Pre-training helps Bayesian optimization too

Zi Wang   WANGZI@GOOGLE.COM

George E. Dahl   GDAHL@GOOGLE.COM

Kevin Swersky   KSWERSKY@GOOGLE.COM

Chansoo Lee   CHANSOO@GOOGLE.COM

Zelda Mariet   ZMARIET@GOOGLE.COM

Zachary Nado   ZNADO@GOOGLE.COM

Justin Gilmer   GILMER@GOOGLE.COM

Jasper Snoek   JSNOEK@GOOGLE.COM

Zoubin Ghahramani   ZOUBIN@GOOGLE.COM

Abstract

Bayesian optimization (BO) has become a popular strategy for global optimization of many expensive real-world functions. Contrary to a common belief that BO is suited to optimizing black-box functions, it actually requires domain knowledge on characteristics of those functions to deploy BO successfully. Such domain knowledge often manifests in Gaussian process priors that specify initial beliefs on functions. However, even with expert knowledge, it is not an easy task to select a prior. This is especially true for hyperparameter tuning problems on complex machine learning models, where landscapes of tuning objectives are often difficult to comprehend. We seek an alternative practice for setting these functional priors. In particular, we consider the scenario where we have data from similar functions that allow us to pre-train a tighter distribution a priori. To verify our approach in realistic model training setups, we collected a large multi-task hyperparameter tuning dataset by training tens of thousands of configurations of near-state-of-the-art models on popular image and text datasets, as well as a protein sequence dataset. Our results show that on average, our method is able to locate good hyperparameters at least 3 times more efficiently than the best competing methods.

1. Introduction

Bayesian optimization (BO) has been successfully applied in numerous real-world global optimization problems, ranging broadly from hyperparameter tuning (Snoek et al., 2012; Kotthoff et al., 2019) to chemical synthesis (Shields et al., 2021; Griffiths and Hernández-Lobato, 2020), drug discovery (Pyzer-Knapp, 2018), aerospace engineering (Lam et al., 2018), robotics (Drieß et al., 2017; Wang et al., 2017a) and the list goes on. These successful BO applications benefit from expert knowledge on characteristics of the function to be optimized and hands-on experience with BO on similar tasks in the past. Such knowledge or experience can give intuitions about a functional form of the problem and thus specifications of a functional prior (Turner et al., 2021). However, without domain knowledge or hands-on experience, BO performance can be susceptible to misspecified priors (Schulz et al., 2016) unless the misspecification is insignificant (Bogunovic and Krause, 2021).

Similar to how practitioners learn how to set good priors from past experience, we automate the prior determination process by *pre-training priors* on data that are available on different but

related tasks. Given the benefit of pre-trained priors on synthetic functions, simple tuning tasks and complex robotics tasks (Wang et al., 2018b; Kim et al., 2019; Perrone et al., 2018), can we take it to the level of real-world hyperparameter tuning problems for modern deep learning models (e.g. ResNet50) and large-scale datasets (e.g. ImageNet)? To the best of our knowledge, there is no such multi-task tuning benchmark available for modern large models and datasets, but this kind of tuning task is most prevalent in hyperparameter tuning in recent years of ML-related publications given the success of large models (He et al., 2016; Raffel et al., 2020; Brown et al., 2020). For these problems, it is difficult to understand the landscapes of tuning objectives, hindering the use of Bayesian optimization with almost unobtainable expert interventions on priors. To fill the vacancy of a dataset for hyperparameter tuning in modern ML, we collected a large multi-task hyperparameter tuning dataset by training tens of thousands of configurations of near-state-of-the-art models on popular image and text datasets, as well as on a protein sequence dataset. Our open-sourced dataset can save roughly 12,000 machine-days of computation for anyone who makes use of it.

On the modeling side, most existing meta BO methods either scale cubically in the number of evaluations and tasks (Swersky et al., 2013; Bardenet et al., 2013), impose a restrictive set of assumptions on the available data (Wang et al., 2018b; Swersky et al., 2013) for efficient solutions, or make assumptions on the availability of GP parameters (Volpp et al., 2020) or descriptive task-level features (Brazdil et al., 1994; Bardenet et al., 2013; Yogatama and Mann, 2014). To address these issues, we introduce HyperBO: a meta BO method that builds upon Wang et al. (2018b) with a simple assumption: all the related functions being optimized are samples from the same GP prior distribution. Concretely, HyperBO assumes the functions are conditionally independent given the hyperparameters, mean and covariance function of the GP. Compared to Wang et al. (2018b), HyperBO does not impose any strict conditions on data or model structures. From a computational perspective, HyperBO scales linearly in the number of tasks during training, and does not depend on the number of tasks when deployed. By not imposing assumptions about the data collection conditions, it can be used with large offline datasets or a few related optimization trajectories.

Our empirical results show that HyperBO is at least 3 times more efficient in function evaluations than recent baseline methods to locate the best hyperparameters. Our main contributions are two-fold: (1) a new method for BO with learned (as opposed to hand-designed) priors; (2) a large multi-task hyperparameter tuning dataset that not only benefits our method but also serves as a realistic benchmark to test future methods. Both open-sourced code and dataset are available at https://github.com/google-research/hyperbo. Our extended paper (Wang et al., 2022) is available for interested readers to deep dive.

2. Problem formulation

We consider the standard black-box function optimization scenario: given a real-valued function f defined over a compact, hyper-rectangular space $\mathfrak{X} \subset \mathbb{R}^d$ and given observations of similar functions $f_1, \cdots, f_N$, we seek an $x \in \mathfrak{X}$ optimizing f. We inherit our problem formulation from Wang et al. (2018b), but we relax impractical assumptions on data availability (we do not require all observations to be made on the same inputs across tasks) and model restrictions.

Assumptions and the goal. Concretely, we assume that there exists a Gaussian process $\mathcal{GP}(\mu, k)$ with unknown mean function $\mu : \mathfrak{X} \to \mathbb{R}$ and kernel $k : \mathfrak{X} \times \mathfrak{X} \to \mathbb{R}$. Let N be the number of tasks and let M_i be the number of observations we have for the ith task. Conditioned on independent

function samples $f_i \sim \mathcal{GP}(\mu, k)$ and inputs $x_j^{(i)} \in \mathfrak{X}, i \in [N], j \in [M_i]$, we observe evaluations $y_j^{(i)} \sim \mathcal{N}(f_i(x_j^{(i)}), \sigma^2)$ perturbed by *i.i.d.* additive Gaussian noise $\mathcal{N}(0, \sigma^2)$ with unknown variance σ^2. Taken together, the collection of sub-datasets $D_{f_i} = \{(x_j^{(i)}, y_j^{(i)})\}_{j=1}^{M_i}$ define a dataset $D_N = \{D_{f_i}\}_{i=1}^N$. Finally, our goal is to maximize a new function independently sampled from the same GP, $f \sim \mathcal{GP}(\mu, k)$; that is, solve $\arg\max_{x \in \mathfrak{X}} f(x)$ given dataset D_N but unknown functions μ, k and unknown parameter σ^2.

An example. In our optimizer hyperparameter tuning application, a task corresponds to finding the best optimizer hyperparameters to train a given model on a particular dataset,[1] e.g. training a ResNet (He et al., 2016) on ImageNet (Russakovsky et al., 2015). Notice that we do not assume that the mean function μ, kernel k and noise variance σ^2 are given. This is consistent with the reality of solving real-world black-box optimization problems including hyperparameter tuning. We must learn those unknown functions and parameters from data. However, in practice, searching in functional spaces to find the right mean μ or kernel k is a daunting task. Hence for practical concerns, a well defined search space for functions is required. More details on this can be found at §3.2.

Metrics. For simplicity, throughout this paper, we focus on the setting where the target function f can only be optimized by iteratively choosing where to evaluate, and defer batch evaluation setups to Sec. B. As we run BO on the target function f for T iterations, we accumulate a set of observations $D_f = \{(x_t, y_t)\}_{t=1}^T$, $y_t \sim \mathcal{N}(f(x_t), \sigma^2)$. We evaluate the quality of the optimization using the *simple regret* metric: $R_T = \max_{x \in \mathfrak{X}} f(x) - f(\hat{x})$, where $\hat{x}$ is the final recommendation at the end of the optimization process. There are various ways of setting $\hat{x}$ based on the observations D_f; we use the input that achieved the best evaluation: $\hat{x} = x_\tau; \tau = \arg\max_{t \in [T]} y_t$.

Notations. Let $[n]$ denote $\{1, \cdots, n\}, \forall n \in \mathbb{Z}^+$. For conciseness, we write the evaluation of a function $f : \mathbb{R} \to \mathbb{R}$ on vector $\boldsymbol{x} = [x_i]_{i=1}^n$ as $\mu(\boldsymbol{x}) := [\mu(x_i)]_{i=1}^n$. Similarly, for two vectors $\boldsymbol{x}, \boldsymbol{x}'$, we write the corresponding kernel matrix as $k(\boldsymbol{x}, \boldsymbol{x}') := [k(x_i, x_j')]_{i \in [n], j \in [n']}$, and shorten $k(\boldsymbol{x}) := k(\boldsymbol{x}, \boldsymbol{x})$.

We denote a (multivariate) Gaussian distribution with mean u and variance Σ by $\mathcal{N}(u, \Sigma)$, and a Gaussian process (GP) with mean function μ and covariance function k by $\mathcal{GP}(\mu, k)$. Let σ^2 be the noise variance in observations. Given a set of observations $D = \{(x_t, y_t)\}_{t=1}^T$, $\boldsymbol{y}_T = [y_t]_{t=1}^T \sim \mathcal{N}(f(\boldsymbol{x}_T), \sigma^2 \boldsymbol{I})$, $\boldsymbol{x}_T = [x_t]_{t=1}^T$ and $f \sim \mathcal{GP}(\mu, k)$, we denote the corresponding conditional GP distribution as $\mathcal{GP}(\mu, k \mid D)$. Recall that the conditional distribution $\mathcal{GP}(\mu, k \mid D) = \mathcal{GP}(\mu_D, k_D)$, is given for any $x, x' \in \mathfrak{X}$ as

$$\mu_D(x) = \mu(x) + \psi(x)(\boldsymbol{y}_T - \mu(\boldsymbol{x}_T)), \tag{1}$$
$$k_D(x, x') = k(x, x') - \psi(x)k(\boldsymbol{x}_T, x'), \tag{2}$$

where we set $\psi(x) = k(x, \boldsymbol{x}_T)(k(\boldsymbol{x}_T) + \sigma^2 \boldsymbol{I})^{-1}$.

3. Our method

As shown in Alg. 1, our approach pre-trains the GP hyperparameters on a representative set of datasets and fixes them for the duration of the optimization procedure; we refer to this approach as

---

1. Technically, we also consider different batch sizes to be different tasks.

HyperBO. HyperBO runs in two steps. First, we learn a GP model $\mathcal{GP}(\hat{\mu}, \hat{k})$ to approximate the ground-truth (unknown) GP that generated the dataset D_N. Then, we do standard BO to optimize a new function f with the learned GP $\mathcal{GP}(\hat{\mu}, \hat{k})$. The initial pre-training process (Alg. 1, line 2) is the critical difference between HyperBO and standard BO algorithms, as well as the key contribution of this paper.

In each iteration of the BO loop (Alg. 1, lines 4-8), we update the conditional GP, but do not re-estimate the GP mean and kernel. By separating the data for conditional GP update and GP parameter pre-training, we minimize the computational cost while still maintaining good performance empirically. Moreover, we avoid the BO chicken-and-egg dilemma (Wang et al., 2018b) where the search strategy is trained on data collected in the BO process and the data points are selected by the search strategy simultaneously.

Algorithm 1 HyperBO with acquisition function $\alpha(\cdot)$.

1: **function** HYPERBO (f, D_N)
2: $\quad$ $\mathcal{GP}(\hat{\mu}, \hat{k}) \leftarrow$ PRE-TRAIN(D_N)
3: $\quad$ $D_f \leftarrow \emptyset$
4: $\quad$ **for** $t = 1, \cdots, T$ **do**
5: $\quad\quad$ $x_t \leftarrow \underset{x \in \mathfrak{X}}{\arg\max}\, \alpha\left(x; \mathcal{GP}(\hat{\mu}, \hat{k} \mid D_f)\right)$
6: $\quad\quad$ $y_t \leftarrow$ OBSERVE$(f(x_t))$
7: $\quad\quad$ $D_f \leftarrow D_f \cup \{(x_t, y_t)\}$
8: $\quad$ **end for**
9: $\quad$ **return** D_f
10: **end function**

Next, we introduce our GP pre-training strategy based on two types of objectives: KL divergence between estimates and model predictions (§ 3.1) and negative log likelihood (§ 3.2).

3.1 Pre-training with empirical KL divergence

We first investigate the case where observations on the same set of inputs are available. This is the main scenario considered by Wang et al. (2018b), but their method only works for Bayesian linear regression. We now present HyperBO based on an empirical KL divergence to allow highest flexibility on mean functions and kernels, e.g. a Matérn kernel on deep features shared with a mean function. The objective is called *empirical KL divergence* because it is the KL divergence between an empirically estimated multivariate Gaussian and model predictions from a GP.

Here we consider a special case of dataset D_N which contains matching inputs across some tasks. More formally, suppose we have a *matching dataset* $D'_N = \{(x_j, \boldsymbol{y}_j)\}_{j=1}^M$ where M is the number of shared inputs across N tasks. For each input index $j \in [M]$ and input $x_j \in \mathfrak{X}$, we have N observed values $\boldsymbol{y}_j = [y_j^{(i)}]_{i=1}^N \in \mathbb{R}^N$ and each observation $y_j^{(i)} \sim \mathcal{N}(f(x_j), \sigma^2)$ corresponds to evaluating input x_j on a different task. In practice, dataset D'_N can be constructed by querying a set of functions $f_1, \cdots, f_N$ at the same set of input locations $\boldsymbol{x} = [x_j]_{j=1}^M \in \mathbb{R}^{M \times d}$ to obtain an observation matrix $\boldsymbol{y} = [\boldsymbol{y}_j]_{j=1}^M \in \mathbb{R}^{M \times N}$.

By definition of a GP, the vector of all function queries $f(\boldsymbol{x})$ is distributed according to a multivariate Gaussian distribution $\mathcal{N}(\mu(\boldsymbol{x}), k(\boldsymbol{x}))$. With our observation model, we get the distribution for observations $\boldsymbol{y} \sim \mathcal{N}(\mu(\boldsymbol{x}), k(\boldsymbol{x}) + \boldsymbol{I}\sigma^2)$ for some unknown mean function μ and kernel k.

However, given that we have access to all observations $\boldsymbol{y}$, we can estimate the mean on inputs $\boldsymbol{x}$ as $\tilde{\boldsymbol{\mu}} = \frac{1}{N} \boldsymbol{y} 1_N \in \mathbb{R}^M$ and estimated covariance as $\tilde{K} = \frac{1}{N}(\boldsymbol{y} - \tilde{\boldsymbol{\mu}} 1_N^\top)(\boldsymbol{y} - \tilde{\boldsymbol{\mu}} 1_N^\top)^\top \in \mathbb{R}^{M \times M}$; here 1_N is a column vector of size N filled with 1s. We use a biased estimate of covariance to be consistent with the maximum likelihood estimator in §3.2. But one may choose to re-scale learned kernel by $\frac{N}{N-1}$ to be unbiased. Notice that the estimated covariance includes in diagonal terms the variance of the observation noise.

4

For any divergence function between the estimate $\mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{K})$ and model prediction $\mathcal{N}(\mu(\boldsymbol{x}), k(\boldsymbol{x}) + \boldsymbol{I}\sigma^2)$, we obtain an objective to minimize, $\mathcal{D}\left(\mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{K}), \mathcal{N}(\mu(\boldsymbol{x}), k(\boldsymbol{x}) + \boldsymbol{I}\sigma^2)\right)$. While there are different measures of distributional discrepancy, we adopt the KL divergence. Let $\boldsymbol{\mu} = \mu(\boldsymbol{x})$ and $K = k(\boldsymbol{x}) + \boldsymbol{I}\sigma^2$. The empirical KL divergence is defined as

$$\mathcal{D}_{\mathrm{KL}}\left(\mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{K}), \mathcal{N}(\boldsymbol{\mu}, K)\right) = \frac{1}{2}\left(\mathrm{tr}(K^{-1}\tilde{K}) + (\boldsymbol{\mu} - \tilde{\boldsymbol{\mu}})^\top K^{-1}(\boldsymbol{\mu} - \tilde{\boldsymbol{\mu}}) + \ln\frac{|K|}{|\tilde{K}|} - M\right), \quad (3)$$

and we can estimate the mean, kernel and noise variance by minimizing $\mathcal{D}_{\mathrm{KL}}$.

3.2 Pre-training with negative log likelihood

If we have arbitrary data points from each task, a straightforward way to pre-train a GP is by optimizing a negative log likelihood (NLL) over parameters of the GP. The regression-based NLL objective corresponds to how supervised pre-training is done with a cross-entropy loss on deep learning models for classification tasks. Here, we use the NLL on the given observations from multiple functions that are assumed to be independently sampled from the GP. The independence naturally results in a summation over NLLs for all observed functions, which is a key difference to the widely used type II maximum likelihood approximation for GP inference on a single function in BO setups. The NLL loss function for our method is $L(\mu, k, \sigma^2) = -\sum_{i=1}^{N} \log p(D_{f_i} \mid \mu, k, \sigma^2)$. We then obtain a solution to the choice of mean function, kernel function and noise variance by minimizing the NLL loss function.

4. Experiments

Our goal in this paper is to provide a practical approach for hyperparameter optimization when we are given data on a range of tasks over the same search space. To analyze the effectiveness of our proposal, we take the optimizer hyperparameter tuning problem in deep learning as a case study. We collected a dataset composed of hyperparameter evaluations on various deep neural network training tasks. The tasks included optimizing deep models on image, text, and other datasets. Our code and dataset can be found at https://github.com/google-research/hyperbo.

To reduce ambiguity, we distinguish between datasets that individual neural networks are trained on and the dataset we collected that includes optimizer hyperparameter points with their validation errors (and other metrics). We will call the former (e.g. MNIST, CIFAR10) task datasets and call the latter the tuning dataset. The tuning dataset is what we described as dataset D_N in §2.

For HyperBO, we used thresholded probability of improvement as the acquisition function and GP with a one-hidden layer neural network of size 8 as mean function and a Matérn32 covariance on the feature layer of the mean function as kernel. H* NLL refers to HyperBO with GPs pre-trained via the NLL objective in §3.2 and H* KL refers to pre-training via the KL objective in §3.1.

Our baselines include (1) Rand: Random search in the corresponding scaled search space. (2) STBO: Single-task BO where in every BO iteration, STBO uses the same GP architecture as HyperBO but optimizes the GP hyperparameters via type II maximum likelihood on data of the test task. This implementation corresponds to the basic off-the-shelf BO setups. (3) STBOH: Single-task GP-UCB with a *hand-tuned* prior on hyper-parameters including UCB coefficient Srinivas et al. (2010); Golovin et al. (2017). (4) MIMO: Multi-task BO with GP bases as an ensemble of feedforward neural networks with shared subnetworks (Kim et al., 2021; Havasi et al., 2021). (5) RFGP: Multi-task BO with GP bases as random features (Snoek et al., 2015; Krause and Ong,
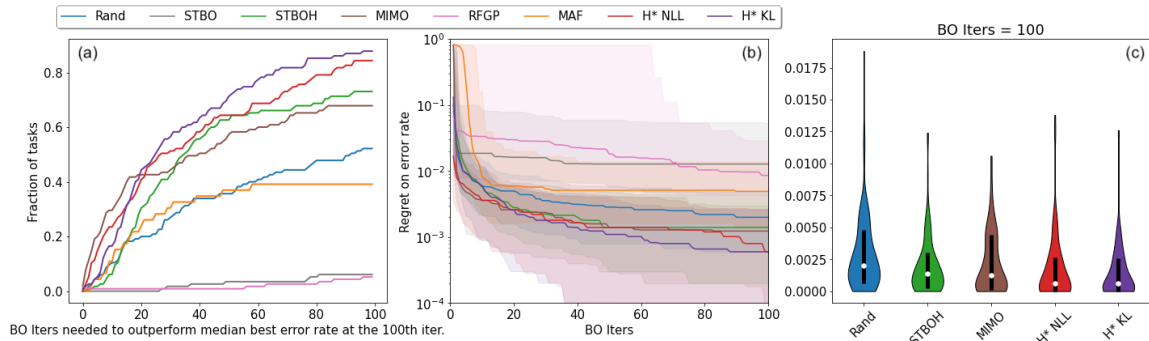
Figure 1: (a) The performance profile for outperforming the median of best error rates at the 100th BO iteration. (b) BO convergence of all methods: the median and 20/80 percentiles of the regrets on error rates over 115 BO runs: 23 tasks and each with 5 repeats of different random seeds. (c) A violin plot on the vertical slices of (b) at the 100th iteration; the white dot is the median and the black line is the 20/80 percentile. Overall, HyperBO methods H* NLL and H* KL are able to achieve the lowest regret on error rate on the majority of tasks with fewer iterations.

2011). (6) MAF: Meta acquisition function method based on reinforcement learning Volpp et al. (2020), which assumes the knowledge of the best GP hyperparameters for each task.

Fig. 1 (a) shows the *performance profiles*, the fraction of all test tasks that each method is able to outperform a baseline criterion at each BO iteration. MIMO is able to outperform other methods in the beginning 20 BO iterations, but its leading position soon gets surpassed by HyperBO (H* NLL and H* KL). Fig. 1 (b,c) illustrates the BO convergence curves of all competing methods, together with the vertical slice at the 100th iterations. MAF, RFGP and STBO are both falling much behind Rand. Surprisingly, the contextual information learned by RFGP did not generalize to a new task. On the other hand, MIMO is able to obtain a slightly better error rate than STBOH. Overall, learning the GP prior through data as with HyperBO outperforms other meta BO methods, and is a more principled and effective way to obtain the GP prior when compared with hand-tuning.

To more precisely quantify HyperBO's advantage, we also computed how much faster HyperBO can get a better error rate than best alternatives, which can be different from task to task. We found that on average, on over 50% tasks, H* NLL is at least 2.86 times faster than best non-HyperBO alternatives; while on over 57% tasks, H* KL is at least 3.26 times faster than best non-HyperBO alternatives. Moreover, on over 73% tasks, H* NLL is at least 7.74 times faster than random search; and on over 75% tasks, H* KL is at least 6.07 times faster than random search.

5. Conclusion

We proposed HyperBO: a novel meta BO approach that supports practical applications that involve continuous inputs queried at possibly non-aligned locations across tasks. HyperBO uses a simple yet effective idea that is easy to implement and efficient to run. We evaluated HyperBO on real-world big model optimizer tuning tasks, and the results demonstrated its superior performance over state-of-the-art competing methods.

References

Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *International Conference on Machine Learning (ICML)*, 2013.

J Baxter. A Bayesian/information theoretic model of bias learning. In *Conference on Learning Theory (COLT)*, New York, New York, USA, 1996.

Ilija Bogunovic and Andreas Krause. Misspecified Gaussian process bandit optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Pavel Brazdil, João Gama, and Bob Henery. Characterizing the applicability of classification algorithms using meta-level learning. In *European Conference on Machine Learning (ECML)*, 1994.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Danny Drieß, Peter Englert, and Marc Toussaint. Constrained Bayesian optimization of combined interaction force/task space controllers for manipulations. In *International Conference on Robotics and Automation (ICRA)*, pages 902–907. IEEE, 2017.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Elliot Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2017.

Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 11(2):577–586, 2020.

Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations (ICLR)*, 2021.

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

Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched Gaussian process bandit optimization via determinantal point processes. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Beomjoon Kim, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning to guide task and motion planning using score-space representation. In *International Conference on Robotics and Automation (ICRA)*, 2017.

Beomjoon Kim, Zi Wang, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning to guide task and motion planning using score-space representation. *International Journal of Robotics Research (IJRR)*, 38(7):793–812, 2019.

Samuel Kim, Peter Y Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljačić. Scalable and flexible deep Bayesian optimization with auxiliary information for scientific problems. *arXiv preprint arXiv:2104.11667*, 2021.

Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka: Automatic model selection and hyperparameter optimization in weka. In *Automated Machine Learning*, pages 81–95. Springer, Cham, 2019.

Andreas Krause and Cheng S Ong. Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in Bayesian optimization with applications in aerospace engineering. In *AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

ChangYong Oh, Efstratios Gavves, and Max Welling. Bock: Bayesian optimization with cylindrical kernels. In *International Conference on Machine Learning (ICML)*, 2018.

Valerio Perrone, Rodolphe Jenatton, Matthias Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6846–6856, 2018.

Matthias Poloczek, Jialei Wang, and Peter I Frazier. Warm starting Bayesian optimization. In *Winter Simulation Conference (WSC)*. IEEE, 2016.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Edward O Pyzer-Knapp. Bayesian optimization for accelerated drug discovery. *IBM Journal of Research and Development*, 62(6):2–1, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015.

David Salinas, Huibin Shen, and Valerio Perrone. A quantile-based approach for hyperparameter transfer learning. In *International Conference on Machine Learning (ICML)*, 2020.

Eric Schulz, Maarten Speekenbrink, José Miguel Hernández-Lobato, Zoubin Ghahramani, and Samuel J Gershman. Quantifying mismatch in Bayesian optimization. In *NeurIPS Workshop on Bayesian Optimization: Black-box Optimization and Beyond*, 2016.

Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning (ICML)*, pages 2171–2180. PMLR, 2015.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, pages 3–26. PMLR, 2021.

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 (ICLR)*, 2020.

Zi Wang, Stefanie Jegelka, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Focused model-learning and planning for non-Gaussian continuous state-action systems. In *International Conference on Robotics and Automation (ICRA)*, 2017a.

Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional Bayesian optimization via structural kernel learning. In *International Conference on Machine Learning (ICML)*, 2017b.

Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018a.

Zi Wang, Beomjoon Kim, and Leslie Pack Kaelbling. Regret bounds for meta Bayesian optimization with an unknown Gaussian process prior. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018b.

Zi Wang, George E Dahl, Kevin Swersky, Chansoo Lee, Zelda Mariet, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-trained Gaussian processes for Bayesian optimization. *arXiv preprint arXiv:2109.08215*, 2022.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal or Artificial Intelligence Research (JAIR)*, 55:361–387, 2016.

Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations (ICLR)*, 2021.

Dani Yogatama and Gideon Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

Appendix A. Related work

There is a rich literature of innovative methodologies to improve the efficiency of BO given related tasks or additional context. Here we discuss the most closely related work and explain why these don't solve the specific scenario which we envision. Specifically, our goal is a methodology that is scalable enough to share information across thousands of tasks, each with potentially hundreds of observations, such as in the context of a large BO service or library.

Pre-training and prior learning is directly related to meta learning, learning to learn and learning multiple tasks (Baxter, 1996). We use the word pre-training to refer to supervised pre-training, which is a general approach in the deep learning community (Girshick et al., 2014) to transfer knowledge from prior tasks to a target task. The same as pre-training deep features on a variety of tasks, Wang et al. (2018b) proposed prior learning for GPs to learn the basis functions by treating the independent function outputs as individual heads of a neural network.

Several methods, including that which HyperBO extends, refer to their method as "meta-BO" (Wang et al., 2018b; Volpp et al., 2020). However, in this work we use the term *meta-BO* more generally to refer to the class of BO methods that use data from existing tasks to optimize a new task. Since standard BO is a learning process, it is consistent to call those methods meta BO methods given that they learn how to learn. Under this viewpoint, meta BO approaches also include multi-task BO (Swersky et al., 2013; Poloczek et al., 2017; Yogatama and Mann, 2014), transfer learning BO using contextual GPs (Krause and Ong, 2011; Bardenet et al., 2013; Poloczek et al., 2016) and transfer learning based on quantiles (Salinas et al., 2020). Some meta BO methods have also been studied for hyperparamter tuning tasks in machine learning (Feurer et al., 2015; Salinas et al., 2020).

HyperBO assumes all tasks are independent (after conditioning on the GP), whereas both multi-task and contextual BO rely heavily on the assumption that tasks are related. Thus the latter approaches typically scale cubically in both the number of tasks and observations in each task, meaning that they cannot gracefully scale across both without heavy approximations. When assuming that all inputs are equal across tasks, multi-task BO can be sped up using a Kronecker decomposition of the kernel to a task kernel and an input kernel which can be inverted separately; a similar assumption is made by Wang et al. (2018b). In comparison, HyperBO scales linearly in the number of tasks.

Another thread of meta BO literature was started in the robot learning area by Kim et al. (2017, 2019), which estimated a multivariate Gaussian to transfer knowledge on scoring functions for search strategies in robot manipulation tasks, and thus only considered finite discrete inputs. Wang et al. (2018b) provided regret bounds for Kim et al. (2017, 2019) and extended it to continuous search spaces by considering a GP as a Bayesian linear regressor with neural net basis functions.

Similar ideas were adopted by Perrone et al. (2018); Wistuba and Grabocka (2021) in the machine learning hyperparameter tuning literature. These ideas, on a high level, can be viewed as special cases of HyperBO or Wang et al. (2018b). Compared to Wang et al. (2018b), Perrone et al. (2018) and Wistuba and Grabocka (2021) only use zero means, which, as shown by Kim et al. (2017, 2019), is critical for learning the initial data points to acquire. As a remedy, Wistuba and Grabocka (2021) developed an evolutionary algorithm based data-driven strategy to warm start the initialization of data selection. Although different terms are used, Wang et al. (2018b) and Perrone et al. (2018) concurrently proposed the idea of learning parameters of GP priors from multi-task datasets, while Wang et al. (2018b) is the first to clarify the assumptions that those multi-task functions need to be conditionally independent so that regret bounds hold for BO with an unknown GP prior.

HyperBO builds upon Wang et al. (2018b) and Kim et al. (2017, 2019), yet principally resolves their limitations on search spaces and data availability. For both finite discrete search spaces and continuous ones, Wang et al. (2018b) requires observations on the same set of inputs across tasks, which is an assumption that is not required for HyperBO. Both arbitrary data points from different tasks and observations on same inputs across tasks can be incorporated into HyperBO efficiently and effectively. Another critical advantage of HyperBO is accommodations of very flexible kernels and mean functions that are not limited by Bayesian linear regressors; this opens meta BO to a lot more GP architectures involving combinations of deep features and kernels with infinite basis functions.

Appendix B. Discussion

In this work, we focused on the question of how to efficiently and effectively make use of multi-task data to enable better BO with pre-trained priors. We simplified other aspects of BO that are orthogonal to our focuses, such as parallel queries or different search spaces. Here we discuss extensions to our work that would enable even more flexible uses.

Batch evaluation. For simplicity of this paper, we did not consider batch evaluation but rather only focused on the prior selection dimension of the challenges in BO. However, it is straightforward to adopt any batch BO methods in conjunction with HyperBO to support obtaining observations in parallel. For example, we can directly use batch methods from Snoek et al. (2012); Kathuria et al. (2016); Wang et al. (2017b) etc. to replace line 5 of Alg. 1.

High-dimensional and large scale data. Similar to batch BO, our method can also be naturally combined with most high-dimensional and large scale BO methods to offer more capabilities. For these cases, typically a probabilistic model different from vanilla GPs may be adopted. In line 2 of Alg. 1, it is straightforward to adapt our method to optimize the cumulative marginal likelihood in §3.2 instead for the new model. Our meta-learning idea in this paper in fact also brings benefit to high-dimensional and large scale BO methods so that they can better identify their critical special structures, e.g. low-dimensional embedding (Wang et al., 2016), cylindrical kernels (Oh et al., 2018) or additive Mondrian kernels (Wang et al., 2018a).

Different search spaces. Roughly speaking, there could be two circumstances for difference search spaces. Case I is that tasks share the same search variables, but the search ranges for some variables are different. For example, we may have each function $f_i : \mathfrak{X}_i \rightarrow \mathbb{R}, i \in [N]$ and $\mathfrak{X}_i = \prod_{j=1}^{d}[l_{ij}, h_{ij}] \subset \mathbb{R}^d$. In this case, our solution still applies by simply setting a union search space as $\mathfrak{X} = \bigcup_{i=1}^{N} \mathfrak{X}_i$ for learning and use the designated search space of new tasks for optimization.

Case II is more complicated: the search space for each function f_i is $\mathfrak{X}_i \subset \mathbb{R}^{d_i}$ and each dimension of $\mathfrak{X}_i$ may have a different meaning than another search space $\mathfrak{X}_j$ $(i \neq j)$. This paper does not have a solution for this scenario. Further research will be needed to reduce Case II to Case I which can be then immediately combined with HyperBO.