JUMBO: Scalable Multi-task Bayesian Optimization using Offline Data

Kourosh Hakhamaneshi[1], **Pieter Abbeel** [1], **Vladimir Stojanovic** [1], **Aditya Grover**[2]

[1]UC Berkeley, [2]FAIR

Abstract

The goal of Multi-task Bayesian Optimization (MBO) is to minimize the number of queries required to accurately optimize a target black-box function, given access to offline evaluations of other auxiliary functions. When offline datasets are large, the scalability of prior approaches comes at the expense of expressivity and inference quality. We propose JUMBO, an MBO algorithm that sidesteps these limitations by querying additional data based on a combination of acquisition signals derived from training two Gaussian Processes (GP): a *cold-GP* operating directly in the input domain and a *warm-GP* that operates in the feature space of a deep neural network pretrained using the offline data. Such a decomposition can dynamically control the reliability of information derived from the online and offline data and the use of pretrained neural networks permits scalability to large offline datasets. Theoretically, we derive regret bounds for JUMBO and show that it achieves no-regret under conditions analogous to GP-UCB [29]. Empirically, we demonstrate significant performance improvements over existing approaches on two real-world optimization problems: hyper-parameter optimization and automated circuit design.

Introduction

Many domains in science and engineering involve the optimization of an unknown black-box function. Such functions can be expensive to evaluate, due to costs such as time and money. Bayesian optimization (BO) is a popular framework for such problems as it seeks to minimize the number of function evaluations required for optimizing a target black-box function [25, 8]. In real-world scenarios however, we often have access to offline evaluations of one or more auxiliary black-box functions related to the target function. For example, one might be interested in finding the optimal hyper-parameters of a machine learning model for a given problem and may have access to an offline dataset from previous runs of training the same model on a different dataset for various configurations. Multi-task Bayesian optimization (MBO) is an optimization paradigm that extends BO to exploit such additional sources of information from related black-box functions for efficient optimization [30].

Early works in MBO employ multi-task Gaussian Processes (GP) with inter-task kernels to capture the correlations between the auxiliary and target function [30, 33, 21]. Multi-task GPs however fail to scale to large offline datasets. More recent works have proposed combining neural networks (NN) with probabilistic models to improve scalability. For example, MT-BOHAMIANN [28] uses Bayesian NNs (BNN) [17] as surrogate models for MBO. The performance however, depends on the quality of the inference procedure. In contrast, MT-ABLR [18] uses a deterministic NN followed by a Bayesian Linear Regression (BLR) layer at the output to achieve scalability while permitting exact inference. However, the use of a linear kernel can limit the expressiveness of the posterior.

We propose JUMBO, an MBO algorithm that sidesteps the limitations in expressivity and tractability of prior approaches. In JUMBO, we first train a NN on the auxiliary data to learn a feature space, akin to MT-ABLR but without the BLR restriction on the output layer. Thereafter, we train two GPs simultaneously for online data acquisition via BO: a *warm-GP* on the feature space learned by the NN and a *cold-GP* on the raw input space. The acquisition function in JUMBO combines the individual acquisition functions of both the GPs. It uses the warm-GP to reduce the search space by filtering out poor points. The remaining candidates are scored by the acquisition function for the cold-GP to account for imperfections in learning the feature space of the warm-GP. The use of GPs in the entire framework ensures tractability in posterior inference and updates.

Theoretically, we show that JUMBO is a no-regret algorithm under conditions analogous to those used for analyzing GP-UCB [29]. In practice, we observe significant improvements over the closest baseline on two real-world applications: transferring prior knowledge in hyper-parameter optimization and automated circuit design.

Background

We are interested in maximizing a target black-box function $f : \mathcal{X} \to \mathbb{R}$ defined over a discrete or compact set $\mathcal{X} \subseteq \mathbb{R}^d$. We assume only query access to f. For every query point x, we receive a noisy observation $y = f(x)+\epsilon$. Here, we assume ϵ is standard Gaussian noise, i.e., $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ where σ_n is the noise standard deviation. Our strategy for optimizing f will be to learn a probabilistic model for regressing the inputs x to y using the available data and using that model to guide the acquisition of additional data for updating the

model. In particular, we will be interested in using Gaussian Process regression models within a Bayesian Optimization framework, as described below.

Gaussian Process (GP) Regression

A Gaussian Process (GP) is defined as a set of random variables such that any finite subset of them follows a multivariate normal distribution. A GP can be used to define a prior distribution over the unknown function f, which can be converted to a posterior distribution once we observe additional data. Formally, a GP prior is defined by a mean function $\mu_0 : \mathcal{X} \to \mathbb{R}$ and a valid kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A kernel function κ is valid if it is symmetric and the Gram matrix K is positive semi-definite. Intuitively, the entries of the kernel matrix $K_{i,j} = \kappa(x_i, x_j)$ measure the similarity between any two points x_i and x_j. Given points $X = \{x_1, x_2, \ldots, x_n\}$, the distribution of the function evaluations $\mathbf{f} = [f(x_1), f(x_2), \ldots f(x_n)]$ in a GP prior follows a normal distribution, such that $\mathbf{f}|X \sim \mathcal{N}(\mu_0(X), K(X, X))$ where $\mu_0(X) = [\mu_0(x_1), \mu_0(x_2), \ldots \mu_0(x_n)]$ and $K(X, X)$ is a covariance matrix. For simplicity, we will henceforth assume μ_0 to be a zero mean function.

Given a training dataset $\mathcal{D}$, let $X_\mathcal{D}$ and $\mathbf{y}_\mathcal{D}$ denote the inputs and their noisy observations. Since the observation model is also assumed to be Gaussian, the posterior over f at a test set of points X^* will follow a multivariate normal distribution with the following mean and covariance:

$$\mu(\mathbf{f}^*|\mathcal{D}, X^*) = K(X^*, X_\mathcal{D})^T \tilde{K}_D^{-1} \mathbf{y}_\mathcal{D},$$
$$\Sigma(\mathbf{f}^*|\mathcal{D}, X^*) = K(X^*, X^*) - K(X^*, X_\mathcal{D})^T \tilde{K}_D^{-1} K(X^*, X_\mathcal{D}),$$
$$\text{where } \tilde{K}_D = K(X_\mathcal{D}, X_\mathcal{D}) + \sigma_n^2 I.$$

Due to the inverse operation during posterior computation, standard GPs can be computationally prohibitive for modeling large datasets. We direct the reader to [23] for an overview on GPs.

Bayesian Optimization (BO)

Bayesian Optimization (BO) is a class of sequential algorithms for sample-efficient optimization of expensive black-box functions [8, 25]. A BO algorithm typically runs for a fixed number of rounds. At every round t, the algorithm selects a query point x_t and observes a noisy function value y_t. To select x_t, the algorithm first infers the posterior distribution over functions $p(\mathbf{f}|\{(x_i, y_i)\}_{i=1}^{t-1})$ via a probabilistic model (e.g., Gaussian Processes). Thereafter, x_t is chosen to optimize an uncertainty-aware acquisition function that balances exploration and exploitation. For example, a popular acquisition function is the Upper Confidence Bound (UCB) which prefers points that have high expected value (exploitation) and high uncertainty (exploration). With the new point (x_t, y_t), the posterior distribution can be updated and the whole process is repeated in the next round.

At round t, we define the instantaneous regret as $r_t = f(x^*) - f(x_t)$ where x^* is the global optima and x_t maximizes the acquisition function. Similarly, we can define the cumulative regret at round T as the sum of instantaneous regrets $R_T = \sum_{t=1}^{T} r_t$. A desired property of any BO algorithms is to be *no-regret* where the cumulative regret is sub-linear in T as $T \to \infty$, i.e., $\lim_{T\to\infty} {}^{R_T}/_T = 0$.

Multi-task Bayesian Optimization (MBO)

Our focus setting in this work is a variant of BO, called Multi-Task Bayesian Optimization (MBO). Here, we assume $K > 0$ auxiliary real-valued black-box functions $\{f_1, \ldots, f_K\}$, each having the same domain $\mathcal{X}$ as the target function f [30, 28]. For each function f_k, we have an offline dataset $\mathcal{D}^{(k)}$ consisting of pairs of input points x and the corresponding function evaluations $f_k(x)$. If these auxiliary functions are related to the target function, then we can transfer knowledge from the offline data $\mathcal{D}^{\text{aux}} = \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(K)}$ to improve the sample-efficiency for optimizing f. In certain applications, we might also have access to offline data from f itself. However, in practice, f is typically expensive to query and its offline dataset $\mathcal{D}^f$ will be very small.

We discuss some prominent works in MBO that are most closely related to our proposed approach below. See Section 12 for further discussion about other relevant work.

Multi-task BO [30] is an early approach that employs a custom kernel within a multi-task GP [33] to model the relationship between the auxiliary and target functions. Similar to standard GPs, multi-task GPs fail to scale for large offline datasets.

On the other hand, parametric models such as neural networks (NN), can effectively scale to larger datasets but do not defacto quantify uncertainty. Hybrid methods such as **DNGO** [27] achieve scalability for (single task) BO through the use of a feed forward deep NN followed by Bayesian Linear Regression (BLR) [2]. The NN is trained on the existing data via a simple regression loss (e.g, mean squared error). Once trained, the NN parameters are frozen and the output layer is replaced by BLR for the BO routine. For BLR, the computational complexity of posterior updates scales linearly with the size of the dataset. This step can be understood as applying a GP to the output features of the NN with a linear kernel (i.e. $\kappa(x_i, x_j) = h_\phi(x_i)^T h_\phi(x_j)$ where h is the NN function with parameters ϕ). For BLR, the computational complexity of posterior inference is linear w.r.t. the number of data points and thus DNGO can scale to large offline datasets.

MT-ABLR [18] extends DNGO to multi task settings by training a single NN to learn a shared representation $h_\phi(x)$ followed by task-specific BLR layers (i.e. predicting $f_1(x), ..., f_K(x)$, and $f(x)$ based on inputs). The learning objective corresponds to the maximization of sum of the marginal log-likelihoods for each task: $\sum_{t=1}^{K+1} p(\mathbf{y}_t|w_t, h_\phi(X_t), \sigma_t)$. The main task is included in the last index, w_t is the Bayesian Linear layer weights for task t with prior $p(w_t) = \mathcal{N}(0, \sigma_{w_t}^2 I)$, σ_t and σ_{w_t} are the hyper-prior parameters, and $(X_t, \mathbf{y}_t)$ is the observed data from task t. Learning $h_\phi(x)$ by directly maximizing the marginal likelihood improves the performance of DNGO while maintaining the computational scalability of its posterior inference in case of large offline data. However, both DNGO and ABLR have implicit assumptions on the existence of a feature space under

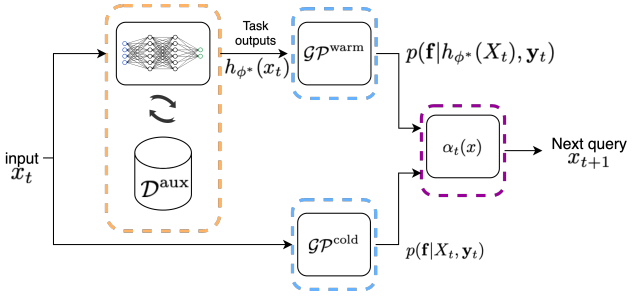


Figure 1: JUMBO. During the pretraining phase, we learn a NN mapping h_{ϕ^*} (orange) for the warm-GP. The next query based on α_t (purple) will be the point that has a high score based on the acquisition function of both warm and cold GP (blue).

which the target function can be expressed as a linear combination. This can be a restrictive assumption and furthermore, there is no guarantee that given finite data such feature space can be learned.

MT-BOHAMIANN [28] addresses the limited expressivity of prior approaches by employing Bayesian NNs to specify the posterior over $\mathbf{f}$ and feed the NN with input x and additional learned task-specific embeddings $\psi(t)$ for task t. While allowing for a principled treatment of uncertainties, fully Bayesian NNs are computationally expensive to train and their performance depends on the approximation quality of stochastic gradient HMC methods used for posterior inference.

Scalable MBO via JUMBO

In the previous section, we observed that prior MBO algorithms make trade-offs in either posterior expressivity or inference quality in order to scale to large offline datasets. Our goal is to show that these trade-offs can be significantly mitigated and consequently, the design of our proposed MBO framework, which we refer to as JUMBO, will be guided by the following desiderata: (1) Scalability to large offline datasets (e.g., via NNs) (2) Exact and computationally tractable posterior updates (e.g., via GPs) (3) Flexible and expressive posteriors (e.g., via non-linear kernels).

Regression Model

The regression model in JUMBO is composed of two GPs: a *warm-GP* and a *cold-GP* denoted by $\mathcal{GP}^{\text{warm}}(0, \kappa^w)$ and $\mathcal{GP}^{\text{cold}}(0, \kappa^c)$, respectively. As shown in Figure 1, both GPs are trained to model the target function f but operate in different input spaces, as we describe next.

$\mathcal{GP}^{\text{warm}}$ (with hyperparameters θ_w) operates on a feature representation of the input space $h_\phi(x)$ derived from the offline dataset $\mathcal{D}^{\text{aux}}$. To learn this feature space, we train a multi-headed feed-forward NN to minimize the mean squared error for each auxiliary task, akin to DNGO [27]. Thereafter, in contrast to both DNGO and ABLR, we do not train separate output BLR modules. Rather, we will directly train $\mathcal{GP}^{\text{warm}}$ on the output of the NN using the data acquired from the target function f. Note that for training $\mathcal{GP}^{\text{warm}}$, we can use any non-linear kernel, which results in an expressive posterior that allows for exact and tractable inference using closed-form expressions.

Additionally, we can encounter scenarios where some of the auxiliary functions are insufficient in reducing the uncertainty in inferring the target function. In such scenarios, relying solely on $\mathcal{GP}^{\text{warm}}$ can significantly hurt performance. Therefore, we additionally initialize $\mathcal{GP}^{\text{cold}}$ (with hyperparameters θ_c) directly on the input space $\mathcal{X}$.

If we also have access to offline data from f (i.e. $\mathcal{D}^f$), the hyperparameters of the warm and cold GPs can also be pre-trained jointly along with the neural network parameters. The overall pre-training objective is then given by:

$$\mathcal{L}(\phi, \theta_w, \theta_c) = \mathcal{L}^{\text{MSE}}(\phi|\mathcal{D}^{\text{aux}}) + \mathcal{L}^{\mathcal{GP}}(\theta_w|\mathcal{D}^f) + \mathcal{L}^{\mathcal{GP}}(\theta_c|\mathcal{D}^f) \quad (1)$$

where $\mathcal{L}^{\mathcal{GP}}(\cdot|\mathcal{D}^f)$ denotes the negative marginal log-likelihood for the corresponding GP on $\mathcal{D}^f$.

Acquisition Procedure

Post the offline pre-training of the JUMBO's regression model, we can use it for online data acquisition in a standard BO loop. The key design choice here is the acquisition function, which we describe next. At round t, let $\alpha_t^{\text{warm}}(x)$ and $\alpha_t^{\text{cold}}(x)$ be the single task acquisition function (e.g. UCB) of the warm and cold GPs, after observing $t-1$ data points, respectively.

Our guiding intuition for the acquisition function in JUMBO is that we are most interested in querying points which are scored highly by both acquisition functions. Ideally, we want to first sort points based on α^{warm} scores and then from the top choices select the ones with highest α^{cold} score. To realize this acquisition function on a continuous input domain, we define it as a convex combination of the individual acquisition functions by employing a *dynamic* interpolation coefficient $\lambda_t(x) \in [0, 1]$. Formally,

$$\alpha_t(x) = \lambda_t(x)\alpha_t^{\text{cold}}(x) + (1 - \lambda_t(x))\alpha_t^{\text{warm}}(x). \quad (2)$$

In Eq. 2, By choosing $\lambda_t(x)$ to be close to 1 for points with $\alpha_t^{\text{warm}}(x) \approx \max_x \alpha_t^{\text{warm}}(x)$, we can ensure to acquire points that have high acquisition scores as per both $\alpha_t^{\text{cold}}(x)$ and $\alpha_t^{\text{warm}}(x)$. Next, we will discuss some theoretical results that shed more light on the design of $\lambda_t(x)$.

Theoretical Analysis

Here, we will formally derive the regret bound for JUMBO and provide insights on the conditions under which JUMBO outperforms GP-UCB [29]. For this analysis, we will use Upper Confidence Bound (UCB) as our acquisition function for the warm and cold GPs. To do so, we utilize the notion of Maximum Information Gain (MIG).

Definition 1 (Maximum Information Gain [29]). *Let $f \sim \mathcal{GP}(0, \kappa)$, $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Consider any $\mathcal{X} \subset \mathbb{R}^d$ and let $\tilde{\mathcal{X}} = \{x_1, ..., x_n\} \subset \mathcal{X}$ be a finite subset. Let $\mathbf{y}_{\tilde{\mathcal{X}}} \in \mathbb{R}^n$ be n noisy observations such that $(\mathbf{y}_{\tilde{\mathcal{X}}})_i = (\mathbf{f}_{\tilde{\mathcal{X}}})_i + \epsilon_i$, $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$. Let I denote the Shannon mutual information.*

The MIG $\Psi_n(\mathcal{X})$ of set $\mathcal{X}$ after n evaluations is the maximum mutual information between the function values and observations among all choices of n points in $\mathcal{X}$. Formally,

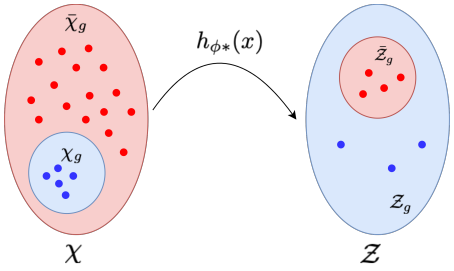


Figure 2: The effect of the pre-trained NN $h_{\phi^*}(x)$ on $\mathcal{X}$. In the desirable case, $\bar{\mathcal{X}}_g$ gets significantly compressed to $\bar{\mathcal{Z}}_g$.

$$\Psi_n(\mathcal{X}) = \max_{\tilde{\mathcal{X}} \subset \mathcal{X}, |\tilde{\mathcal{X}}|=n} I(\mathbf{y}_{\tilde{\mathcal{X}}}; \mathbf{f}_{\tilde{\mathcal{X}}})$$

This quantity depends on kernel parameters and the set $\mathcal{X}$, and also serves as an important tool for characterizing the difficulty of a GP-bandit. For a given kernel, it can be shown that $\Psi_n(\mathcal{X}) \propto \Pi(\mathcal{X})$ where $\Pi(\mathcal{X}) = |\mathcal{X}|$ for discrete and $\text{Vol}(\mathcal{X})$ for the continuous case [29]. For example for *Radial Basis* kernel $\Psi_n([0,1]^d) \in O(\log(n)^{d+1})$. For brevity, we focus on settings where $\mathcal{X}$ is discrete. Further results and analysis for the continuous case are deferred to Appendix **??**.

For GP-UCB [29], it has been shown that for any $\delta \in (0,1)$, if $f \sim \mathcal{GP}(0, \kappa)$ (i.e., the GP assigns non-zero probability to the target function f), then the cumulative regret R_T after T rounds will be bounded with probability at least $1-\delta$:

$$Pr\{R_T \leq \sqrt{CT\beta_T \Psi_T(\mathcal{X})}, \forall T \geq 1\} \geq 1-\delta \qquad (3)$$

with $C = \frac{8}{\log(1+\sigma_n^{-2})}$ and $\beta_T = 2\log\left(\frac{|\mathcal{X}|\pi^2 T^2}{6\delta}\right)$.

Recall that $h_\phi : \mathcal{X} \to \mathcal{Z}$ is a mapping from input space $\mathcal{X}$ to the feature space $\mathcal{Z}$. We will further make the following modeling assumptions to ensure that the target black-box function f is a sample from both the cold and warm GPs.

Assumption 1. $f \sim \mathcal{GP}^{cold}(0, \kappa^c)$.

Assumption 2. *Let ϕ^* denote the NN parameters obtained via pretraining (Eq. 1).Then, there exists a function $g \sim \mathcal{GP}^{warm}(0, \kappa^w)$ such that $f = g \circ h_{\phi^*}$.*

Theorem 1. *Let $\mathcal{X}_g \subset \mathcal{X}$ and $\bar{\mathcal{X}}_g = \mathcal{X} \setminus \mathcal{X}_g$ be some arbitrary partitioning of the input domain $\mathcal{X}$. Define the interpolation coefficient as an indicator $\lambda_t(x) = \mathbb{1}(x \in \mathcal{X}_g)$. Then under Assumptions 1 and 2, JUMBO is no-regret.*

Specifically, let s be the number of rounds such that the JUMBO queries for points $x_t \in \bar{\mathcal{X}}_g$. Then, for any $\delta \in (0,1)$, running JUMBO for T iterations results in a sequence of candidates $(x_t)_{t=1}^{t=T}$ for which the following holds with probability at least $1-\delta$:

$$R_T < \sqrt{CT\beta_T\{\Psi_{T-s}(\mathcal{X}_g) + \Psi_s(\bar{\mathcal{Z}}_g)\}}, \forall T \geq 1 \qquad (4)$$

where $C = \frac{8}{\log(1+\sigma_n^{-2})}$, $\beta_t = 2\log\left(\frac{|\mathcal{X}|\pi^2 t^2}{3\delta}\right)$, and $\bar{\mathcal{Z}}_g = \{h_{\phi^*}(x) | x \in \bar{\mathcal{X}}_g\}$ is the set of output features for $\bar{\mathcal{X}}_g$.

Based on the regret bound in Eq. 4, we can conclude that if the partitioning $\mathcal{X}_g$ is chosen such that $\Pi(\bar{\mathcal{Z}}_g) \ll \Pi(\bar{\mathcal{X}}_g)$ *and* $\Pi(\mathcal{X}_g) \ll \Pi(\mathcal{X})$, then JUMBO has a tighter bound than GP-UCB. The first condition implies that the second term in Eq. 4 is negligible and intuitively means that $\mathcal{GP}^{\text{warm}}$ will only need a few samples to infer the posterior of f defined on $\bar{\mathcal{X}}_g$, making BO more sample efficient. The second condition implies that the $\Psi_{T-s}(\mathcal{X}_g) \ll \Psi_T(\mathcal{X})$ which in turn makes the regret bound of JUMBO tighter than GP-UCB. Note that $\mathcal{X}_g$ cannot be made arbitrarily small, since $\bar{\mathcal{X}}_g$ (and therefore $\bar{\mathcal{Z}}_g$) will get larger which conflicts with the first condition.

Figure 2 provides an illustrative example. If the learned feature space $h_{\phi^*}(x)$ compresses set $\bar{\mathcal{X}}_g$ to a smaller set $\bar{\mathcal{Z}}_g$, then $\mathcal{GP}^{\text{warm}}$ can infer the posterior of $g(h_{\phi^*}(x))$ with only a few samples in $\bar{\mathcal{X}}_g$ (because MIG is lower). Such $h_{\phi^*}(x)$ will likely emerge when tasks share high-level features with one another. In the appendix, we have included an empirical analysis to show that $\mathcal{GP}^{\text{warm}}$is indeed operating on a compressed space $\mathcal{Z}$. Consequently, if $\mathcal{X}_g$ is reflective of promising regions consisting of near-optimal points i.e. $\mathcal{X}_g = \{x \in \mathcal{X} \mid f(x^*) - f(x) \leq l_f\}$ for some $l_f > 0$, BO will be able to quickly discard points from subset $\bar{\mathcal{X}}_g$ and acquire most of its points from $\mathcal{X}_g$.

Choice of interpolation coefficient $\lambda_t(x)$

The above discussion suggests that the partitioning $\mathcal{X}_g$ should ideally consist of near-optimal points. In practice, we do not know f and hence, we rely on our surrogate model to define $\mathcal{X}_g^{(t)} = \{x \in \mathcal{X} \mid \alpha_t^{\text{warm}*} - \alpha_t^{\text{warm}}(x) \leq l_\alpha\}$. Here, $\alpha_t^{\text{warm}*}$ is the optimal value of $\alpha_t^{\text{warm}}(x)$ and the acquisition threshold $l_\alpha > 0$ is a hyper-parameter used for defining near-optimal points w.r.t. $\alpha_t^{\text{warm}}(x)$. At one extreme, $l_\alpha \to \infty$ corresponds to the case where $\alpha_t(x) = \alpha_t^{\text{cold}}(x)$ (i.e. the GP-UCB routine) and the other extreme $l_\alpha \to 0$ corresponds to case with $\alpha_t(x) = \alpha_t^{\text{warm}}(x)$.

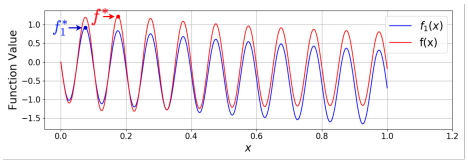
Figure 3 illustrates a synthetic 1D example on how JUMBO obtains the next query point. Figure 3a shows the main objective $f(x)$ (red) and the auxiliary task $f_1(x)$ (blue). They share a periodic structure but have different optimums. Figure 3b shows the correlation between the two.

Applying GP-UCB [29] will require a considerable amount of samples to learn the periodic structure and the optimal solution. However in JUMBO, as shown in Figure 3c, the warm-GP, trained on $(h_{\phi^*}(x), y)$ samples, can learn the periodic structure using only 6 samples, while the posterior of the cold-GP has not yet learned this structure.

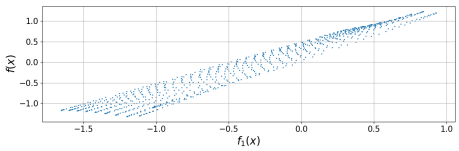
It can also be noted from Figure 3c that JUMBO's acquisition function is $\alpha_t^{\text{cold}}(x)$ when the value of $\alpha_t^{\text{warm}}(x)$ is close to $\alpha_t^{\text{warm}*}$. Therefore, the next query point (marked with a star) has a high score based on both acquisition functions. We summarize JUMBO in Algorithm 1.

Experiments

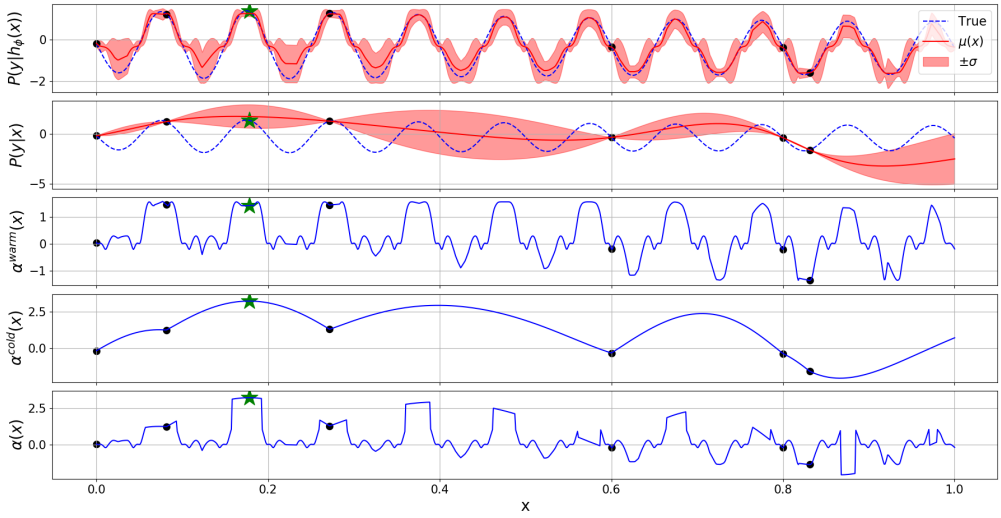
We are interested in investigating the following questions: (1) How does JUMBO perform on benchmark real-world black-box optimization problems relative to baselines? (2) How does the choice of threshold l_α impact the performance of JUMBO? (3) Is it necessary to have a non-linear mapping



(a) Target (red) and auxiliary (blue) task



(b) Correlation between target and auxiliary tasks



(c) Posterior of $\mathcal{GP}^{\text{warm}}$, $\mathcal{GP}^{\text{cold}}$, their UCB and JUMBO's acquisition function

Figure 3: Dynamics of JUMBO after observing 6 data points (a) The two functions have different optimums (b) The tasks are related (c) Iteration 4 of the BO with our proposed model, from top to bottom: (1) GP modeling input to objective using $(x, h_{\phi^*}(x), y)$ samples (2) GP modeling input to objective using (x, y) samples (3) UCB acquisition function for $\mathcal{GP}^{\text{warm}}$(4) UCB acquisition function for $\mathcal{GP}^{\text{cold}}$(5) JUMBO's acquisition function that compromises between the optimum of the two.

on the features learned from the offline dataset or a BLR layer is sufficient?

Our codebase is based on BoTorch [1] and is provided in the Supplementary Materials with additional details in Appendix **??**.

Application: Hyperparameter optimization

Datasets. We consider the task of optimizing hyperparameters for fully-connected NN architectures on 4 regression benchmarks from HPOBench [15]: *Protein Structure* [22], *Parkinsons Telemonitoring* [31], *Naval Propulsion* [4], and *Slice Localization* [10]. HPOBench provides a look-up-table-based API for querying the validation error of all possible hyper-parameter configurations for a given regression task. These configurations are specified via 9 hyperparameters, that include continuous, categorical, and integer valued variables.

The objective we wish to minimize is the validation error of a regression task after 100 epochs of training. For this purpose, we consider an offline dataset that consists of validation errors for some randomly chosen configurations after 3 epochs on a given dataset. The target task is to optimize this error after 100 epochs. In [15], the authors show that this problem is non-trivial as there is small correlation between epochs 3 and 100 for top-1% configurations across all datasets of interest.

Evaluation protocol. We validate the performance of JUMBO against the following baselines with a UCB acquisition function [29]:

- **GP-UCB** [29] (i.e. cold-GP only) trains a GP from scratch disregarding $\mathcal{D}^{\text{aux}}$ completely. Equivalently, it can be interpreted as JUMBO with $\lambda_t(x) = 1 \quad \forall x, t \geq 1$ in Eq. 2 and $\alpha(x) = \alpha^{\text{UCB}}(x)$.
- **MT-BOHAMIANN** [28] trains a BNN on all tasks jointly via SGHMC (Section).
- **MT-ABLR** [18] trains a shared NN followed by task-specific BLR layers (Section).
- **GCP** [24] uses Gaussian Copula Processes to jointly model the offline and online data.
- **MF-GP-UCB** [13] extends the GP-UCB baseline to a multi-fidelity setting where the source task can be interpreted as a low-fidelity proxy for the target task.
- **Offline DKL** (i.e. warm-GP only) is our proposed extension to Deep Kernel Learning, where we train a single GP online in the latent space of a NN pretrained on $\mathcal{D}^{\text{aux}}$ (See Section 12 for details). Equivalently, it can be interpreted as JUMBO with $\lambda_t(x) = 0$ in Eq. 2.

Results. We run JUMBO (with $l_\alpha = 0.1$) on all baselines for 50 rounds and 5 random seeds each and measure the simple regret per iteration. The regret curves are shown in Figure 4. We find that JUMBO achieves lower regret than the previous state-of-the-art algorithms for MBO in almost all cases. We believe the slightly worse performance on the slice dataset relative to other baselines is due to the extremely low top-1%

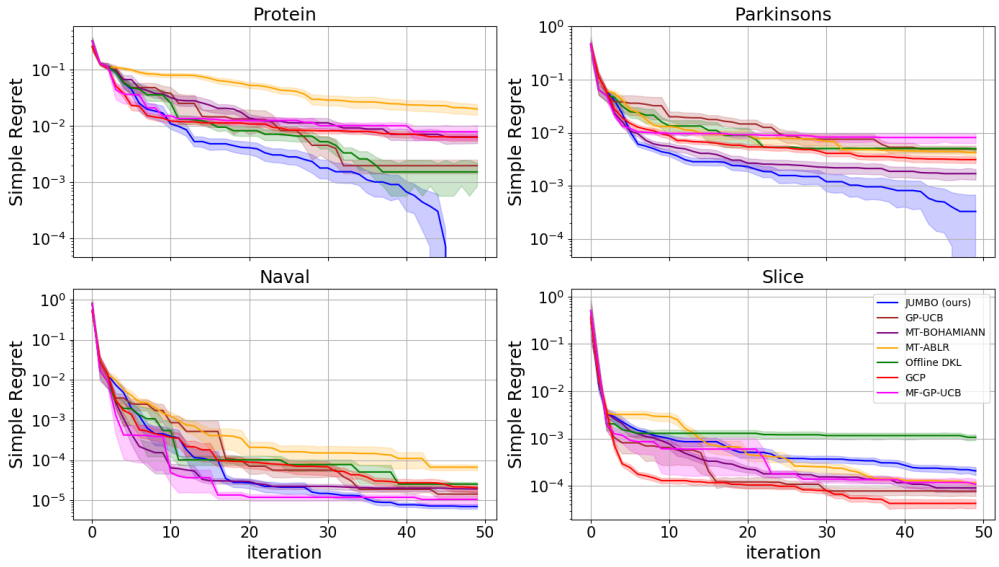


Figure 4: The regret of MBO algorithms on Protein, Parkinsons, Naval, and Slice datasets. Standard errors are measured across 20 random seeds.

Algorithm 1: JUMBO

Input: Offline auxiliary dataset $\mathcal{D}^{\text{aux}}$, Offline target dataset $\mathcal{D}_0^f$ (optional; default: empty set), Threshold l_α

Output: Sequence of solution candidates $\{x_t\}_{t=1}^T$ maximizing target function f

Initialize NN $h_\phi(x)$, $\mathcal{GP}^{\text{cold}}$, $\mathcal{GP}^{\text{warm}}$.
Pretrain NN params jointly with $\mathcal{GP}^{\text{cold}}$ and $\mathcal{GP}^{\text{warm}}$ hyper-params using $\mathcal{D}^{\text{aux}}$ and $\mathcal{D}_0^f$ as per Eq. 1.;
Initialize $\mathcal{D}_0^{\text{cold}} = \{\}$, $\mathcal{D}_0^{\text{warm}} = \{\}$.
for *round* $t = 1$ **to** T **do**
  Set $\alpha_t^{\text{warm}*} = \text{argmax}_{x\in\chi}\, \alpha_t^{\text{warm}}(x)$.
  Set $\lambda_t(x) = \mathbb{1}(\alpha_t^{\text{warm}*} - \alpha_t^{\text{warm}}(x) \leq l_\alpha)$.
  Set $\alpha_t(x) = \lambda_t(x)\alpha_t^{\text{cold}}(x) + (1 - \lambda_t(x))\alpha_t^{\text{warm}}(x)$
  Pick $x_t = \text{argmax}_{x\in\chi}\, \alpha_t(x)$.
  Obtain noisy observation y_t for x_t.
  Update $\mathcal{D}_t^{\text{cold}} \leftarrow \mathcal{D}_{t-1}^{\text{cold}} \cup \{(x_t, y_t)\}$ and $\mathcal{GP}^{\text{cold}}$.
  Update $\mathcal{D}_t^{\text{warm}} \leftarrow \mathcal{D}_{t-1}^{\text{warm}} \cup \{(h_{\phi^*}(x_i), y_i)\}$ and $\mathcal{GP}^{\text{warm}}$.
end

correlation between epoch 3 and epoch 100 on this dataset as compared to others (See Figure 10 in [15]), which could result in a suboptimal search space partitioning obtained via the warm-GP. For all other datasets, we find JUMBO to be the best performing method. Notably, on the Protein dataset, JUMBO is always able to find the global optimum, unlike the other approaches.

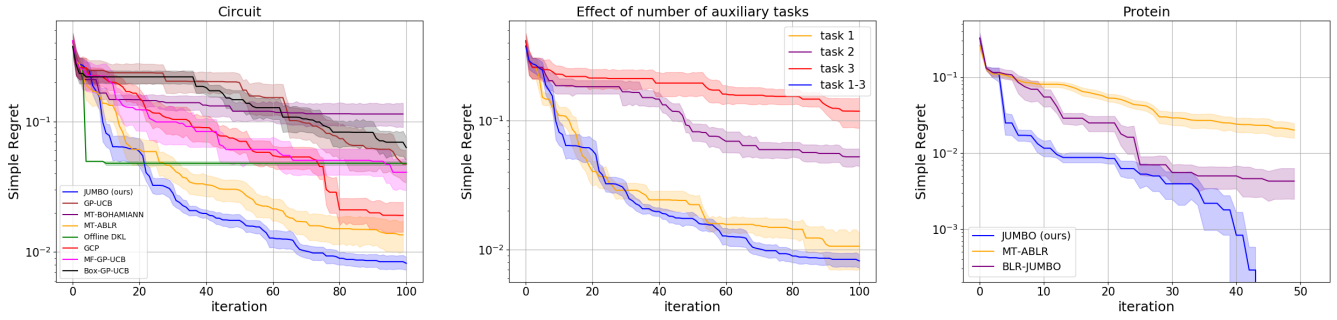
Application: Automated Circuit Design

Next, we consider a real-world use case in optimizing circuit design configurations for a suitable performance metric, e.g., power, noise, etc. In practice, designers are interested in performing *layout* simulations for measuring the performance metric on any design configuration. These simulations are however expensive to run; designers instead often turn to *schematic* simulations which return inexpensive proxy metrics correlated with the target metric.

In this problem, the circuit configurations are represented by an 8 dimensional vector, with elements taking continuous values between 0 and 1. The offline dataset consists of 1000 pairs of circuit configurations and 3 auxiliary signals including a scalar goodness score based on the schematic simulations. We consider the same baselines as before. We also consider BOX-GP-UCB [19] which confines the search space to a hyper-cube over the promising region based on all auxiliary tasks in the offline data. Unlike the considered HPO problems, the offline circuit dataset contains data from more than just one auxiliary task, allowing us to consider BOX-GP-UCB as a viable baseline. MF-GP-UCB was ran with the schematic score as the lower fidelity approximation of the target function. We ran each algorithm with $l_\alpha = 0.1$ for 100 iterations and measured simple regret against iteration. As reflected in the regret curves in Figure 5a, JUMBO outperforms other algorithms.

Ablations

Effect of auxiliary tasks. It is important to analyze how learning on other tasks affects the performance. To this end, we considered the circuit design problem with 1 and 3 auxiliary offline tasks. In Figure 5b, task 1 (yellow) is the most correlated and task 3 (red) is the least correlated task with the objective function. The regret curves suggest that the per-



(a) (b) (c)

Figure 5: (a) Circuit Design results (b) JUMBO with 3 aux. tasks is better than JUMBO with each individual aux. tasks. (c) The Non-linear mapping is a crucial piece of JUMBO's algorithm.

formance would be poor if the correlation between tasks is low. Moreover, the features pre-trained on the combination of all three tasks provide more information to the warm-GP than those pre-trained only on one of the tasks.

BLR with JUMBO's acquisition function. A key difference between JUMBO and ABLR [18] is replacing the BLR layer with a GP. To show the merits of having a GP, we ran an experiment on Protein dataset and replaced the GP with a BLR in JUMBO's procedure. Figure 5c shows that JUMBO with $\mathcal{GP}^{\text{warm}}$ significantly outperforms JUMBO with a BLR layer.

Related Work

Transfer Learning in Bayesian Optimization: Utilizing prior information for applying transfer learning to improve Bayesian optimization has been explored in several prior papers. Early work of [30] focuses on the design of multi-task kernels for modeling task correlations [21]. These models tend to suffer from lack of scalability; [35, 6] show that this challenge can be partially mitigated by training an ensemble of task-specific GPs that scale linearly with the number of tasks but still suffer from cubic complexity in the number of observations for each task. To address scalability and robust treatment of uncertainty, several prior works have been suggested [24, 28, 18]. [24] employs a Gaussian Copula to learn a joint prior on hyper-parameters based on offline tasks, and then utilizes a GP on the online task for adapt to the target function. [28] uses a BNN as surrogates for MBO; however, since training BNNs is computationally intensive [18] proposes to use a deterministic NN followed by a BLR layer at the output to achieve scalability.

Some other prior work exploit certain assumptions between the source and target data. For example [26, 9] assume an ordering of the tasks and use this information to train GPs to model residuals between the target and auxiliary tasks. [7, 34] assume existence of a similarity measure between prior and target data which may not be easy to define for problems other than hyper-parameter optimization. A simpler idea is to use prior data to confine the search space to promising regions [19]. However, this highly relies on whether the confined region includes the optimal solution to the target task. Another line of work studies utilizing prior optimization runs to meta-learn acquisition functions [32]. This idea can be utilized in addition to our method and is not a competing direction.

Multi-fidelity Black-box Optimization (MFBO): In multi-fidelity scenarios we can query for noisy approximations to the target function relatively cheaply. For example, in hyperparameter optimization, we can query for cheap proxies to the performance of a configuration on a smaller subset of the training data [20], early stopping [16], or by predicting learning curves [5, 14]. We direct the reader to Section 1.4 in [12] for a comprehensive survey on MBFO. Such methods, similar to MF-GP-UCB [13] (section 12), are typically constrained to scenarios where such low fidelities are explicitly available and make strong continuity assumptions between the low fidelities and the target function.

Deep Kernel Learning (DKL): Commonly used GP kernels (e.g. RBF, Matern) can only capture simple correlations between points a priori. DKL [11, 3] addresses this issue by learning a latent representation via NN that can be fed to a standard kernel at the output. [27] employs linear kernels at the output of a pre-trained NN while [11] extends it to use non-linear kernels. The *warm-GP* in JUMBO can be understood as a DKL surrogate model trained using offline data from auxiliary tasks.

Conclusion

We proposed JUMBO, a no-regret algorithm that employs a careful hybrid of neural networks and Gaussian Processes and a novel acquisition procedure for scalable and sample-efficient Multi-task Bayesian Optimization. We derived JUMBO's theoretical regret bound and empirically showed it outperforms other competing approaches on set of real-world optimization problems.

References

[1] Balandat, M.; Karrer, B.; Jiang, D.; Daulton, S.; Letham, B.; Wilson, A. G.; and Bakshy, E. 2020. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, 33.

[2] Bishop, C. M. 2006. *Pattern recognition and machine learning*. springer.

[3] Calandra, R.; Peters, J.; Rasmussen, C. E.; and Deisenroth, M. P. 2016. Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 3338–3345. IEEE.

[4] Coraddu, A.; Oneto, L.; Ghio, A.; Savio, S.; Anguita, D.; and Figari, M. 2016. Machine learning approaches for improving condition-based maintenance of naval propulsion plants. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 230(1): 136–153.

[5] Domhan, T.; Springenberg, J. T.; and Hutter, F. 2015. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-fourth international joint conference on artificial intelligence*.

[6] Feurer, M.; Letham, B.; and Bakshy, E. 2018. Scalable meta-learning for Bayesian optimization. *arXiv preprint arXiv:1802.02219*.

[7] Feurer, M.; Springenberg, J.; and Hutter, F. 2015. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

[8] Frazier, P. I. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

[9] Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J.; and Sculley, D. 2017. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 1487–1495.

[10] Graf, F.; Kriegel, H.-P.; Schubert, M.; Pölsterl, S.; and Cavallaro, A. 2011. 2d image registration in ct images using radial image descriptors. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 607–614. Springer.

[11] Huang, W.; Zhao, D.; Sun, F.; Liu, H.; and Chang, E. 2015. Scalable gaussian process regression using deep neural networks. In *Twenty-fourth international joint conference on artificial intelligence*. Citeseer.

[12] Hutter, F.; Kotthoff, L.; and Vanschoren, J. 2019. *Automated machine learning: methods, systems, challenges*. Springer Nature.

[13] Kandasamy, K.; Dasarathy, G.; Oliva, J.; Schneider, J.; and Poczos, B. 2019. Multi-fidelity gaussian process bandit optimisation. *Journal of Artificial Intelligence Research*, 66: 151–196.

[14] Klein, A.; Falkner, S.; Bartels, S.; Hennig, P.; and Hutter, F. 2017. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*, 528–536. PMLR.

[15] Klein, A.; and Hutter, F. 2019. Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv preprint arXiv:1905.04970*.

[16] Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; and Talwalkar, A. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1): 6765–6816.

[17] Neal, R. M. 2012. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

[18] Perrone, V.; Jenatton, R.; Seeger, M.; and Archambeau, C. 2018. Scalable hyperparameter transfer learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6846–6856.

[19] Perrone, V.; Shen, H.; Seeger, M.; Archambeau, C.; and Jenatton, R. 2019. Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. *arXiv preprint arXiv:1909.12552*.

[20] Petrak, J. 2000. Fast subsampling performance estimates for classification algorithm selection. In *Proceedings of the ECML-00 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, 3–14.

[21] Poloczek, M.; Wang, J.; and Frazier, P. I. 2016. Warm starting Bayesian optimization. In *2016 Winter Simulation Conference (WSC)*, 770–781. IEEE.

[22] Rana, P. 2013. Physicochemical properties of protein tertiary structure data set. *UCI Machine Learning Repository*.

[23] Rasmussen, C. E. 2003. Gaussian processes in machine learning. In *Summer school on machine learning*, 63–71. Springer.

[24] Salinas, D.; Shen, H.; and Perrone, V. 2020. A quantile-based approach for hyperparameter transfer learning. In *International Conference on Machine Learning*, 8438–8448. PMLR.

[25] Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; and De Freitas, N. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175.

[26] Shilton, A.; Gupta, S.; Rana, S.; and Venkatesh, S. 2017. Regret bounds for transfer learning in Bayesian optimisation. In *Artificial Intelligence and Statistics*, 307–315. PMLR.

[27] Snoek, J.; Rippel, O.; Swersky, K.; Kiros, R.; Satish, N.; Sundaram, N.; Patwary, M.; Prabhat, M.; and Adams, R. 2015. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, 2171–2180. PMLR.

[28] Springenberg, J. T.; Klein, A.; Falkner, S.; and Hutter, F. 2016. Bayesian optimization with robust Bayesian neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 4141–4149.

[29] Srinivas, N.; Krause, A.; Kakade, S.; and Seeger, M. 2010. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on Machine Learning*. Omnipress.

[30] Swersky, K.; Snoek, J.; and Adams, R. P. 2013. Multi-Task Bayesian Optimization. *Advances in Neural Information Processing Systems*, 26: 2004–2012.

[31] Tsanas, A.; Little, M. A.; McSharry, P. E.; and Ramig, L. O. 2010. Enhanced classical dysphonia measures and sparse regression for telemonitoring of Parkinson's disease progression. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 594–597. IEEE.

[32] Volpp, M.; Fröhlich, L. P.; Fischer, K.; Doerr, A.; Falkner, S.; Hutter, F.; and Daniel, C. 2019. Meta-learning acquisition functions for transfer learning in bayesian optimization. *arXiv preprint arXiv:1904.02642*.

[33] Williams, C.; Bonilla, E. V.; and Chai, K. M. 2007. Multi-task Gaussian process prediction. *Advances in neural information processing systems*, 153–160.

[34] Wistuba, M.; Schilling, N.; and Schmidt-Thieme, L. 2015. Learning hyperparameter optimization initializations. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, 1–10. IEEE.

[35] Wistuba, M.; Schilling, N.; and Schmidt-Thieme, L. 2018. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1): 43–78.