Appendix D.5 for more details.

To avoid biasing this experiment towards a single method, we decided to use a heteroscedastic noise that is incompatible with any assumptions about the noise of any method. In particular, this violates ABLR's assumption of homoscedastic, Gaussian noise. GC3P makes a similar assumption, but after the nonlinear transformation applied to the y-values, which does not translate to any particular noise model. LFBO and by extension all MALIBO variants, do not make any explicit noise assumptions, but will optimize for the best mean. We choose a multiplicative noise, i.e. $y = f(x) \cdot (1 + \epsilon \cdot n)$, where $n \sim \mathcal{N}(0, 1)$. To test the robustness, we choose the $\epsilon = \{0.1, 1.0\}$, whose noise level will change according to the magnitude of the function values. It dramatically increases the noise level for observations with large values, while having smaller effect on those with lower function values, e. g. the local minima. For meta-training, we randomly samples 128 noisy observations from 256 functions in the ensemble as meta-data. We show our results in Figure 6, where we can see across all magnitudes of multiplicative noise, our method still learns a meaningful prior for the optimization. As the level of noise increase, all MALIBO variants and LFBO remain similar performance, while the performance of GP-based methods, i. e. ABLR and GC3P, has dramatically degraded.

6 Conclusion

We introduced MetA-learning for LIkelihood-free BO (MALIBO), a novel meta-learning optimization algorithm that is computationally efficient and robust to varying scales of the observations and heterogeneous noise. By directly modeling the acquisition function from observations, the method makes fewer assumptions about the data and noise distributions. Coupled with meta-learning, it leverages information from prior tasks for more sample efficiency. To ensure adaption to new tasks, possibly different from prior ones, we incorporate our model into gradient boosting, transitioning from a meta-learning driven model towards a specialized one on the current task. Empirical results demonstrate superior performance on both HPO and NAS benchmarks, as well as synthetic benchmarks with heteroscedastic noise.

Despite the promising experimental results, some limitations of the method should be noted. As discussed by Tiao et al. (2021), the threshold parameter τ in likelihood-free BO algorithms, which controls the exploitation and exploration trade-off, should be treated more carefully. One might consider probabilistic treatment for this hyperparameter. We also observed over-confident predictions in our experiments, for example the Forrester ensemble in Appendix B, where not all runs find the true optimum. This is especially important for problems with more complicated loss landscapes, but mitigation strategies exist.

Directions for future work include extensions to parallel BO with Thompson sampling (Kandasamy et al., 2018), multi-fidelity optimization for HPO problems (Falkner et al., 2018), multi-objective optimization (Hernandez-Lobato et al., 2016), BO with automatic stopping (Makarova et al., 2022) and Bayesian deep active learning (Gal et al., 2017).

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 3988–3996. Curran Associates Inc., 2016.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, 2012.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

Felix Berkenkamp, Anna Eivazi, Lukas Grossberger, Kathrin Skubch, Jonathan Spitz, Christian Daniel, and Stefan Falkner. Probabilistic meta-learning for bayesian optimization, 2021.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Springer, 2006.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets, 2017.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Laurence Charles Ward Dixon. The global optimization problem. an introduction. *Toward global optimization*, 2:1–15, 1978.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, Rene Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1436–1445, 2018.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Using meta-learning to initialize bayesian optimization of hyperparameters. In *Proceedings of the 2014 International Conference on Meta-Learning and Algorithm Selection - Volume 1201*, MLAS'14, pp. 3–10, Aachen, DEU, 2014. CEUR-WS.org.

Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *ICML 2018 AutoML Workshop*, 2018a.

Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. Practical transfer learning for bayesian optimization, 2018b.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 06–11 Aug 2017.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1183–1192. PMLR, 2017.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2022.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pp. 1487–1495. Association for Computing Machinery, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Daniel Hernandez-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1492–1501. PMLR, 20–22 Jun 2016.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer Publishing Company, Incorporated, 2019.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Parallelised bayesian optimisation via thompson sampling. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 133–142. PMLR, 2018.

Jungtaek Kim, Saehoon Kim, and Seungjin Choi. Learning to warm-start bayesian hyperparameter optimization, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Aaron Klein and Frank Hutter. Tabular benchmarks for joint architecture and hyperparameter optimization, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. pp. 32–33, 2009.

H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86(1):97–106, 03 1964.

Anastasia Makarova, Huibin Shen, Valerio Perrone, Aaron Klein, Jean Baptiste Faddoul, Andreas Krause, Matthias Seeger, and Cedric Archambeau. Automatic termination for hyperparameter optimization. In *First Conference on Automated Machine Learning (Main Track)*, 2022.

Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P. Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1557–1563. IEEE Press, 2017.

J. Močkus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pp. 400–404. Springer Berlin Heidelberg, 1975.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cedric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Valerio Perrone, Huibin Shen, Matthias Seeger, Cédric Archambeau, and Rodolphe Jenatton. Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2019.

Jing Qin. Inferences for case-control and semiparametric two-sample density ratio models. *Biometrika*, 85(3):619–630, 1998.

Carl Edward Rasmussen. *Gaussian Processes in Machine Learning*. Springer Berlin Heidelberg, 2004.

David Salinas, Huibin Shen, and Valerio Perrone. A quantile-based approach for hyperparameter transfer learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8438–8448. PMLR, 2020.

Amrutha Saseendran, Kathrin Skubch, Stefan Falkner, and Margret Keuper. Shape your space: A gaussian mixture regularization approach to deterministic autoencoders. In *Advances in Neural Information Processing Systems*, volume 34, pp. 7319–7332. Curran Associates, Inc., 2021.

Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Scalable hyperparameter optimization with products of gaussian process experts. In *ECML/PKDD*, 2016.

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, 2016.

András Sobester, Alexander Forrester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

Jiaming Song, Lantao Yu, Willie Neiswanger, and Stefano Ermon. A general recipe for likelihood-free Bayesian optimization. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 20384–20404. PMLR, 2022.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Louis C Tiao, Aaron Klein, Matthias W Seeger, Edwin V. Bonilla, Cedric Archambeau, and Fabio Ramos. Bore: Bayesian optimization by density-ratio estimation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 10289–10300. PMLR, 2021.

Joaquin Vanschoren. Meta-learning: A survey, 2018.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explor. Newsl.*, 15(2):49–60, jun 2014.

Michael Volpp, Lukas P. Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. Meta-learning acquisition functions for transfer learning in bayesian optimization. In *International Conference on Learning Representations*, 2020.

Karl R. Weiss, Taghi M. Khoshgoftaar, and Dingding Wang. A survey of transfer learning. *J. Big Data*, 3:9, 2016.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–10, 2015.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Mach. Learn.*, 107(1):43–78, 2018.

A PROBIT APPROXIMATION

Let's set $a = \boldsymbol{z}^T\boldsymbol{\phi} + m(\boldsymbol{\phi})$, and the distribution of a would be a Gaussian $\mathcal{N}(a|\mu_a, \sigma_a)$, where μ_a and σ_a are respectively:

$$\mu_a = \mathbb{E}[a] = \int p(a)a\,da = \int q(\boldsymbol{z})(\boldsymbol{z}^T\boldsymbol{\phi} + m(\boldsymbol{\phi}))\,d\boldsymbol{z} = \boldsymbol{z}_{\text{MAP}}^T\boldsymbol{\phi} + m(\boldsymbol{\phi}) \tag{8}$$

$$\begin{aligned}\sigma_a^2 = \text{var}[a] &= \int p(a)\{a^2 - \mathbb{E}[a]^2\}\,da \\ &= \int q(\boldsymbol{z})\{(\boldsymbol{z}^T\boldsymbol{\phi} + m(\boldsymbol{\phi}))^2 - (\mathbf{m}_N^T\boldsymbol{\phi} + m(\boldsymbol{\phi}))^2\}\,d\boldsymbol{z} \\ &= \boldsymbol{\phi}^T\boldsymbol{\Sigma}_N\boldsymbol{\phi}\end{aligned} \tag{9}$$

Thus our approximation to the predictive distribution becomes:

$$p(k = 1|g(\boldsymbol{x}), \mathcal{D}_N) = \int \sigma(a)\mathcal{N}(a|\mu_a, \sigma_a^2)\,da \tag{10}$$

To evaluate the integral in Equation 10, we can obtain a good approximation by making use of the close similarity between the logistic sigmoid function $\sigma(a)$ and the probit function, which is given by the cumulative distribution of the standard Gaussian $\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta|0, 1)\,d\theta$:

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu, \sigma^2)\,da = \Phi\left(\frac{\mu}{(\lambda^{-2} + \sigma^2)^{1/2}}\right) \tag{11}$$

We apply the approximation $\sigma(a) \simeq \Phi(\lambda a)$ to the probit functions appearing on both sides of the equation:

$$\int \sigma(a)\mathcal{N}(a|\mu, \sigma^2)\,da \simeq \sigma((1 + \pi\sigma^2/8)^{-1/2}\mu) \tag{12}$$

Therefore we obtain predictive distribution in the form:

$$p(k = 1|g(\boldsymbol{x}), \mathcal{D}_N) \simeq \sigma((1 + \pi\sigma_a^2/8)^{-1/2}\mu_a) \tag{13}$$

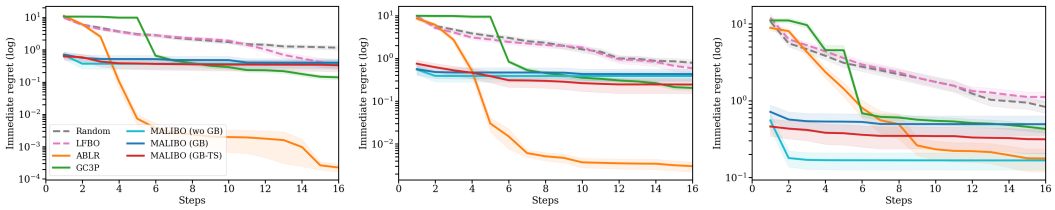
where μ_a is Equation 8 and σ_a^2 is Equation 9.

B ADDITIONAL BENCHMARKS

In this section we show more results for experiments with multiplicative noise as in Section 5. With the settings remaining the same, we perform our experiments on Forester (Sobester et al., 2008) and Hartmann3 (Dixon, 1978) function ensembles. We refer to Appendix D.4 and D.6 respectively for more details.

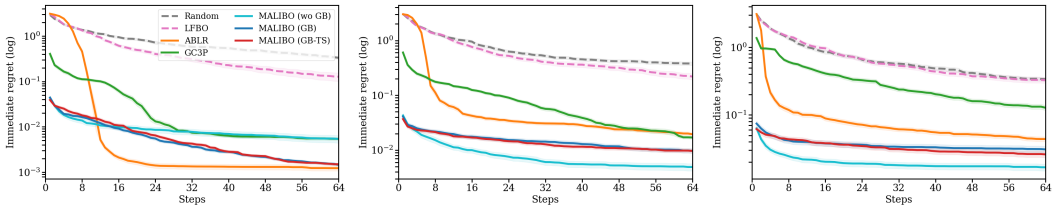
Forrester function ensemble For meta-training in Forester ensemble experiment, we randomly sampled noisy 32 observations in 64 prior-tasks. The results is shown in Figure 7. MALIBO and its variants keep showing strong warm-starting performance and stay robust to noise compared to other methods. However, all the likelihood-free BO methods, namely LFBO, MALIBO and its variants, seem to stuck in local minimum in some runs, resulting in almost no improvement over the optimization process. The performance of all MALIBO variants is on-par with GC3P in all cases while ABLR remain the best method.

Hartmann3 function ensemble We randomly sampled noisy 512 observations in 256 prior-tasks in the Hartmann3 ensemble experiment. The results is shown in Figure 8. All MALIBO variants shows the strongest meta-learning performance in all settings, and the gradient boosting variants report the lowest regrets together with ABLR in noise-free function. All MALIBO variants remain robust to noise, especially the vanilla MALIBO, while the performance of the gradient boosting variants only slightly degraded under noise and stay stable.



(a) Forrester ($\epsilon = 0.0$) (b) Forrester ($\epsilon = 0.1$) (c) Forrester ($\epsilon = 1.0$)

Figure 7: Immediate regret for different BO algorithms on Forrester function ensembles ($D = 1$) with different levels of multiplicative noise.



(a) Hartmann3 ($\epsilon = 0.0$) (b) Hartmann3 ($\epsilon = 0.1$) (c) Hartmann3 ($\epsilon = 1.0$)

Figure 8: Immediate regret for different BO algorithms on Hartmann3 function ensembles ($D = 3$) with different levels of multiplicative noise.

C EXPERIMENTAL DETAILS

We use a ResNet (He et al., 2016) to learn the latent feature representation, with 4 hidden layers, each with 64 units. For the mean layer $m(\cdot)$ and task embedding layer $\boldsymbol{z}$, we use a fully connected layer with 50 units for each. We use ELU (Clevert et al., 2016) as activation function for our problem following Tiao et al. (2021).

During meta-training, we optimize the weights with ADAM (Kingma & Ba, 2015) using batch size of $B = 256$, and polynomial decay for learning rate, with the initial learning rate $\text{lr}_{\text{initial}} = 10^{-3}$, end learning rate $\text{lr}_{\text{final}} = 2^{-4}$ and the exponent set to 2. The model is trained for 2048 epochs with early stopping. We set the regularization factor $\lambda = 0.1$ in Equation 4 and follow the approach in (Berkenkamp et al., 2021) to estimate the weights for modified Kolmogorov-Smirnov test and covariance regularization. In task adaptation, we optimize the task embedding with L-BFGS (Byrd et al., 1995).

For the gradient boosting applied to LFBO and MALIBO, we use the implementation in scikit-learn (Pedregosa et al., 2011) with default setting. The only difference in gradient boosting variants of MALIBO is that we set the initial estimator as the vanilla MALIBO classifer.

D DETAILS OF BENCHMARKS

D.1 HPOBENCH

The hyperparameters for HPOBench and their ranges are demonstrated in Table 1. All hyparameters are discrete and there are in total 66,208 possible combinations. More details can be found in Klein & Hutter (2019).

D.2 ML ALGORITHMS IN HPOBENCH

The hyperparameters for machine learning (ML) algortihms in HPOBench (Eggensperger et al., 2021) and their ranges are summarized in Table 2. More details can be found in Eggensperger et al. (2021).

Table 1: Configuration spaces for HPOBench

Hyperparameter	Range
Initial LR	{ 5×10^{-4}, 1×10^{-3}, 5×10^{-3}, 1×10^{-2}, 5×10^{-2}, 1×10^{-1} }
LR Schedule	{ cosine, fixed }
Batch size	{ 2^3, 2^4, 2^5, 2^6 }
Layer 1 Width	{ 2^4, 2^5, 2^6, 2^7, 2^8, 2^9 }
Activation	{ relu, tanh }
Dropout rate	{ 0.0, 0.3, 0.6 }
Layer 2 Width	{ 2^4, 2^5, 2^6, 2^7, 2^8, 2^9 }
Activation	{ relu, tanh }
Dropout rate	{ 0.0, 0.3, 0.6 }

Table 2: Configuration spaces for ML algorithms in HPOBench

Benchmark	Hyperparameter	type	Log	Range
SVM	C	float	✓	[2^{-10}, 2^{10}]
	gamma	float	✓	[2^{-10}, 2^{10}]
LogReg	alpha	float	✓	[$1e^{-5}$, 1.0]
	eta0	float	✓	[$1e^{-5}$, 1.0]
XGBoost	colsample_bytree	float	✗	[0.1, 1.0]
	eta	float	✓	[2^{-10}, 1.0]
	max_depth	int	✓	[1, 50]
	reg_lambda	float	✓	[2^{-10}, 2^{-10}]
RandomForest	max_depth	int	✓	[1, 50]
	max_features	float	✗	[0.0, 1.0]
	min_samples_leaf	int	✗	[1, 2]
	min_samples_split	int	✓	[2, 128]
MLP	alpha	float	✓	[$1.0e^{-8}$, 1.0]
	batch_size	int	✓	[4, 256]
	depth	int	✗	[1, 3]
	learning_rate_init	float	✓	[$1.0e^{-5}$, 1.0]
	width	int	✓	[16, 1024]

D.3 NASBENCH201

The hyperparameters for NASBench201 and their ranges are summarized in Table 3. All hyparameters are discrete and there are in total 15,625 possible combinations. More details can be found in Dong & Yang (2020).

Table 3: Configuration spaces for NASBench201

Hyperparameter	Range
ARC 0	{ none, skip-connect, conv-1 $\times$ 1, conv-3 $\times$ 3, avg-pool-3 $\times$ 3 }
ARC 1	{ none, skip-connect, conv-1 $\times$ 1, conv-3 $\times$ 3, avg-pool-3 $\times$ 3 }
ARC 2	{ none, skip-connect, conv-1 $\times$ 1, conv-3 $\times$ 3, avg-pool-3 $\times$ 3 }
ARC 3	{ none, skip-connect, conv-1 $\times$ 1, conv-3 $\times$ 3, avg-pool-3 $\times$ 3 }
ARC 4	{ none, skip-connect, conv-1 $\times$ 1, conv-3 $\times$ 3, avg-pool-3 $\times$ 3 }
ARC 5	{ none, skip-connect, conv-1 $\times$ 1, conv-3 $\times$ 3, avg-pool-3 $\times$ 3 }

D.4 THE FORRESTER ENSEMBLE

The original Forrester function (Sobester et al., 2008) is defined following:

$$f(x, a, b, c) = a \cdot (6x - 2)^2 \sin(12x - 4) + b(x - 0.5) - c \qquad x \in [0, 1] \tag{14}$$

The function has one local and one global minimum, and a zero-gradient inflection point in the domain $x \in [0, 1]$. To form the ensemble, we choose the distribution for the parameters as:

$$a \sim \mathcal{U}(0.2, 3) \quad b \sim \mathcal{U}(-5, 15) \quad c \sim \mathcal{U}(-5, 5) \tag{15}$$

Let $\tau = \{a, b, c\}$ and $p(\tau)$ is a three dimensional uniform distribution. The ranges are chosen around the usually used fixed values for the parameters, namely $a = 0.5$, $b = 10$, $c = -5$.

D.5 THE BRANIN ENSEMBLE

The function for the Branin ensemble is the following:

$$f(x, a, b, c) = a(x_2 - bx_1^2 + cx_1 - r) + s(1 - t)\cos(x_1) + s \qquad x_1 \in [-5, 10], x_2 \in [0, 15] \tag{16}$$

The distribution for the parameters are chosen as:

$$\begin{aligned} &a \sim \mathcal{U}(0.5, 1.5) \quad b \sim \mathcal{U}(0.1, 0.15) \quad c \sim \mathcal{U}(1.0, 2.0) \\ &r \sim \mathcal{U}(5.0, 7, 0) \quad s \sim \mathcal{U}(8.0, 12.0) \quad t \sim \mathcal{U}(0.03, 0.05) \end{aligned} \tag{17}$$

Let $\tau = \{a, b, c, r, s, t\}$ and $p(\tau)$ is a six dimensional uniform distribution. The ranges are chosen around the usually used fixed values for the parameters, namely $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = 10$ and $t = 1/(8\pi)$.

D.6 THE HARTMANN3 ENSEMBLE

The function for Hartmann3 (Dixon, 1978) ensemble reads:

$$\begin{aligned} &f(x, \alpha_1, \alpha_2, \alpha_3, \alpha_4) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{i,j}(x_j - P_{i,j})^2\right) \qquad x \in [0, 1] \\ &\boldsymbol{A} = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix} \qquad \boldsymbol{P} = 10^{-4} \cdot \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix} \end{aligned} \tag{18}$$

To form the ensemble, we choose the distribution for the parameters as:

$$\alpha_1 \sim \mathcal{U}(0.0, 2.0) \quad \alpha_2 \sim \mathcal{U}(0.0, 2.0) \quad \alpha_3 \sim \mathcal{U}(2.0, 4.0) \quad \alpha_4 \sim \mathcal{U}(2.0, 4.0) \tag{19}$$

Let $\tau = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ and $p(\tau)$ is a four dimensional uniform distribution.