Transfer Learning with Gaussian Processes for Bayesian Optimization

Petru Tighineanu **Kathrin Skubch** **Paul Baireuther**
Attila Reiss **Felix Berkenkamp** **Julia Vinogradska**
Bosch Center for Artificial Intelligence, Renningen, Germany

Abstract

Bayesian optimization is a powerful paradigm to optimize black-box functions based on scarce and noisy data. Its data efficiency can be further improved by transfer learning from related tasks. While recent transfer models meta-learn a prior based on large amount of data, in the low-data regime methods that exploit the closed-form posterior of Gaussian processes (GPs) have an advantage. In this setting, several analytically tractable transfer-model posteriors have been proposed, but the relative advantages of these methods are not well understood. In this paper, we provide a unified view on hierarchical GP models for transfer learning, which allows us to analyze the relationship between methods. As part of the analysis, we develop a novel closed-form boosted GP transfer model that fits between existing approaches in terms of complexity. We evaluate the performance of the different approaches in large-scale experiments and highlight strengths and weaknesses of the different transfer-learning methods.

1 INTRODUCTION

Bayesian optimization (BO) is an elegant and powerful approach to black-box optimization. It has been successfully applied to several challenging black-box optimization problems, ranging from hyperparameter optimization (Snoek et al., 2012) to materials design (Zhang et al., 2020) and controller tuning (Calandra et al., 2016). One major advantage of BO is sample efficiency: it is specifically tailored to expensive black-box functions, where each function evaluation is costly, e.g., when laborious experiments on physical systems are involved. For such applications, the benefit of fewer function evaluations outweighs the increase in computational cost to make informed decisions about where to measure next. A key ingredient to BO's sample efficiency is a probabilistic surrogate model of the objective. In the low-data regime, the most commonly employed model is a Gaussian process (GP) (Rasmussen and Williams, 2006), as it provides closed-form posteriors with accurate uncertainty estimates to guide the search for the global optimum.

Many black-box optimization problems are not one-off tasks, but rather several related instances of the tasks can be encountered. Here, the data efficiency of optimization can be further improved by transferring knowledge from related tasks. Especially in the low-data regime that BO is specialized on, transfer learning provides great value. To this end, Swersky et al. (2013); Joy et al. (2016); Shilton et al. (2017); Golovin et al. (2017) propose several approaches to transfer learning for BO. These transfer learning models combine related task data and the data of the current task into a joint model that guides BO in the search for the current task's global optimum. This setting implies a certain asymmetry: while we may use data from related tasks, we are only interested in informative models for the current task. Further, the related task data are considered as given and additional samples cannot be acquired. One common approach to account for this asymmetry, often referred to as *hierarchical model*, is to model the difference between the current and related tasks. However, when combining different task data into a single GP model, the computational complexity scales cubically with the number of data points. This complexity can easily become prohibitive if data from multiple related tasks are available. Several other transfer models with tractable posteriors and more favourable scaling have been proposed, e.g., an ensemble of GPs (Feurer et al., 2018) or a hierarchical model for the mean prior (Golovin et al., 2017). However, their relation to the joint models with full cu-

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

bic complexity, their relative strengths and weaknesses, and underlying assumptions are not well understood.

2 RELATED WORK AND CONTRIBUTIONS

The problem of speeding up Bayesian optimization by using additional information besides the objective function evaluations is a focus topic in the black-box optimization community. A large body of literature promotes the idea to meta-learn models (Finn et al., 2017; Perrone et al., 2018a; Flennerhag et al., 2019), whole optimizers (Chen et al., 2017) or acquisition functions (Volpp et al., 2020) for BO. These methods are powerful, but require large amounts of training data to extract the characteristics of a class of tasks.

In the low-data regime, methods that rely on the uncertainty estimates of a GP posterior to guide the search outperform large-scale meta-learning. Several GP-based transfer learning approaches have been proposed, which can mostly be summarized into two categories that either *(i)* build a global GP model, or *(ii)* maintain separate GPs for each data set. Several options exist to aggregate this data into a single global model: multitask GPs by Swersky et al. (2013); Yogatama and Mann (2014) model correlations between tasks, Poloczek et al. (2017) model task biases with a joint linear coregionalization kernel and some independence assumptions, Joy et al. (2016) propose envelope GPs that adapt the noise model to accommodate all data, and DiffGPs by Shilton et al. (2017) regress on prediction bias corrected related task data.

In multi-fidelity BO, which only differs from the transfer learning setting by removing the constraint that related task data cannot be acquired during optimization, Forrester et al. (2007); Marco et al. (2017) successfully employ GP models. All these approaches ultimately compute a GP model on a data set of the size of the joint related and target task data sets and suffer from the cubic complexity. On the other hand, Golovin et al. (2017), Feurer et al. (2018) and Wistuba et al. (2016) maintain separate GPs for each data set to avoid the accumulated scaling. In the hierarchical model presented in Golovin et al. (2017), this reduction in computational complexity stems from neglecting the uncertainty of the source models for the transfer, which is detrimental to the optimization process. Instead, RGPE (Feurer et al., 2018) builds a ranking-based ensemble of the separate GPs, which involves careful tuning of the weight estimator and also does not result in a Bayesian treatment of the uncertainty. In light of the described properties of these two classes of GP-based transfer learning models, the need for models that can bridge the gap between fully considering source uncertainty and not considering it at all becomes evident. Such an intermediate regime may be particularly relevant for optimizing monetarily expensive processes with relatively little duration such as destructive measurements. In the engineering community, efficient hierarchical models have been developed for the special case where related tasks are evaluated on decreasing subsets of inputs, see e.g. Kennedy and O'Hagan (2000); Le Gratiet and Garnier (2014). However, this setting is not suitable for transfer learning from generic historical data.

While computationally efficient approximations to GPs exist that could be applied to any GP model (including those proposed in this paper), e.g. by Lázaro-Gredilla et al. (2010), this is an orthogonal research direction that is agnostic to the structure of the underlying data and treats every data point the same. Contextual optimization (Krause and Ong, 2011) is another related line of research in which context variables influence the process to-be-optimized. By contrast, transfer learning leverages discrete historical tasks as context.

Contributions In this paper, we provide a unifying view on GP based transfer learning models and shed light on their relation and assumptions. This unified view enables a thorough analysis of the methods: their computational complexity, their modelling assumptions and their respective advantages for different transfer scenarios. In addition, we fill the gaps in this unified framework. Firstly, we present a modular version of Hierarchical GP that we name *Sequential Hierarchical GP* with improved scaling while maintaining competitive performance. Secondly, we develop a new transfer learning model that is a middle ground between existing approaches in terms of computational complexity and optimization performance. This method is conceptually inspired by boosting architectures, and yet results in an analytically tractable posterior, which we name *Boosted Hierarchical GP*. A comprehensive empirical evaluation of all methods supports our analytical insights and provides valuable guidance on the different methods' comparative advantages.

3 PROBLEM STATEMENT

Our goal is to find the optimum of a *target* black-box function $f_t \colon D \to \mathbb{R}$, based on noisy observations $\mathcal{D}_t = \{\mathbf{x}_n, y_n\}_{n=1}^{N_t}$ where each observation $y_n = f_t(\mathbf{x}_n) + \varepsilon_n$ is corrupted by i.i.d. zero-mean Gaussian noise, $\varepsilon_n \sim \mathcal{N}(0, \sigma_t^2)$. We model our prior belief over the target function f_t with a GP (Rasmussen and Williams, 2006), $f_t \sim \mathcal{GP}(m, k)$, with mean function $m(\cdot)$ and kernel function $k(\cdot, \cdot)$. Conditioned on the data, the posterior distribution is a GP again with the posterior mean and variance at a query point $\mathbf{x}_*$

$$\begin{aligned}\mathbb{E}\left(f_t \mid \mathbf{x}_*, \mathcal{D}_t\right) &= m(\mathbf{x}_*) + k(\mathbf{x}_*, \mathbf{X}_t) \\ &\quad \times \left(k(\mathbf{X}_t, \mathbf{X}_t) + \sigma_t^2 \mathbf{I}\right)^{-1} \left(\mathbf{y}_t - m(\mathbf{X}_t)\right), \\ \operatorname{var}\left(f_t \mid \mathbf{x}_*, \mathcal{D}_t\right) &= k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{X}_t) \\ &\quad \times \left(k(\mathbf{X}_t, \mathbf{X}_t) + \sigma_t^2 \mathbf{I}\right)^{-1} k(\mathbf{X}_t, \mathbf{x}_*),\end{aligned}$$

where $\mathbf{X}_t = (\mathbf{x}_1, \ldots, \mathbf{x}_{N_t})$ and $\mathbf{y}_t = (y_1, \ldots, y_{N_t})$ is the vector of corresponding, noisy observations. Given this belief over the function f_t, BO aims to actively select a new query point $\mathbf{x}_{N_t+1}$ that is informative about the optimum of f_t. This requires trading off exploitation and exploration and is usually accomplished through an acquisition function α that depends on the posterior,

$$\mathbf{x}_{N_t+1} = \underset{\mathbf{x} \in D}{\operatorname{argmax}}\, \alpha(f_t \mid \mathbf{x}, \mathcal{D}_t). \tag{1}$$

Common choices for α are the expected improvement (Jones et al., 1998) and the upper confidence bound (Srinivas et al., 2010), among several others. The data-efficiency of these methods crucially depends on how fast the posterior distribution collapses around the true target function f_t.

To enable faster optimization, transfer learning additionally exploits data from $n_s \geq 1$ related *source* tasks based on functions $f_s \colon D \to \mathbb{R}$. While all methods discussed in the following apply to this general setting, we focus on $n_s = 1$ for ease of exposition. In this case, we have N_s additional data points $\mathcal{D}_s = \{\mathbf{X}_s, \mathbf{y}_s\}$ that can be used to improve the target model. Further, we assume without loss of generality that the source is modelled with a GP, $f_s \sim \mathcal{GP}(0, k_s)$, with zero mean prior and arbitrary kernel function k_s.

4 GAUSSIAN PROCESS TRANSFER MODELS

In this section, we provide a unified overview of existing GP models for transfer learning that allow for a closed-form posterior distribution.

Multi-Task GP (MTGP) Our starting point is the kernel function k of the *joint* model of source and target. Let $(\mathbf{x}, i), (\mathbf{x}', j)$ be two points from tasks $i, j \in \{s, t\}$. We assume that k is a sum of separable kernels

$$k((\mathbf{x}, i), (\mathbf{x}', j)) = \sum_{\nu \in \{s,t\}} [\mathbf{W}_\nu]_{i,j}\, k_\nu(\mathbf{x}, \mathbf{x}') + \delta_{\mathbf{x}\mathbf{x}'} \delta_{ij} \sigma_i^2, \tag{2}$$

where the dirac-delta function δ_{ij} is equal to one if $i = j$ and zero otherwise, and k_ν are arbitrary kernel functions (Álvarez et al., 2012). The positive semi-definite matrices $\mathbf{W}_\nu$ are often referred to as *coregionalization matrices*, since their diagonal (off-diagonal) entries control correlations within (between) data sets. When counting the hyperparameters, we assume that each kernel function has at least one hyperparameter. For $n_s > 1$ sources, this model has $\mathcal{O}(n_s^3)$ scalar hyperparameters from the matrices $\mathbf{W}_\nu$, $\mathcal{O}(n_s)$ scalar noise hyperparameters σ_i, and, in addition, the hyperparameters of the kernels k_ν. The computational complexity of training this model with a given set of hyperparameters scales as $\mathcal{O}(N^3)$, where N is the total number of data points from all tasks. This makes it expensive and challenging to optimize the hyperparameters of the model. In the following, we introduce several simplifications from the literature that have fewer hyperparameters and/or better scaling properties.

Multi-Task-Single-k GP (MTKGP) A common heuristic is to assume that the source and target functions share a common structure and consider $k_s = k_t$. The joint kernel contains only one coregionalization matrix, which reduces the number of scalar hyperparamters to $\mathcal{O}(n_s^2)$, while the computational complexity is the same as (2).

Weighted Source GP (WSGP) Another common simplification is to set correlations between different source data sets to zero

$$[\mathbf{W}_s]_{i,j} = \delta_{i,s}\delta_{j,s} + w_{st}, \qquad [\mathbf{W}_t]_{i,j} = \delta_{i,t}\delta_{j,t}. \tag{3}$$

The hyperparameters w_{st} quantify how much the source is correlated with the target. Each coregionalization matrix has at most one parameter, which reduces the total parameter number to $\mathcal{O}(n_s)$.

Hierarchical GP (HGP) The asymmetric setting in transfer learning, i.e., modelling the target but not the source tasks, motivates a hierarchical approach to model the differences of target to source data with an additive kernel defined by

$$[\mathbf{W}_s]_{i,j} = 1, \qquad [\mathbf{W}_t]_{i,j} = \delta_{i,t}\delta_{j,t}. \tag{4}$$

As in WSGP, the number of scalar parameters is of order $\mathcal{O}(n_s)$. If optimized jointly, the computational complexity is the same as of MTGP.

Mean Hierarchical GP (MHGP) Golovin et al. (2017) propose a simpler way to transfer information from a pretrained source model to the target and only use the posterior mean of the source model as prior mean for the target. All correlations of the source model are neglected, which leads to

$$[\mathbf{W}_s]_{i,j} = \delta_{i,s}\delta_{j,s} \qquad [\mathbf{W}_t]_{i,j} = \delta_{i,t}\delta_{j,t}. \tag{5}$$

The kernel is block diagonal in the different tasks. The training complexity of MHGP is therefore the complexity of training the target model plus the additional cost of evaluating the source posterior mean at the target points. Note that Golovin et al. (2017) additionally combine source and target uncertainty heuristically, which we do not consider for ease of exposition.

5 INTERMEDIATE-COMPLEXITY TRANSFER MODELS

The Bayesian methods presented above require an expensive optimization of the joint likelihood. In this section, we introduce our novel models, Sequential Hierarchical GP (SHGP) and Boosted Hierarchical GP (BHGP), which leverage the asymmetric setting of transfer learning to lower the complexity.

Sequential Hierarchical GP (SHGP) The starting point is the HGP model from (4) in which both the source, $p(\theta_s|\mathcal{D})$, and target, $p(\theta_t|\mathcal{D})$, model parameters are influenced by *all* task data, $\mathcal{D} = \mathcal{D}_s \cup \mathcal{D}_t$. Here, the quantities θ_s and θ_t parametrize the prior distribution of the GP. The asymmetric nature of transfer learning, in which the target data are accumulated during optimization while the source data remain invariant, motivates the use of a modular Bayesian approach in which the target data do not influence the source model (Bayarri et al., 2009). Training such a model involves (i) finding a suitable $p(\theta_s|\mathcal{D}_s)$ by optimizing the *partial* likelihood, $\mathcal{L}_s$, and (ii) finding $p(\theta_t|\mathcal{D})$ by optimizing the total likelihood while keeping $\mathcal{L}_s$ fixed to the value from the first step. This sequential training procedure is the telltale of SHGP. Under more restrictive assumptions and scope, a similar approach was proposed in Kennedy and O'Hagan (2000); Le Gratiet and Garnier (2014). The modular nature of SHGP leads to a large decrease in training complexity compared to HGP as shown later. The price to pay for the complexity reduction is the inability of the source model to react to the target data, which may become an issue in case $p(\theta_s|D_s)$ is misspecified.

We focus, for concreteness, on hierarchical models in this paper but note that such modularizations, which decrease the training complexity, can, in principle, be conducted for other Bayesian designs from Sec. 4, too. Exploring such approaches is left for future work.

Boosted Hierarchical GP (BHGP) In SHGP, the source model parameters, θ_s, do not depend on the target data but the source posterior distribution does. We propose BHGP, which makes also the latter independent of the target data. This ad-hoc assumption takes BHGP outside the Bayesian realm but provides an elegant connection to boosting, a well-known approach in machine learning that combines an ensemble of weak learners into a strong one (Schapire, 2003). We study this connection in Sec. 6. In contrast to HGP, this leads to an asymmetric treatment of observed target data and query points. In Sec. 6.2 we show that this approach results in the kernel

$$k((\mathbf{x}, i), (\mathbf{x}', j)) = \sum_r [\mathbf{W}_\nu]_{i,j} k_\nu(\mathbf{x}, \mathbf{x}') + \delta_{xx'}\delta_{ij}\sigma_i^2 \quad (6)$$

with $i, j, r \in \{s, t, *\}$, $k_* = k_t + \Sigma_*^{\text{boost}}$, $\sigma_* = 0$, and

$$[\mathbf{W}_*]_{ij} = \delta_{i*}\delta_{j*}, \quad [\mathbf{W}_s]_{ij} = \delta_{is}\delta_{js}, \quad [\mathbf{W}_t]_{ij} = \delta_{it}\delta_{jt}.$$

Note that BHGP introduces an additive term $\Sigma_*^{\text{boost}} = \Sigma_{*,*}^s + \alpha_{*,t}\Sigma_{t,t}^s\alpha_{*,t}^T - \alpha_{*,t}\Sigma_{t,*}^s - \Sigma_{*,t}^s\alpha_{*,t}^T$ to the covariance of the query points, which resembles a robustified version of the target model. Here $\alpha_{*,t} = k_t(\mathbf{x}_*, \mathbf{X}_t)\left(k_t(\mathbf{X}_t, \mathbf{X}_t) + \sigma_t^2\mathbf{I}\right)^{-1}$, and $\Sigma_{t,t}^s$, $\Sigma_{t,*}^s$, $\Sigma_{*,t}^s$, $\Sigma_{*,*}^s$ are the blocks of the posterior covariance matrix of the source evaluated at target and query points. The number of hyperparameters is equal to MHGP, while the computational complexity is the same as of SHGP.

6 THEORETICAL FOUNDATIONS

In Secs. 4 and 5 we introduced several transfer-learning models that rely on the hierarchical kernel architecture. In the following, we present insights on the connection between these methods and their design choices.

Mean Hierarchical GP is the simplest of the models. It trains a GP on the source data and propagates the posterior mean to be the prior mean of the target model. This approach has the lowest computational complexity, see Table 1 (MHGP). Neglecting the transfer of uncertainty from the source to target comes at a cost, as shown in the experimental section (Sec. 7), since well-calibrated uncertainty estimates are at the core of BO's sample efficiency and neglecting the uncertainty may be detrimental to the optimization.

So far we have adopted a Bayesian view when presenting the transfer models in Secs. 4 and 5. This view is, however, not particularly useful for the non-Bayesian nature of BHGP. We therefore switch to an alternative and unifying view in which uncertainty propagates via the stochastic realizations of the models. In particular, we study the creation of an ensemble of target models based on functional samples from the source posterior. Averaging over the ensemble would then lead to the desired target model. In this section, we show that both our proposed transfer-learning models, SHGP and BHGP, are closed-form solutions to such averaging procedures.

Performing model averaging over the ensemble of target *priors* results in the target model of Hierarchical GP, as we show in Sec. 6.1. In light of the ensemble averaging, this kernel naturally lends itself to a sequential optimization of hyperparameters resulting in our proposed Sequential Hierarchical GP. The key advantages

Table 1: Computational complexity for one source task. Training the source model has a complexity of $\mathcal{O}(N_s^3)$. Querying the target has a complexity of $\mathcal{O}(N_t^2 + N_s)$ for MHGP and $\mathcal{O}((N_t + N_s)^2)$ for the other techniques. In Appendix B we discuss the generalization to multiple sources.

Models	Abbreviation	HPs	Joint HPO	Training of target
Multi-Task GP	MTGP	$\mathcal{O}(n_s^3)$	yes	$\mathcal{O}((N_t + N_s)^3)$
Multi-Task-Single-k GP	MTKGP	$\mathcal{O}(n_s^2)$	yes	$\mathcal{O}((N_t + N_s)^3)$
Weighted Source GP	WSGP	$\mathcal{O}(n_s)$	yes	$\mathcal{O}((N_t + N_s)^3)$
Hierarchical GP	HGP	$\mathcal{O}(n_s)$	yes	$\mathcal{O}((N_t + N_s)^3)$
Sequential Hierarchical GP	SHGP	$\mathcal{O}(n_s)$	no	$\mathcal{O}(N_t^3 + N_t^2 N_s + N_t N_s^2)$
Boosted Hierarchical GP	BHGP	$\mathcal{O}(n_s)$	no	$\mathcal{O}(N_t^3 + N_t^2 N_s + N_t N_s^2)$
Mean Hierarchical GP	MHGP	$\mathcal{O}(n_s)$	no	$\mathcal{O}(N_t^3 + N_t N_s)$

of this approach are the accurate uncertainty estimates with reduced computational complexity, see Table 1 (HGP vs SHGP), and the sequential optimization of weakly correlated subsets of hyperparameters, which stabilizes training and may improve performance, see Fig. 4.

By contrast, averaging over the ensemble of target *posterior distributions* leads to Boosted Hierarchical GP, as we show in Sec. 4. Here, each target posterior in the ensemble is a GP with a sample from the source posterior as prior mean function. This model inherits the uncertainty in a non-Bayesian fashion and is fundamentally different from Sequential Hierarchical GP.

Before diving into the theoretical analysis, we illustrate the fundamental differences between the three hierarchical models in Fig. 1. In the plots, we train the techniques on data generated from an Alpine function family, $f(x; c) = x \sin(x + \pi) + cx$, with one input, $x \in (-10, 10)$, and one parameter defining the family, $c \in \mathbb{R}$. The source data are generated uniformly within $x \in (-10, 0)$ with $c = 1/2$, which is why the source posterior distribution is uncertain on the right, $x \in (0, 10)$, see Fig. 1(a). The target data are generated with $c = -1/2$ for $X_t = (1, 2, 3, 4)$. The posterior of Mean Hierarchical GP underestimates the uncertainty of the target function on the right, since its uncertainty originates solely from the target data, see Fig. 1(b). Boosted Hierarchical GP alleviates this shortcoming on the right, while on the left it has the same uncertainty since the source model is confident. The posterior of Sequential Hierarchical GP is Bayesian and correctly captures the variation of the target function on the right, see Fig. 1(c). On the left, Sequential Hierarchical GP follows the source model, since the target points are well explained by the source model alone, see Fig. 1(a), and no extra uncertainty is required.

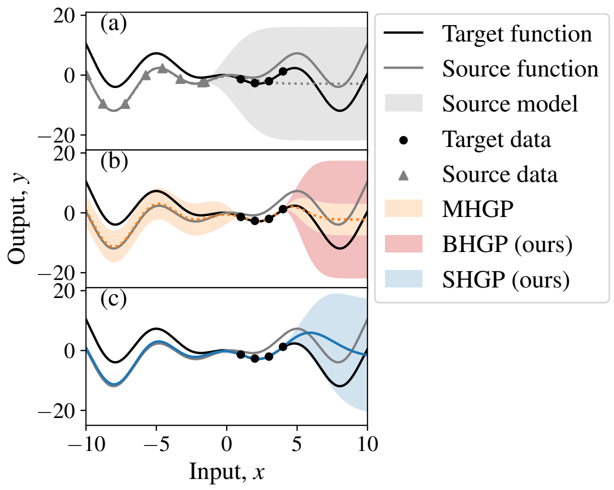

Figure 1: Visualization of key hierarchical transfer-learning models presented in the paper. The posteriors of the source model (a), of MHGP and BHGP (b), and of SHGP (c) are shown in terms of the mean function and 95% confidence intervals.

6.1 Bayesian model averaging

In this section, we show that Bayesian model averaging over prior mean functions of the target, which are distributed according to the posterior of the source, leads to the HGP model defined by (4).

Proposition 1. *Let $f_s \sim \mathcal{GP}(0, k_s)$, $f_t | f_s \sim \mathcal{GP}(f_s, k_t)$, and*

$$p(f_t^{\text{bayes}} \mid \mathcal{D}_t) = \int p(f_t \mid f_s, \mathcal{D}_t) p(f_s \mid \mathcal{D}_t) df_s. \quad (7)$$

Then, the joint model of f_s and f_t^{bayes} is a GP with zero mean prior and the kernel (4)*.*

We provide a closed form solution for the marginalization of (7) over the source posterior in Appendix A. Since both integrands are Gaussian, the integral yields a Gaussian distribution for $p(f_t^{\text{bayes}} \mid \mathcal{D}_t)$. By conditioning the joint model from Proposition 1 on the source data, we can derive the following prior for the target model.

Corollary 1. *Let $k_\Sigma = k_t + \operatorname{cov}(f_s \mid \mathcal{D}_s)$. Then under the assumptions of Proposition 1 it holds that $f_t^{\text{bayes}} \sim \mathcal{GP}\left(\mathbb{E}\left(f_s \mid \mathcal{D}_s\right), k_\Sigma\right)$.*

We provide the detailed derivations in Appendix A. The training complexity scales as $\mathcal{O}(N_t^3 + N_t^2 N_s + N_t N_s^2)$, see Appendix B.5 for the proof. We therefore save, during training, the steep complexity contribution of $\mathcal{O}(N_s^3)$ of the conventional Bayesian methods.

6.2 Posterior prediction averaging

In an alternative approach we take inspiration from the well-known principle of boosting (Schapire, 2003) and average over the posterior distributions of the target models. In contrast to Bayesian model averaging in Sequential Hierarchical GP, this approach neglects the implicit dependency of the source model on the target data, $p(f_s \mid \mathcal{D}_t) \to p(f_s)$ in (7). This distribution is analytically tractable and equals the posterior distribution of a GP with kernel (6), i.e., the Boosted Hierarchical GP.

Proposition 2. *Let f_s, f_t be as in Proposition 1 and*

$$p(f_t^{\text{boost}} \mid \mathcal{D}_t) = \int p(f_t \mid f_s, \mathcal{D}_t) p(f_s) df_s.$$

Then, the joint model of f_s and f_t^{boost} is a GP with zero mean and the covariance function defined in (6). Further, $f_t^{\text{boost}} \mid \mathbf{x}_*, \mathcal{D}_t$ is multi-variate normal with mean $\mathbb{E}\left(f_s \mid \mathbf{x}_*, \mathcal{D}_s\right) - \alpha_{*,t}(\mathbf{y}_t - \mathbb{E}\left(f_s \mid \mathbf{X}_t, \mathcal{D}_s\right))$ and covariance matrix $k_t(\mathbf{x}_*, \mathbf{x}_*) - \alpha_{*,t} k_t(\mathbf{X}_t, \mathbf{x}_*) + \Sigma_*^{\text{boost}}$.

The proof follows the same ideas as for Corollary 1 and can be found in Appendix A. Since the prior for the target of BHGP coincides with the prior of MHGP, the hyperparameter optimization is the same. The training complexity is the same as for SHGP, $\mathcal{O}(N_t^3 + N_t^2 N_s + N_t N_s^2)$, see Appendix B.7 for proof.

7 EXPERIMENTS

We provide an experimental evaluation of the transfer-learning algorithms discussed in Table 1. In addition, we evaluate three baselines: GP-based BO (GPBO) without any source data, RGPE (Feurer et al., 2018) in which the target function is modelled as a weighted sum of the predictions of all task GPs, and GC3P (Salinas et al., 2020) in which a Gaussian Copula regression with a parametric prior is employed to scale to large data. We implement the models using GPy (GPy, since 2012) and run BO with Emukit (Paleyes et al., 2019), licensed under BSD 3 and Apache 2.0, respectively. For GC3P we use the publicly available code (Salinas et al., 2020). GP hyperparameters are optimized by maximizing the likelihood for the observed data. We focus on maximum likelihood for simplicity but, in high dimensions, it may be preferable to employ a Bayesian treatment of hyperparameters. Code to reproduce our results can be found on Github[1].

7.1 Synthetic Function Families

The synthetic function families that are derived from conventional benchmark functions by placing probability distributions on their parameters. We consider a broad spectrum of families, where the optimum of tasks varies locally for some and in the entire domain for others: the one-dimensional Alpine (Momin and Yang, 2013) function, and the multi-dimensional Branin, Hartmann3 and Hartmann6. Details about the function families are provided in Appendix C.1. Performance is reported via mean simple regret over 50 runs together with standard error of the mean.

Single-source benchmark functions We start the experimental analysis in the simplest setting with a single source task. This fundamental regime provides valuable insight into the trade-off between performance and scalability of the transfer learning algorithms. Applications with scarce historical data where the transfer efficiency is important may particularly benefit from this analysis.

The algorithms are benchmarked on three function families, see Fig. 2. Results on two further function families are available in Appendix D.1. A notable difference in the convergence on the different benchmarks can be observed. This is likely to be caused by the different particularities of each function family, which we discuss in Appendix C.1. Despite these differences, there are a few important generalizing features that characterize the algorithms: (i) The Bayesian techniques SHGP, HGP, and WSGP outperform the non-Bayesian algorithms. This is not surprising, since Bayesian models compute highly informed probabilistic predictions. The best and most consistent performers are the HGP and WSGP algorithms. We attribute this to the flexibility of their design. During training, all kernel's hyperparameters are jointly optimized allowing for superior model quality. (ii) Our modular Bayesian technique, SHGP, is competitive and an attractive alternative with lower computational complexity. The other non-Bayesian methods trail behind, which is likely due to their inability to reliably propagate model uncertainty. Among these, our technique, BHGP, performs consistently better than other non-Bayesian techniques

[1] `https://github.com/boschresearch/transfergpbo`

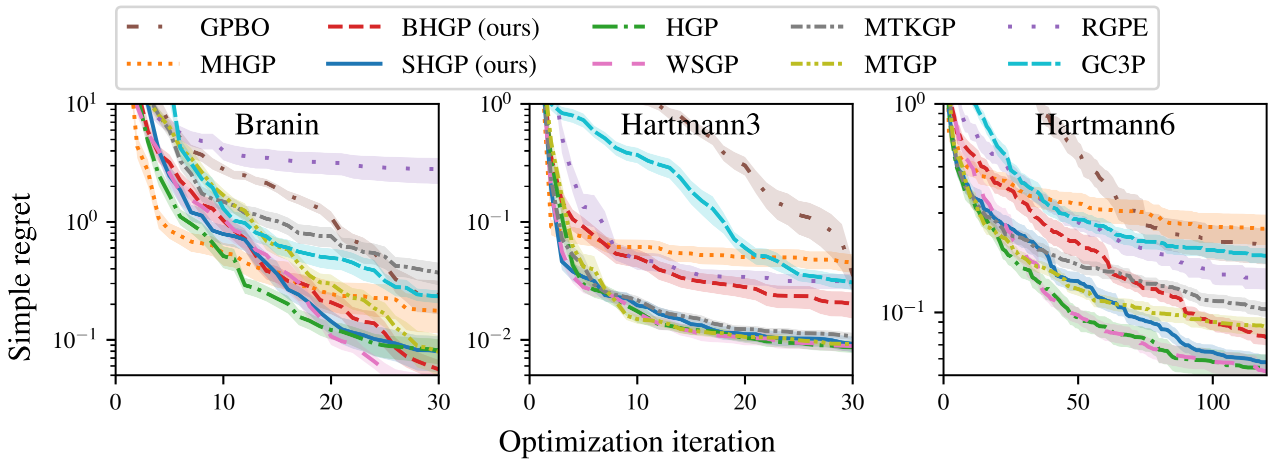

Figure 2: Performance on the single-source benchmarks: the two-dimensional Branin (left), three-dimensional Hartmann3 (middle), and six-dimensional Hartmann6 (right). The source data are sampled randomly from the source function and contain $20N_{\text{dim}}$ points with N_{dim} being the input dimension of the benchmark. We add i.i.d. observational noise of standard deviation $\sigma_s = \sigma_t = 0.1$ for Hartmann3, Hartmann6, and $\sigma_s = \sigma_t = 1.0$ for Branin during data generation.

and provides a reasonable compromise between computational complexity and efficiency. (iii) Surprisingly, the most advanced and expensive techniques, MTKGP and MTGP, perform worse than the other Bayesian techniques. This is likely caused by the challenging training procedure, where a large number of hyperparameters are optimized.

In addition to the asymptotic complexities in Table 1, we demonstrate the lower computational complexity of our methods in a direct runtime analysis of model training in Fig. 3(a). SHGP and BHGP are orders of magnitude faster than the full Bayesian methods for moderately large source data sets. The runtimes reported in Fig. 3(a) are representative of the entire optimization runtime since model training has steeper complexity than acquisition-function optimization, see Appendix D.2 for empirical evidence.

Observational noise and propagation of uncertainty Beyond these generic trends, the performance of the techniques depends on other factors like the amount of source data and the magnitude of the observational noise. Such a study is presented in Fig. 3(b, c); more results are available in Appendix D.3. We distinguish three data regimes: (i) In the limit of scarce source data, which is insufficient to describe the global shape of the source function, the kernel methods with joint HPO, HGP and WSGP, outperform the other methods significantly. The reason is that they model all data jointly, in contrast to the methods trained sequentially that end up with source models of poor quality. (ii) The case of moderate amount of source data, which is sufficient to describe the source function probabilistically but not deterministically, is discussed in the previous section. (iii) In the limit of lots of source data, which is sufficient to deterministically describe the function, all transfer-learning methods converge to a similar performance. Here, propagation of uncertainty is not required and the non-Bayesian approaches are appealing due to their favorable scaling.

The observational noise affects the boundary between the aforementioned data regimes. Enhanced noise leads to an increased amount of data required by a model of fixed quality.

Multi-source benchmark functions For the more general case of multiple source data sets, the algorithms are benchmarked on two function families: (i) the six-dimensional Hartmann6 family with three source data sets in which the functions are sampled randomly as in Sec. 7.1, and (ii) the one-dimensional Alpine family with five source data sets, where the source and target functions are fixed as in (Feurer et al., 2018). MTGP is not benchmarked here because of its complexity. The performance of the algorithms on the three-source Hartmann6 is consistent with the single-source benchmarks, compare Figs. 2 and 4. HGP and WSGP still perform best, while our less complex SHGP provides a competitive compromise. The picture changes in the Alpine benchmark, where the more structured approaches to HPO show superior performance. Our techniques, SHGP and BHGP, outperform others despite the lower computational complexity. This is likely due to the stable,

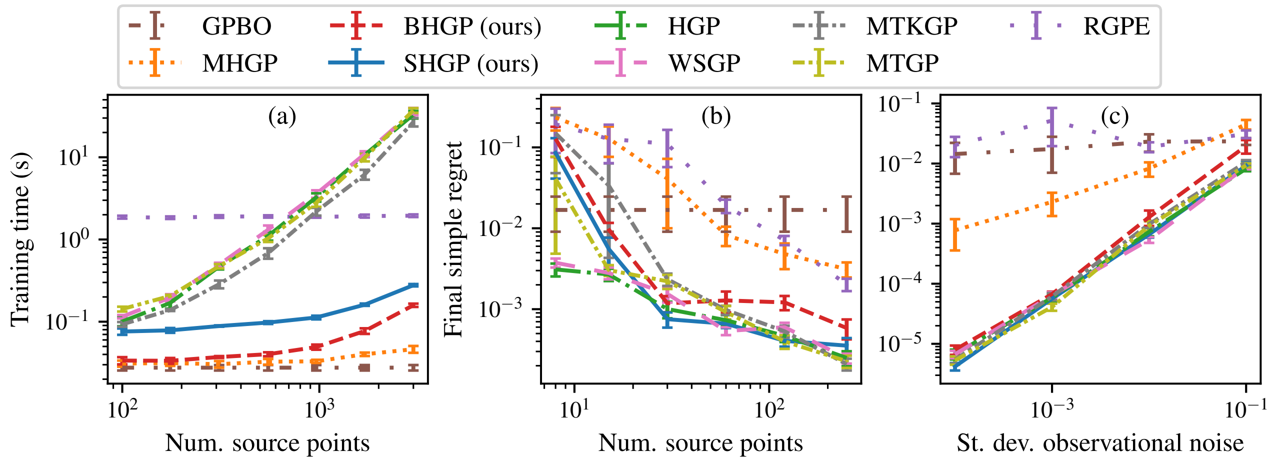

Figure 3: Runtime analysis (a) and the impact of the size of the source data set (b) and observational noise (c) on the algorithm performance. In (a) we plot the training time in seconds versus number of points in the source data set for the Hartmann6 function family. We consider the timing of one single step of gradient ascent during the optimization of the likelihood function. The target data set is set to 100 randomly sampled points. Statistics are acquired via 7 independent runs. The final simple regret versus number of points for constant observational noise, $\sigma_s = \sigma_t = 0.01$ (b), and standard deviation of observational noise for 60 source points (c), is plotted for the Hartmann3 function family. The performance of GPBO is independent of the number of historical points.

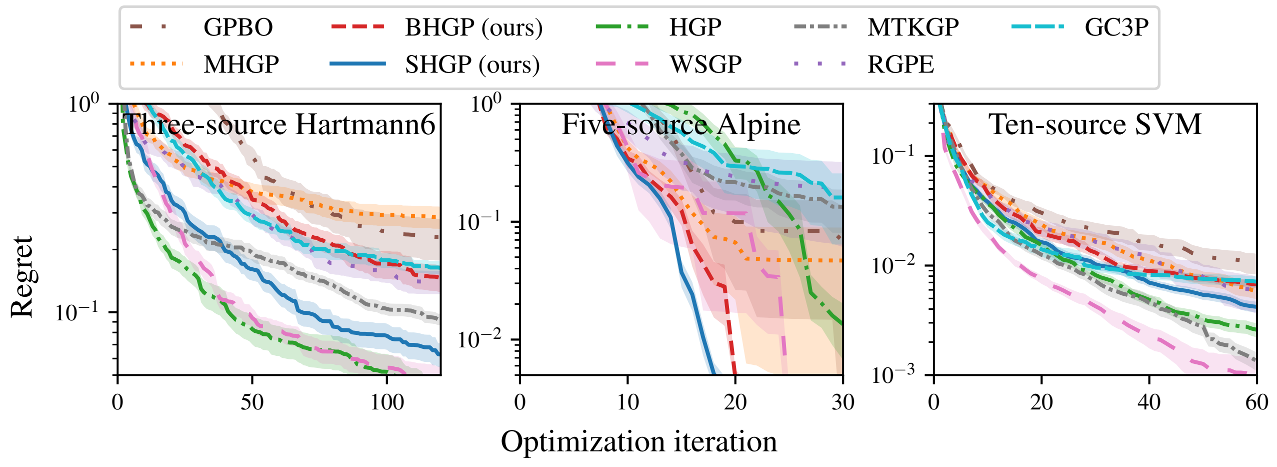

Figure 4: Performance on the multi-source benchmarks: the three-source Hartmann6 (left), five-source Alpine (middle), and ten-source OpenML SVM benchmark (right). The source data are sampled randomly and contain 60, 20, and 60 points per task for Hartmann6, Alpine, and SVM, respectively. I.i.d. observational noise of standard deviation $\sigma_s = \sigma_t = 0.1$ is added to the source and target data of the synthetic benchmarks.

hierarchical training procedure, where at most three hyperparemeters are optimized at a time. By contrast, HGP optimizes 18 hyperparameters jointly, WSGP 23, and MTKGP 33. Algorithms employing hierarchical hyperparameter optimization are therefore ever more appealing for increasing number of source tasks.

7.2 Meta-Learning Surrogate Benchmarks

In the following we study the performance of the transfer-learning techniques on hyperparameter optimization problems in machine learning. We follow Perrone et al. (2018b) and use evaluations of support vector machine (SVM) models on 28 data sets. The data were published by Kühn et al. (2018a) on the OpenML platform (van Rijn et al., 2013) under the creative commons license (Kühn et al., 2018b). More details are available in Appendix C.2. As in Salinas et al. (2020), we carry out a discrete optimization at the observed evaluations to avoid potential problems related to the bias of a fitted surrogate model.

Since the data sets contain over fifty thousand evaluations, which is too much for the closed-form Bayesian techniques to handle, we perform downsampling and choose 11 tasks at random. The data of the target task are used as the ground truth for the discrete optimization, while the data of the source tasks are downsampled to 60 points per task. We present results averaged over 500 independent runs. For better comparability, we employ a rescaled version of the regret function called *average distance to the global minimum* (Wistuba et al., 2016), where the regret values on each data set are rescaled between zero and one *before* the averaging procedure. The results are shown in the right panel of Fig. 4.

Despite the markedly different nature of the data compared to the synthetic benchmarks, the performance of the different transfer-learning techniques exhibits somewhat similar trends. The performance is dominated by WSGP, which achieves achieve a regret of 0.01 more than three times faster than classical GPBO. Interestingly, HGP performs worse. We hypothesize this to be caused by the incompatibility between the hierarchical nature of HGP and the structure of the SVM data set. Our techniques, SHGP and BHGP, provide, as before, a tradeoff between complexity and performance, and achieve the same regret level about twice as fast as GPBO.

8 DISCUSSION AND CONCLUSION

In this paper, we have presented a unified view on transfer learning methods based on Gaussian processes with tractable model posteriors. One end of this spectrum is populated by models that maintain a full covariance matrix of all source and target data and perform proper Bayesian inference on all data jointly. While these are powerful models with well-calibrated uncertainty estimates, they quickly become computationally infeasible due to their cubic complexity in the cumulated number of source and target data points. The other side of the spectrum is occupied by heuristics that aggregate the predictions of separate, independently trained, task models. These methods exhibit a much more favourable computational complexity, but unfortunately fail to properly propagate uncertainty between the individual task models. To amend this unsatisfactory binary choice between computational feasibility and uncertainty propagation, we proposed two novel transfer learning models, Sequential Hierarchical GP and Boosted Hierarchical GP, that both provide a compromise between these extremes: they add some computational complexity to account for the uncertainty of the involved models with some slight approximations as compared to the full covariance approaches. Our analysis is supported by comprehensive experiments that pinpoint strengths, weaknesses, and trade-offs of these transfer learning methods. The benchmarks demonstrate the appeal of Sequential Hierarchical GP and Boosted Hierarchical GP as robust and competitive techniques.

Limitations

The low-data regime is challenging for any method. In particular, the empirical success of Gaussian processes hinges on the smoothness assumptions encoded in the kernel. At the same time, it is not obvious how non-smooth functions can be optimized efficiently. Another limitation is that they do not scale to large scale data without approximations, which is not a problem in our setting since we specifically focus on the low-data regime. We are not aware of any negative societal impacts of our work, since we focus on improving the data efficiency of optimization methods.

Acknowledgements

We would like to thank Stefan Falkner for insightful discussions on the results.

References

M. J. Bayarri, J. O. Berger, and F. Liu. Modularization in Bayesian analysis, with emphasis on analysis of computer models. *Bayesian Analysis*, 4(1):119 – 150, 2009. doi: 10.1214/09-BA404.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

Yutian Chen, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 748–756. PMLR, 06–11 Aug 2017.

Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for Bayesian optimization using ranking-weighted Gaussian process ensembles. In *AutoML Workshop at ICML*, volume 7, 2018.

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

Sebastian Flennerhag, Pablo Garcia Moreno, Neil Lawrence, and Andreas Damianou. Transferring knowledge across learning processes. In *International Conference on Learning Representations*, 2019.

Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463 (2088):3251–3269, 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*, pages 1487–1495, 2017.

GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Tinu Theckel Joy, Santu Rana, Sunil Kumar Gupta, and Svetha Venkatesh. Flexible Transfer Learning Framework for Bayesian Optimisation. In James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang, editors, *Advances in Knowledge Discovery and Data Mining*. Springer International Publishing, 2016.

Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

Andreas Krause and Cheng Ong. Contextual Gaussian Process Bandit Optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

Daniel Kühn, Philipp Probst, Janek Thomas, and Bernd Bischl. Automatic Exploration of Machine Learning Experiments on OpenML, 2018a.

Daniel Kühn, Philipp Probst, Janek Thomas, and Bernd Bischl. OpenML R Bot Benchmark Data (final subset), Mar 2018b.

Miguel Lázaro-Gredilla, Joaquin Quinonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

Loic Le Gratiet and Josselin Garnier. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5), 2014.

Tzon-Tzer Lu and Sheng-Hua Shiou. Inverses of 2 × 2 block matrices. *Computers & Mathematics with Applications*, 43(1):119–129, 2002.

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 *Proc. of the International Conference on Robotics and Automation (ICRA)*, pages 1557–1563, 2017.

Jamil Momin and Xin-She Yang. A literature survey of benchmark functions for global optimization problems. *Int. Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

Andrei Paleyes, Mark Pullin, Maren Mahsereci, Neil Lawrence, and Javier González. Emulation of physical processes with emukit. In *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS*, 2019.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cedric Archambeau. Scalable hyperparameter transfer learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018a.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cedric Archambeau. Scalable hyperparameter transfer learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, January 2006.

David Salinas, Huibin Shen, and Valerio Perrone. A quantile-based approach for hyperparameter transfer learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8438–8448. PMLR, 13–18 Jul 2020.

Robert E Schapire. The boosting approach to machine learning: An overview. *Nonlinear estimation and classification*, pages 149–171, 2003.

Alistair Shilton, Sunil Gupta, Santu Rana, and Svetha Venkatesh. Regret Bounds for Transfer Learning in Bayesian Optimisation. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 307–315, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

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

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-Task Bayesian Optimization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Jan N. van Rijn, Bernd Bischl, Luis Torgo, Bo Gao, Venkatesh Umaashankar, Simon Fischer, Patrick Winter, Bernd Wiswedel, Michael R. Berthold, and Joaquin Vanschoren. Openml: A collaborative science platform. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 645–649, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40994-3.

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.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Two-stage transfer surrogate model for automatic hyperparameter optimization. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 199–214, Cham, 2016. Springer International Publishing.

Dani Yogatama and Gideon Mann. Efficient Transfer Learning Method for Automatic Hyperparameter Tuning. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 1077–1085, Reykjavik, Iceland, 22–25 Apr 2014. PMLR.

Yichi Zhang, Daniel W Apley, and Wei Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports*, 10(1):1–13, 2020.

Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266, 2012.

Supplementary Material: Transfer Learning with Gaussian Processes for Bayesian Optimization

Overview

In the appendix we provide the detailed proofs for all claims in the paper, complexity analysis for all discussed methods for multiple sources, ablation studies, and the detailed hyperparameter configurations used in the paper. An overview is shown below.

Table of Contents

A EQUIVALENCES **13**

A.1 Proof of Corollary 1 13

A.2 Proof of Proposition 1 14

A.3 Proof of Proposition 2 14

A.4 Proof of Lemma 1 15

B GENERALIZATION TO MULTIPLE SOURCES **16**

B.1 Multi-Task GP (MTGP) 16

B.2 Multi-Task-Single-k GP (MTKGP) 16

B.3 Weighted Source GP (WSGP) 16

B.4 Hierarchical GP (HGP) 16

B.5 Sequential Hierarchical GP (SHGP) 17

B.6 Mean Hierarchical GP (MHGP) 17

B.7 Boosted Hierarchical GP (BHGP) 18

C DETAILS ON EXPERIMENTS **20**

C.1 Details on the Synthetic Function Families 20

C.2 Details on the Surrogate Meta-Learning Benchmarks 21

D FURTHER EXPERIMENTS **22**

D.1 Results on one-dimensional benchmark functions 22

D.2 Runtime Analysis 22

D.3 Impact of Observational Noise and Propagation of Uncertainty: Ablation Studies 22

E IMPLEMENTATION AND COMPUTATIONAL RESOURCES **30**

A EQUIVALENCES

In the following sections, we provide the proofs of the claims in the main paper. For ease of reference, we restate those claims here. We begin with deriving closed form solutions for the prior and posterior of Bayesian model averaging.

Proposition 1. *Let $f_s \sim \mathcal{GP}(0, k_s)$, $f_t|f_s \sim \mathcal{GP}(f_s, k_t)$, and*

$$p(f_t^{\text{bayes}} \mid \mathcal{D}_t) = \int p(f_t \mid f_s, \mathcal{D}_t) p(f_s \mid \mathcal{D}_t) df_s. \tag{7}$$

Then, the joint model of f_s and f_t^{bayes} is a GP with zero mean prior and the kernel (4).

Corollary 1. *Let $k_\Sigma = k_t + \operatorname{cov}(f_s \mid \mathcal{D}_s)$. Then under the assumptions of Proposition 1 it holds that $f_t^{\text{bayes}} \sim \mathcal{GP}(\mathbb{E}(f_s \mid \mathcal{D}_s), k_\Sigma)$.*

Subsequently, we show that posterior prediction averaging leads to a analytically tractable distribution which equals the Boosted Hierarchical GP (6).

Proposition 2. *Let f_s, f_t be as in Proposition 1 and*

$$p(f_t^{\text{boost}} \mid \mathcal{D}_t) = \int p(f_t \mid f_s, \mathcal{D}_t) p(f_s) df_s.$$

Then, the joint model of f_s and f_t^{boost} is a GP with zero mean and the covariance function defined in (6)*. Further, $f_t^{\text{boost}} \mid \mathbf{x}_*, \mathcal{D}_t$ is multi-variate normal with mean $\mathbb{E}(f_s \mid \mathbf{x}_*, \mathcal{D}_s) - \alpha_{*,t}(\mathbf{y}_t - \mathbb{E}(f_s \mid \mathbf{X}_t, \mathcal{D}_s))$ and covariance matrix $k_t(\mathbf{x}_*, \mathbf{x}_*) - \alpha_{*,t} k_t(\mathbf{X}_t, \mathbf{x}_*) + \Sigma_*^{\text{boost}}$.*

The analysis of both ensemble averaging techniques introduced in Sec. 6 is based on marginalization of the models over the posterior of the source at target data points $\mathbf{X}_t$ and query points $\mathbf{x}_*$. Samples from the source posterior can be obtained from the posterior mean $\mathbb{E}(f_s \mid \mathcal{D}_s)$ and covariance of the source $\operatorname{cov}(f_s \mid \mathcal{D}_s)$ posterior mean and covariance of the source model as follows.

Let L be the Cholesky decomposition of posterior covariance of the source $LL^T = \operatorname{cov}\left(f_s \mid \tilde{\mathbf{X}}_t, \mathcal{D}_s\right)$ evaluated at target data points and query points $\tilde{\mathbf{X}}_t = \mathbf{X}_t \cup \mathbf{x}_*$. Let ε be a standard normal vector of size $N_t + 1$ then

$$f_{s,\varepsilon}(\tilde{\mathbf{X}}_t) = \mathbb{E}\left(f_s \mid \tilde{\mathbf{X}}_t, \mathcal{D}_s\right) + L_s \varepsilon \tag{8}$$

is a sample from the source posterior at target data and query points. The following commonly known result will be useful for deriving the desired Gaussian distributions. For the sake of completeness, we provide a proof of Lemma 1 in Appendix A.4.

Lemma 1. *Let ε be a standard normal random vector of size $N \times 1$. Let $\mu \in \mathbb{R}^N$, $L \in \mathbb{R}^{M \times N}, \Sigma \in \mathbb{R}^{M \times M}$. If $Y \mid \varepsilon \sim \mathcal{N}(\mu + L\varepsilon, \Sigma)$, then $Y \sim \mathcal{N}\left(\mu, \Sigma + LL^T\right)$*

We begin with the proof of Corollary 1, which is a simple application of Lemma 1 to the ensemble of priors used in Bayesian model averaging. We derive Proposition 1 from Corollary 1 by showing that conditioning the joint model from (4) leads to the same posterior. The proof of Proposition 2 requires a more involved application of Lemma 1 to the ensemble of posteriors, that we propose in posterior prediction averaging. A detailed derivation of Lemma 1 can be found in Appendix A.4.

A.1 Proof of Corollary 1

In Bayesian model averaging, we build an ensemble of priors for the target model from multiple functional samples of the source posterior. For a fixed functional sample $f_{s,\varepsilon}(\tilde{\mathbf{X}}_t)$ the prior of the target is a multi-variate normal with mean $\mathbb{E}\left(f_s \mid \tilde{\mathbf{X}}_t, \mathcal{D}_s\right) + L\varepsilon$ and covariance matrix $k_t(\mathbf{X}_t, \mathbf{X}_t)$. Consequently, Corollary 1 follows from Lemma 1 and $LL^T = \operatorname{cov}\left(f_s \mid \tilde{\mathbf{X}}_t, \mathcal{D}_s\right)$.

A.2 Proof of Proposition 1

To show Proposition 1, we start with the model defined in (4) and condition it on the source data. The joint kernel of the Hierarchical GP kernel is given by

$$k((\mathbf{x}, i), (\mathbf{x}', j)) = k_s(\mathbf{x}, \mathbf{x}') + \delta_{i,t}\delta_{j,t}k_t(\mathbf{x}, \mathbf{x}') \tag{9}$$

and after conditioning this model on source data $\mathcal{D}_s$, we obtain the prior of the target model by standard GP inference. Then,

$$k((\tilde{\mathbf{X}}_t, t), (\mathbf{X}_s, s)) = k_s(\tilde{\mathbf{X}}_t, \mathbf{X}_s), \qquad k((\mathbf{X}_s, s), (\tilde{\mathbf{X}}_t, t)) = k_s(\mathbf{X}_s, \tilde{\mathbf{X}}_t), \tag{10}$$

$$k((\tilde{\mathbf{X}}_t, t), (\tilde{\mathbf{X}}_t, t)) = k_s(\tilde{\mathbf{X}}_t, \tilde{\mathbf{X}}_t) + k_t(\tilde{\mathbf{X}}_t, \tilde{\mathbf{X}}_t), \qquad k((\mathbf{X}_s, s), (\mathbf{X}_s, s)) = k_s(\mathbf{X}_s, \mathbf{X}_s). \tag{11}$$

As a consequence, the posterior of the Hierarchical GP model given the source data has mean and covariance

$$\mu^{\text{bayes}} = k_s(\tilde{\mathbf{X}}_t, \mathbf{X}_s)[k_s(\mathbf{X}_s, \mathbf{X}_s)]^{-1}y_s, \tag{12}$$

$$\Sigma^{\text{bayes}} = k_s(\tilde{\mathbf{X}}_t, \tilde{\mathbf{X}}_t) + k_t(\tilde{\mathbf{X}}_t, \tilde{\mathbf{X}}_t) + k_s(\tilde{\mathbf{X}}_t, \mathbf{X}_s)[k_s(\mathbf{X}_s, \mathbf{X}_s)]^{-1}k_s(\mathbf{X}_s, \tilde{\mathbf{X}}_t). \tag{13}$$

We can rewrite

$$\mu^{\text{bayes}} = \mathbb{E}\left(f_s \mid \tilde{\mathbf{X}}_t, \mathcal{D}_s\right) \tag{14}$$

$$\Sigma^{\text{bayes}} = k_t(\tilde{\mathbf{X}}_t, \tilde{\mathbf{X}}_t) + \operatorname{cov}\left(f_s \mid \tilde{\mathbf{X}}_t, \mathcal{D}_s\right). \tag{15}$$

Together with Corollary 1, we obtain that conditioning the Hierarchical GP model defined in (4) on source data leads to the prior target model of Bayesian model averaging. Proposition 1 follows because under the same modelling assumption for the source, the prior is unique.

A.3 Proof of Proposition 2

In this subsection, we compute the distribution obtained from directly averaging over the posteriors on the target data induced by the mean priors sampled from the source posterior (and, thus, ignoring the implicit dependency of the source model on the target data). Denote by $L_{s,*}$ and $L_{s,t}$ the first N_* and remaining N_t rows of L_s. For a fixed sample $f_{s,\varepsilon}(\mathbf{X}_t, \mathbf{x}_*)$, the posterior of the target GP at $\mathbf{x}_*$ is a multivariate normal with mean and covariance matrix

$$\mu^{\text{boost}} = \mathbb{E}\left(f_s \mid \mathbf{x}_*, \mathcal{D}_s\right) + L_{s,*}\varepsilon + \alpha_{*,t}\left[y_t - \mathbb{E}\left(f_s \mid \mathbf{X}_t, \mathcal{D}_s\right) - L_{s,t}\varepsilon\right],$$

$$\Sigma^{\text{boost}} = k_t(\mathbf{x}_*, \mathbf{x}_*) - \alpha_{*,t}k_t(\mathbf{X}_t, \mathbf{x}_*).$$

The posterior mean can be rewritten as

$$\mu^{\text{boost}} = \mathbb{E}\left(f_s \mid \mathbf{x}_*, \mathcal{D}_s\right) + \alpha_{*,t}\left[y_t - \mathbb{E}\left(f_s \mid \mathbf{X}_t, \mathcal{D}_s\right)\right] + L^{\text{boost}}\varepsilon \tag{16}$$

with $L^{\text{boost}} = L_{s,*} + \alpha_{*,t}L_{s,t}$. Recall, that $\Sigma^s_{t,t}$, $\Sigma^s_{t,*}$, $\Sigma^s_{*,t}$, $\Sigma^s_{*,*}$ denote the blocks of the posterior covariance matrix of the source evaluated at target and query points. Then it holds that

$$L_{s,*}L_{s,*}^T = \Sigma^s_{*,*}, \qquad L_{s,t}L_{s,t}^T = \Sigma^s_{t,t}, \tag{17}$$

$$L_{s,*}L_{s,t}^T = \Sigma^s_{*,t}, \qquad L_{s,t}L_{s,*}^T = \Sigma^s_{t,*}. \tag{18}$$

From this, we obtain

$$L^{\text{boost}}(L^{\text{boost}})^T = \Sigma^s_{*,*} + \alpha_{*,t}\Sigma^s_{t,t}\alpha_{*,t}^T - \alpha_{*,t}\Sigma^s_{t,*} - \Sigma^s_{*,t}\alpha_{*,t}^T = \Sigma^{\text{boost}}_*. \tag{19}$$

With this observation, the posterior distribution stated in Proposition 2 can be derived from Lemma 1.

Finally, it remains to show that the joint model defined in (6) leads to the same posterior distribution. Since the source kernel is only evaluated at source points, conditioning on the source first yields

$$\begin{pmatrix} f_t^{\text{boost}}(\mathbf{x}_*) \\ \mathbf{y}_t \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbb{E}\left(f_s \mid \mathbf{x}_*, \mathcal{D}_s\right) \\ \mathbb{E}\left(f_s \mid \mathbf{X}_t, \mathcal{D}_s\right) \end{pmatrix}, \begin{pmatrix} k_t(\mathbf{x}_*, \mathbf{x}_*) + \Sigma^{\text{boost}}_* & k_t(\mathbf{x}_*, \mathbf{X}_t) \\ k_t(\mathbf{X}_t, \mathbf{x}_*) & k_t(\mathbf{X}_t, \mathbf{X}_t) + \sigma_t^2\mathbf{I} \end{pmatrix}\right). \tag{20}$$

Note that the additive term Σ^{boost}_* only contributes to the covariance at query points. That is, it will only contribute an additive term to the posterior covariance. Using standard formulas for conditional distributions in multi-variate Gaussians, we obtain that $f_t^{\text{boost}}(\mathbf{x}_*)|\mathbf{y}_t$ is multi-variate normally distributed with mean $\mathbb{E}\left(f_s \mid \mathbf{x}_*, \mathcal{D}_s\right) - \alpha_{*,t}(\mathbf{y}_t - \mathbb{E}\left(f_s \mid \mathbf{X}_t, \mathcal{D}_s\right))$ and covariance matrix $k_t(\mathbf{x}_*, \mathbf{x}_*) - \alpha_{*,t}k_t(\mathbf{X}_t, \mathbf{x}_*) + \Sigma^{\text{boost}}_*$.

A.4 Proof of Lemma 1

Let $p\left(x; \mu + L\varepsilon, \Sigma\right)$ denote the density of a multi-variate normal distribution with mean $\mu + L\varepsilon$ and covariance matrix Σ. Further, denote by $|\Sigma|$ the determinant of Σ. Then,

$$\begin{aligned} p\left(x; \mu + L\varepsilon, \Sigma\right) &= (2\pi)^{-N/2} \left|\Sigma\right|^{-1/2} \exp\left[-\frac{1}{2}\left[x - \mu - L\varepsilon\right]^T \Sigma^{-1} \left[x - \mu + L\varepsilon\right]\right] && (21) \\ &= \left|L^T L\right|^{-1/2} p\left(\varepsilon; L^{-1}(x-\mu), L^{-1}\Sigma\left(L^{-1}\right)^T\right) && (22) \end{aligned}$$

Similarly as above, denote by $p\left(x; 0, \mathbb{1}\right)$ denote the density of a spherical multi-variate Gaussian. Then from (22), we obtain

$$\begin{aligned} &p\left(x; \mu + L\varepsilon, \Sigma\right) p\left(\varepsilon; 0, \mathbf{1}\right) \\ &\quad = p\left(x; \mu, \Sigma + LL^T\right) p\left(\varepsilon; L^{-1}(L^T + \Sigma^{-1})(x-\mu), \mathbb{1} + L^{-1}\Sigma\left(L^{-1}\right)^T\right). && (23) \end{aligned}$$

From this the result follows by integrating out ε

$$\int p\left(x; \mu + L\varepsilon, \Sigma\right) p\left(\varepsilon; 0, \mathbf{1}\right) d\varepsilon = p\left(x; \mu, \Sigma + LL^T\right). \qquad (24)$$

B GENERALIZATION TO MULTIPLE SOURCES

In this appendix, we discuss the generalization of Sec. 4 to multiple sources. We discuss the computational complexity to predict mean and covariance of the target model at a new query point x_*. We collect all calculations that can be done without knowledge of the query point as "training complexity" and the remainder as "prediction complexity". In this section we assume $n_s > 1$ sources. For easier generalization we denote the sources by $\nu = 1, 2, ..., n_s$ and the target by $\nu = n_s + 1$. Where appropriate, the analysis is structured in four parts: Training of the first source, training of the $n > 1$th source assuming all "lower" sources $n' < n$ have already been trained, training of the target assuming all sources have been trained, and prediction from the target model. Our basis operation for evaluating the computational complexity is scalar multiplication. Sometimes, we will also present our results using $N_t = N_{n_s+1}$ and $N_s = \sum_{\nu=1}^{n_s} N_\nu$.

B.1 Multi-Task GP (MTGP)

The joint kernel for n_s sources and the target in Multi-Task GP between two points $(\mathbf{x}, i), (\mathbf{x}', j)$ from tasks $i, j \in \{1, 2, ..., n_s + 1\}$ is a sum of separable kernels given by

$$k((\mathbf{x}, i), (\mathbf{x}', j)) = \sum_{\nu=1}^{n_s+1} [\mathbf{W}_\nu]_{i,j}\, k_\nu(\mathbf{x}, \mathbf{x}') + \delta_{\mathbf{x}\mathbf{x}'}\delta_{ij}\sigma_i^2. \tag{25}$$

The number of hyperparameters is $\mathcal{O}(n_s^3)$. Training and prediction complexity are $\mathcal{O}((N_t + N_s)^3)$ and $\mathcal{O}((N_t + N_s)^2)$, as for a simple GP with $N_t + N_s$ data points.

B.2 Multi-Task-Single-k GP (MTKGP)

The only difference to Multi-Task GP is that $k_\nu = k_s$, which reduces the number of hyperparameters to $\mathcal{O}(n_s^2)$.

B.3 Weighted Source GP (WSGP)

The joint kernel for n_s sources and the target in Weighted Source GP is given by

$$[\mathbf{W}_\nu]_{i,j} = \delta_{i,\nu}\delta_{j,\nu} + w_\nu\,(\delta_{i,\nu} + \delta_{i,n_s+1})\,(\delta_{j,\nu} + \delta_{j,n_s+1}) \text{ for } \nu = \{1, 2, ..., n_s\}, \tag{26}$$

$$[\mathbf{W}_{n_s+1}]_{i,j} = \delta_{i,n_s+1}\delta_{j,n_s+1}. \tag{27}$$

Since all the blocks between different sources are zero, inverting the kernel matrix K is computationally simpler than for the more general kernels. We define the source source-block (which is diagonal in i, j) $A = [K]_{i\leq n_s, j\leq n_s}$, the target block $D = [K]_{n_s+1,n_s+1}$, and the connecting blocks $B = [K]_{i\leq n_s, j=n_s+1}$ and $C = [K]_{i=n_s+1, j\leq n_s}$. With this definition, we see that using block-inversion (Lu and Shiou, 2002).

$$K^{-1} = \begin{pmatrix} A & B \\ B^T & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} + A^{-1}B(D - B^TA^{-1}B)^{-1}B^TA^{-1} & -A^{-1}B(D - B^TA^{-1}B)^{-1} \\ -(D - B^TA^{-1}B)^{-1}B^TA^{-1} & (D - B^TA^{-1}B)^{-1} \end{pmatrix} \tag{28}$$

we can use the block-diagonal structure of A for an efficient inversion of the kernel matrix. The total complexity is $\mathcal{O}(N_t^3 + N_t^2 N_s + N_t N_s^2 + \sum_{\nu=1}^{n_s} N_\nu^3)$. Note how due to the blockdiagonal structure of A, the term $(\sum_{\nu=1}^{n_s} N_\nu)^3$ for the inversion of a matrix of the size of A collapses to the much more favourable $\sum_{\nu=1}^{n_s} N_\nu^3$. Prediction at a query point is of order $\mathcal{O}((N_t + N_s)^2)$.

B.4 Hierarchical GP (HGP)

The joint kernel for n_s sources and the target in HGP is defined by (25) with

$$[\mathbf{W}_\nu]_{i\geq\nu, j\geq\nu} = 1 \text{ and } 0 \text{ otherwise }. \tag{29}$$

The number of hyperparameters is hence the number of hyperparameters in the kernel functions, which is of order $\mathcal{O}(n_s)$. When optimizing the hyperparameters of all sources and the target jointly, the training and prediction complexity are $\mathcal{O}((N_t + N_s)^3)$ and $\mathcal{O}((N_t + N_s)^2)$, as for a simple GP with $N_t + N_s$ data points.

B.5 Sequential Hierarchical GP (SHGP)

SHGP carries out a sequential training, one source at at time. In the following we analyse the complexity of these sequential steps individually.

Training of the first source The first source is a simple GP, with training complexity $\mathcal{O}(N_1^3)$ and prediction complexity $\mathcal{O}(N_1^2)$.

Training of the nth source In order to train the nth source, we need to evaluate the posterior mean and covariance of the $n-1$th source at the data points X_n. This requires evaluating the posterior means of all underlying sources at the same points, resulting in $\mathcal{O}(N_n \sum_{\nu=1}^{n-1} N_\nu)$ multiplications.

In addition we need to invert the covariance matrix $\left[\Sigma_n(X_n, X_n) + \sigma_n^2 \mathbf{I}\right]$ which is of order $\mathcal{O}(N_n^3)$. In order to obtain the covariance matrix, we need to evaluate the posterior covariance of the $n-1$th source X_n

$$\begin{aligned}\Sigma_n(X_n, X_n) &= k_n(X_n, X_n) + \Sigma_{n-1}^{\text{post}}(X_n, X_n),\\ \Sigma_{n-1}^{\text{post}}(X_n, X_n) &= \Sigma_{n-1}(X_n, X_n)\\ &\quad - \Sigma_{n-1}(X_n, X_{n-1}) \left[\Sigma_{n-1}(X_{n-1}, X_{n-1}) + \sigma_{n-1}^2 \mathbf{I}\right]^{-1} \Sigma_{n-1}(X_{n-1}, X_n)\end{aligned} \tag{30}$$

Inverting $\left[\Sigma_{n-1}(X_{n-1}, X_{n-1}) + \sigma_{n-1}^2 \mathbf{I}\right]$ is part of the training of the $n-1$ source.

From (30) we can see that we need to evaluate $\Sigma_{n-i}^{\text{post}}(X_n, X_n) \forall i \in \{1, 2, ..., n-1\}$. This requires $\mathcal{O}(\sum_{i=1}^{n-1}(N_n^2 N_{n-i} + N_n N_{n-i}^2))$ multiplications and the calculation of $\Sigma_{n-i-1}^{\text{post}}(X_n, X_{n-i}) \forall i \in \{1, 2, ..., n-2\}$. The later has complexity $\mathcal{O}(\sum_{i=1}^{n-2}(N_n N_{n-i} N_{n-i-1} + \min(N_n, N_{n-i}) N_{n-i-1}^2))$ plus the cost of calculating $\Sigma_{n-i-j}^{\text{post}}(X_n, X_{n-i}) \forall j \in \{2, 3, ..., n-i-1\}$. Hence, we actually need the complexity of $\Sigma_{n-i-j}^{\text{post}}(X_n, X_{n-i}) \forall j \in \{1, 2, ..., n-i-1\} \forall i \in \{1, 2, ..., n-2\}$ which is given by $\mathcal{O}(\sum_{i=1}^{n-2} \sum_{j=1}^{n-i-1}(N_n N_{n-i} N_{n-i-j} + \min(N_n, N_{n-i}) N_{n-i-j}^2))$. All together we have a training complexity of

$$\begin{aligned}&\mathcal{O}\left(N_n^3 + \sum_{i=1}^{n-1}(N_n^2 N_{n-i} + N_n N_{n-i}^2) + \sum_{i=1}^{n-2} \sum_{j=1}^{n-i-1} (N_n N_{n-i} N_{n-i-j} + \min(N_n, N_{n-i}) N_{n-i-j}^2)\right)\\ &= \mathcal{O}\left(N_n^3 + N_n^2 \sum_{i=1}^{n-1} N_i + N_n \left(\sum_{i=1}^{n-1} N_i\right)^2\right).\end{aligned} \tag{31}$$

Training of the target Training of the target is analogous to training the nth source. Setting $n = n_s + 1$, we have complexity $\mathcal{O}(N_t^3 + N_t^2 N_s + N_t N_s^2)$.

Prediction from the target The prediction at a new query point x_* is dominated by evaluating the covariance matrix. We can read-off this complexity from (31) by ignoring the N_n^3 term and setting $n = n_s + 2$ and $N_{n_s+2} = 1$, giving $\mathcal{O}((N_t + N_s)^2)$.

B.6 Mean Hierarchical GP (MHGP)

The joint kernel for n_s sources and the target in Mean Hierarchical GP is defined by (25) with

$$[\mathbf{W}_\nu]_{i,j} = \delta_{i,\nu} \delta_{j,\nu}. \tag{32}$$

The number of hyperparameters is hence the number of hyperparameters in the kernel functions, which is of order $\mathcal{O}(n_s)$.

Training of the first source The first source is a simple GP, with training complexity $\mathcal{O}(N_1^3)$ and prediction complexity $\mathcal{O}(N_1^2)$.

Training of the nth source In order to train the nth source, we need to evaluate the posterior mean of the $n-1$th source at the data points of the nth source. This in turn requires the evaluating the posterior means of all underlying sources at the same points, all together $\mathcal{O}(N_n \sum_{\nu=1}^{n-1} N_\nu)$ multiplications. Except for this, the complexity is the same as in a simple GP with N_n data points, hence all together we get a training complexity of $\mathcal{O}(N_n^3 + N_n \sum_{\nu=1}^{n-1} N_\nu)$.

Training of the target Training of the target is analogous to training the nth source. Setting $n = n_s + 1$, we have complexity $\mathcal{O}(N_t^3 + N_t N_s)$.

Prediction from the target In order to predict mean of the target model at a new query point x_*, we need to evaluate the posterior means of all sources at x_*, requiring $\mathcal{O}(N_s)$ multiplications. For predicting the covariance, the complexity is the same as for a simple GP $\mathcal{O}(N_t^2)$. The overall prediction complexity is hence $\mathcal{O}(N_t^2 + N_s)$.

B.7 Boosted Hierarchical GP (BHGP)

To the best of our knowledge in Boosted Hierarchical GP there is no unique generalization to $n_s > 1$ sources, because of the heuristic aggregation of uncertainty. Below we outline several avenues that one could consider, which differ in terms of complexity. The most straightforward approach is to do everything as in Mean Hierarchical GP, and add a boosting term to the posterior prediction of the target task. The kernel for that is simply given by

$$k((\mathbf{x}, i), (\mathbf{x}', j)) = \sum_{\nu=1}^{n_s+2} [\mathbf{W}_\nu]_{i,j} k_\nu(x, x') + \delta_{xx'} \delta_{ij} \sigma_i^2 \qquad \forall i, j \in \{1, 2, ..., n_s + 2\}. \tag{33}$$

In this single layer boosting, we have $[\mathbf{W}_\nu]_{ij} = \delta_{i\nu}\delta_{j\nu}$, $k_{n_s+2} = k_{n_s+1} + \Sigma_{n_s+1}^{\text{boost}}$, and $\sigma_{n_s+2} = 0$. Here $\nu, i, j = n_s + 2$ describes the query points and

$$\begin{aligned}\Sigma_{n_s+1}^{\text{boost}}(\mathbf{x}_*, \mathbf{x}_*) = \Sigma_{n_s}^{\text{post}}(\mathbf{x}_*, \mathbf{x}_*) + \alpha_{n_s} \Sigma_{n_s}^{\text{post}}(\mathbf{X}_{n_s+1}, \mathbf{X}_{n_s+1}) \alpha_{n_s}^T \\ - \alpha_{n_s} \Sigma_{n_s}^{\text{post}}(\mathbf{X}_{n_s+1}, \mathbf{x}_*) - \Sigma_{n_s}^{\text{post}}(\mathbf{x}_*, \mathbf{X}_{n_s+1}) \alpha_{n_s}^T, \end{aligned} \tag{34}$$

$$\alpha_{n_s} = k_{n_s+1}(\mathbf{x}_*, \mathbf{X}_{n_s+1}) \left(k_{n_s+1}(\mathbf{X}_{n_s+1}, \mathbf{X}_{n_s+1}) + \sigma_{n_s+1}^2 \mathbb{1}\right)^{-1}. \tag{35}$$

In that case, the boosting is only done in the last layer. The complexity of Boosted Hierarchical GP has two contributions. One, the complexity of Mean Hierarchical GP and the complexity of calculating the boosting term. For the boosting term, we need to calculate $\Sigma_{n_s}^{\text{post}}(\mathbf{X}_{n_s+1}, \mathbf{X}_{n_s+1})$ which is of order $\mathcal{O}\left(N_{n_s+1}^2 N_{n_s} + N_{n_s+1} N_{n_s}^2\right)$, this can be done during training, since it does not depend on the query point. During prediction we need to evaluate α_{n_s} which is of order $\mathcal{O}\left(N_{n_s+1}^2\right)$ because the matrix inversion can be done as part of the training. We also need to to evaluate $\Sigma_{n_s}^{\text{post}}(\mathbf{x}_*, \mathbf{x}_*)$ and $\Sigma_{n_s}^{\text{post}}(\mathbf{x}_*, \mathbf{X}_{n_s+1})$ which is of order $\mathcal{O}\left(N_{n_s+1} N_{n_s} + N_{n_s}^2\right)$. To conclude the calculation of the boosting term, we need to evaluate (34) which is of order $\mathcal{O}\left(N_{n_s+1}^2\right)$. All together we have training complexity $\mathcal{O}\left(\sum_{\nu=1}^{n_s+1} N_\nu^3 + N_{n_s+1}^2 N_{n_s} + N_{n_s+1} N_{n_s}^2\right)$ and prediction complexity $\mathcal{O}\left(N_{n_s+1}^2 + N_{n_s+1} N_{n_s} + N_{n_s}^2\right) + \sum_{\nu=1}^{n_s} N_\nu)$.

An alternative approach which we follow in our experiments is to add such boosting terms at each intermediate prediction in the hierarchy by substituting $\Sigma_{n_s}^{\text{post}} \to \Sigma_{n_s}^{\text{post}} + \Sigma_{n_s}^{\text{boost}}$ recursively in (34). Note that $\Sigma_1^{\text{post}} = 0$. Note that the likelihood of the boosted model is the same as of Mean Hierarchical GP. Thus, this change only affects the prediction. However, before being able to make predictions, we now have to pre-calculate additional terms. In particular, this requires to recursively evaluate the posterior covariances, very similarly as in (30). This leads to an additional training complexity of $\mathcal{O}(N_t^2 N_s + N_t N_s^2)$. During prediction, the additional cost comes from the boost term which involves recursive evaluation of the posterior covariance terms containing the query point. This increases the total prediction cost to $\mathcal{O}((N_t + N_s)^2)$.

Table 2: Computational complexity of the algorithms. Meta training is performed once before starting the optimization. Training the target model is carried out once at every optimization step. Prediction occurs multiple times during each BO step when optimizing the acquisition function. The kernel-based techniques model the data from all tasks jointly and do therefore not have a meta-training stage.

Models	Stage	Complexity
MTGP	meta training	N/A
	target training	$\mathcal{O}\left[(N_t + N_s)^3\right]$
	prediction	$\mathcal{O}\left[(N_t + N_s)^2\right]$
MTKGP	meta training	N/A
	target training	$\mathcal{O}\left[(N_t + N_s)^3\right]$
	prediction	$\mathcal{O}\left[(N_t + N_s)^2\right]$
WSGP	meta training	N/A
	target training	$\mathcal{O}\left[N_t^3 + N_t^2 N_s + N_t N_s^2 + \sum_{\nu=1}^{n_s} N_\nu^3\right]$
	prediction	$\mathcal{O}\left[(N_t + N_s)^2\right]$
HGP	meta training	N/A
	target training	$\mathcal{O}\left[(N_t + N_s)^3\right]$
	prediction	$\mathcal{O}\left[(N_t + N_s)^2\right]$
SHGP	meta training	$\mathcal{O}\left[N_s^3\right]$
	target training	$\mathcal{O}\left[N_t^3 + N_t^2 N_s + N_t N_s^2\right]$
	prediction	$\mathcal{O}\left[(N_t + N_s)^2\right]$
MHGP	meta training	$\mathcal{O}\left[\sum_{\nu=1}^{n_s} N_\nu^3 + N_s^2\right]$
	target training	$\mathcal{O}\left[N_t^3 + N_t N_s\right]$
	prediction	$\mathcal{O}\left[N_t^2 + N_s\right]$
BHGP	meta training	$\mathcal{O}\left[N_s^3\right]$
	target training	$\mathcal{O}\left[N_t^3 + N_t^2 N_s + N_t N_s^2\right]$
	prediction	$\mathcal{O}\left[(N_t + N_s)^2\right]$

C DETAILS ON EXPERIMENTS

All the experiments use default BO parameters with Upper Confidence Bound (UCB) and exploration coefficient $\beta = 3$ as acquisition function. The source and target functions are sampled randomly from the function family, and the source data are sampled randomly from each source task, see Appendix C.1 for details. BO is used to optimize the target function from scratch, i.e., without initial samples. A certain amount of i.i.d. zero-mean Gaussian observational noise is added during the data generation process with a standard deviation specified in each figure caption.

All GP models are based on the squared-exponential kernel with automatic relevance determination. The GP hyperparameters are optimized by maximizing the likelihood function for the observed data using the L-BFGS-B optimizer with 10 initial guesses, $\mathbf{x}_0$, where each instance is sampled from

$$x_0 = \log\left(1 + \exp\left(x_0'\right)\right), \qquad x_0' \sim \mathcal{N}(0, 1). \tag{36}$$

This generic optimization strategy is possible since we normalize the observed data to zero-mean, unit-variance. All other details regarding training and prediction are kept fixed to GPy's defaults (GPy, since 2012). The only exception is the OpenML SVM benchmark, where, for complexity reasons, we only use 3 initial guesses and a maximum of 20 gradient-ascent steps for the optimization of the likelihood function of HGP and WSGP.

C.1 Details on the Synthetic Function Families

The Forrester Family The Forrester function is a one-dimensional synthetic function with one global minimum, one local minimum, and a zero-gradient inflection point. The function is given by

$$f(x; a, b, c) = a\left(6x - 2\right)^2 \sin\left(12x - 4\right) + b\left(x - \frac{1}{2}\right) - c, \quad x \in [0, 1]. \tag{37}$$

The original function is given by $a = 1, b = 0, c = 0$. In this paper, a family of functions is formed by choosing the following probability distributions for the parameters (a, b, c):

$$a \sim \mathcal{U}(0.2, 3), \quad b \sim \mathcal{U}(-5, 15), \quad c \sim \mathcal{U}(-5, 5), \tag{38}$$

where $\mathcal{U}(\cdot, \cdot)$ denotes the uniform distribution. The Forrester family is therefore defined over a three-dimensional uniform distribution.

For generating the data of n_s source data sets, we draw n_s random tasks using (38), and sample a given number of points per task using a random distribution. In Fig. 2, 20 points per task are sampled.

The Alpine Family The Alpine function is a one-dimensional synthetic function with one global minimum and three local minima. The function is given by

$$f(x; s) = x\sin(x + \pi + s) + 0.1x, \quad x \in [-10, 10]. \tag{39}$$

The original function is given by $s = 0$. In this paper, the Alpine function is used for the multi-source experiments in Fig. 4. There, $s = 0$ is used for the target function, while the five source functions are given by $s = k\pi/12, k = 1, ..., 5$. Twenty random points are sampled for each source function. Note that this benchmark is identical to the one presented in (Feurer et al., 2018, Sec. 5.1).

In the single-source experiments presented in Fig. 2, the target function is also set to $s = 0$, while the source function is sampled randomly from the five choices above.

The Branin Family The Branin function is a two-dimensional synthetic function defined as

$$f(x_1, x_2; a, b, c, r, s, t) = a(x_2 - bx_1^2 + cx_1 - r) + s(1 - t)\cos(x_1) + s, \quad x_1 \in [-5, 10], x_2 \in [0, 15] \tag{40}$$

A family of functions is formed by choosing the following probability distributions for the parameters (a, b, c, r, s, t):

$$a \sim \mathcal{U}(0.5, 1.5), b \sim \mathcal{U}(0.1, 0.15), c \sim \mathcal{U}(1, 2), r \sim \mathcal{U}(5, 7), s \sim \mathcal{U}(8, 12), t \sim \mathcal{U}(0.03, 0.05). \tag{41}$$

The Branin family is therefore defined over a six-dimensional uniform distribution. For generating the data of n_s source data sets, we draw n_s random tasks using (41), and sample a given number of points per task using a random distribution. In Fig. 2, 40 points per task are sampled.

The Hartmann3 Family The Hartmann3 function is a sum of four three-dimensional Gaussian distributions and is defined by

$$f(\mathbf{x};\boldsymbol{\alpha}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{i,j}\left(x_j - P_{i,j}\right)^2\right), \quad \mathbf{x} \in [0,1]^3, \tag{42}$$

with

$$\mathbf{A} = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad \mathbf{P} = 10^{-4} \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}.$$

The original Hartmann3 function is given by $\boldsymbol{\alpha} = (1.0, 1.2, 3.0, 3.2)^T$. In this paper, a family of functions is formed by choosing the following probability distributions for the parameters $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$:

$$\alpha_1 \sim \mathcal{U}(1.00, 1.02), \alpha_2 \sim \mathcal{U}(1.18, 1.20), \alpha_3 \sim \mathcal{U}(2.8, 3.0), \alpha_4 \sim \mathcal{U}(3.2, 3.4). \tag{43}$$

The Hartmann3 family is therefore defined over a four-dimensional uniform distribution. For generating the data of n_s source data sets, we draw n_s random tasks using (43), and sample a given number of points per task using a random distribution. In Fig. 2, 60 points per task are sampled.

The Hartmann6 Family The Hartmann6 family is a sum of four six-dimensional Gaussian distributions and is defined by

$$f(\mathbf{x};\boldsymbol{\alpha}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{i,j}\left(x_j - P_{i,j}\right)^2\right), \quad \mathbf{x} \in [0,1]^6, \tag{44}$$

with

$$\mathbf{A} = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix},$$

$$\mathbf{P} = 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}.$$

The original Hartmann6 function is given by $\boldsymbol{\alpha} = (1.0, 1.2, 3.0, 3.2)^T$. In this paper, a family of functions is formed by choosing the following probability distributions for the parameters $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$:

$$\alpha_1 \sim \mathcal{U}(1.00, 1.02), \alpha_2 \sim \mathcal{U}(1.18, 1.20), \alpha_3 \sim \mathcal{U}(2.8, 3.0), \alpha_4 \sim \mathcal{U}(3.2, 3.4). \tag{45}$$

The Hartmann6 family is therefore defined over a four-dimensional uniform distribution. For generating the data of n_s source data sets, we draw n_s random tasks using (45), and sample a given number of points per task using a random distribution. In Fig. 2 and Fig. 4, 120 and 60 points per task are sampled, respectively.

C.2 Details on the Surrogate Meta-Learning Benchmarks

The SVM benchmark tunes four hyperparameters: cost (float), gamma (float), degree (integer), and kernel type (discrete). The same data set was used in Perrone et al. (2018b) for transfer-learning benchmarks. Further details are provided in Table 3 and in Kühn et al. (2018b).

Scenario Name	OpenML flow id	OpenML data set ids	$\min N_t$	$\max N_t$	$\sum_t N_t$
OpenML_SVM	5891	10101, 145878, 146064, 14951, 34537, 3485, 3492, 3493, 3494, 37, 3889, 3891, 3899, 3902, 3903, 3913, 3918, 3950, 9889, 9914, 9946, 9952, 9967, 9971, 9976, 9978, 9980, 9983	987	3217	50217

Table 3: OpenML Random Bot surrogate benchmark test data set characteristics, selected from the full repertoire of 37 data sets, cf. Kühn et al. (2018b, Table 2). $\min N_t$ ($\max N_t$) denotes the minimum (maximum) number of data points in a task, while $\sum_t N_t$ is the total number of points in the data set.

D FURTHER EXPERIMENTS

D.1 Results on one-dimensional benchmark functions

Results on the single-source one-dimensional Forrester and Alpine function families are presented in Fig. 5. The function families are described in Appendix C.1. These results are consistent with the discussion and analysis from Sec. 7.1.

D.2 Runtime Analysis

Here we present runtime studies for the entire optimization procedure, which includes (i) training the models including optimizing their hyperparameters, and (ii) optimization of the acquisition function. We time the runs by optimizing the Hartmann6 benchmark function. We consider one single source task with the same configuration as in Fig. 2. Results are shown in Fig. 6 and are consistent with training runtimes reported in Fig. 3. This reinforces that model training is more expensive than acquisition-function optimization since the former has steeper complexity than the latter for the Bayesian models, see Tables 1 and 2.

D.3 Impact of Observational Noise and Propagation of Uncertainty: Ablation Studies

We perform a systematic experimental study on the impact of observational noise and amount of source data on the algorithm performance. The three regimes discussed in Sec. 7.1 can be clearly observed for the Hartmann3 function family in Fig. 3.

Here, we showcase the dynamics of these three regimes for the challenging Hartmann6 function family in Figs. 7–11. The first regime, in which the amount of source data is insufficient to build a model that faithfully describes the global shape of the source function, is presented in Fig. 7, where only 30 data points are considered. Here, algorithms that model the data jointly (HGP, WSGP) have a clear advantage. A crossover to the intermediate regime, in which the source data describe the source function much better but with significant uncertainty, happens gradually in Figs. 8–10, where the source data are increased from 60 to 250 points. Here, our relatively lightweight SHGP is competitive and on par in terms of performance with the more general HGP. The third regime is reached in Fig. 11, where the performance of all methods except GPBO and RGPE converge to similar performance.

The observational noise adds a further level of complexity to these dynamics. In essence, the observational noise controls the boundary between the three regimes: the more noise is present, the more data points are required to describe the function for a fixed model quality. In other words, observational noise leads to enhanced model uncertainty, whose propagation is key to obtaining a high-quality model. It is therefore expected that (i) the performance gap between Bayesian and non-Bayesian techniques becomes ever bigger for increased observational noise, and (ii) our developed techniques, SHGP and BHGP, particularly excel for elevated observational noise

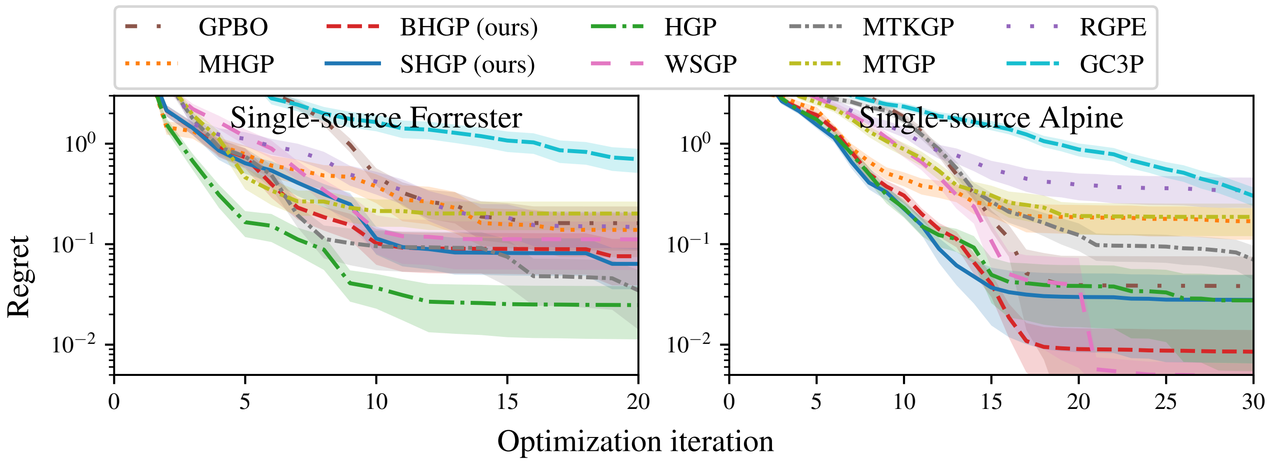

Figure 5: Performance on the single-source one-dimensional benchmarks: the Forrester (left) and Alpine (right) function families. The mean and standard error of the mean of the regret function are plotted. Due to enhanced stochasticity, the statistics are computed over 200 independent runs. The source data are sampled randomly from the source function and contain 20 points. I.i.d. observational noise of standard deviation $\sigma_s = \sigma_t = 0.1$ is added during data generation.

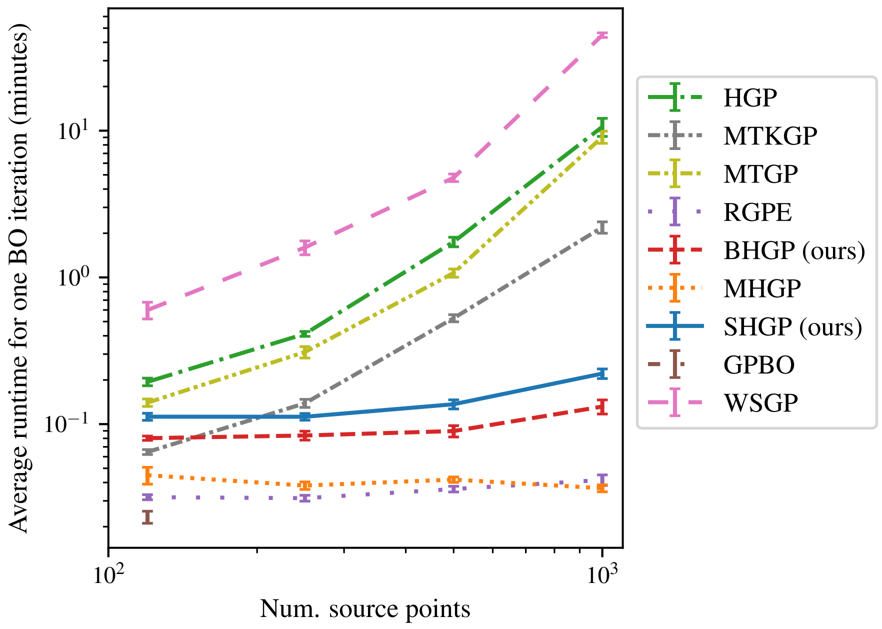

Figure 6: Average runtime per BO step versus the number of points in the source data set when optimizing on the Hartmann6 benchmark. The average is taken over the first twenty-five BO steps. The source data are sampled randomly from the source function. I.i.d. observational noise of standard deviation $\sigma_s = \sigma_t = 0.1$ is added during data generation. The mean and standard error of the mean are computed over 7 independent runs. The performance of GPBO is independent of the number of historical points.

compared to algorithms that do not propagate uncertainty like MHGP and RGPE. These aspects can be observed in Figs. 8 and 9.

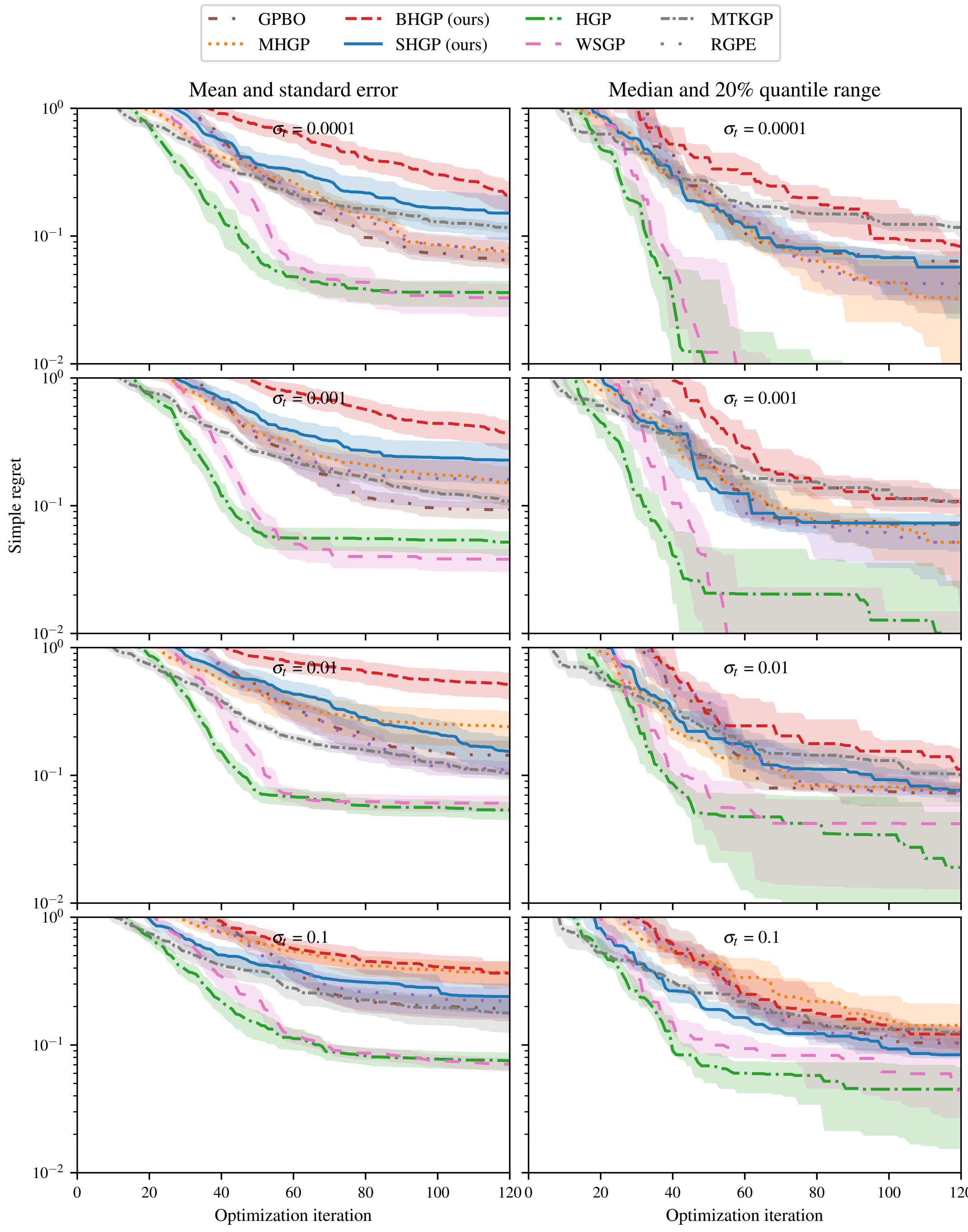

Figure 7: Performance on the single-source Hartmann6 function family for 30 points in the source data set. We vary the amount of observational noise, $\sigma_s = \sigma_t$, from 0.001 to 0.1. The left panel depicts the mean ± standard error of the mean. The right panel shows the median and 20% quantile range.

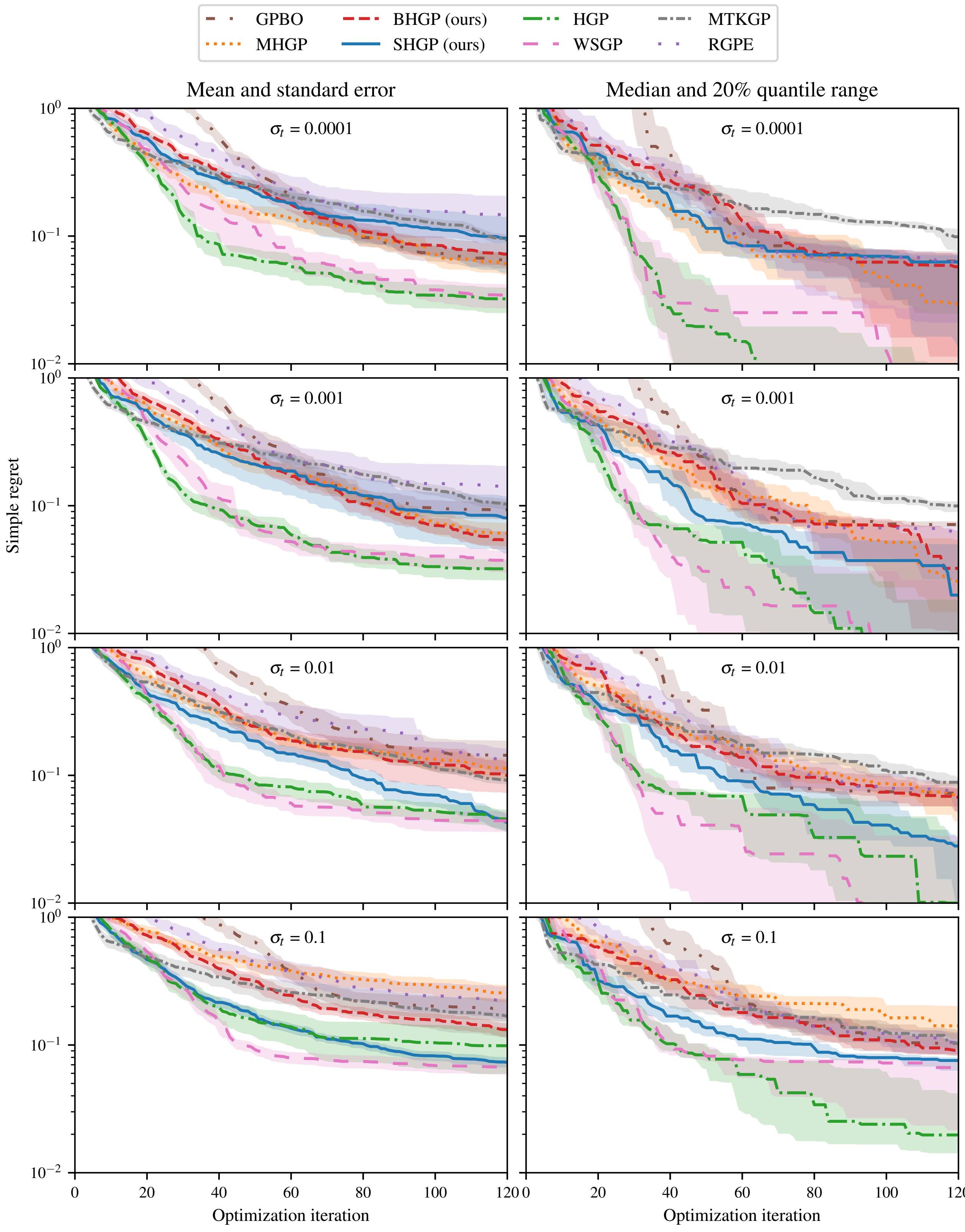

Figure 8: Performance on the single-source Hartmann6 function family for 60 points in the source data set. We vary the amount of observational noise, $\sigma_s = \sigma_t$, from 0.001 to 0.1. The left panel depicts the mean ± standard error of the mean. The right panel shows the median and 20% quantile range.

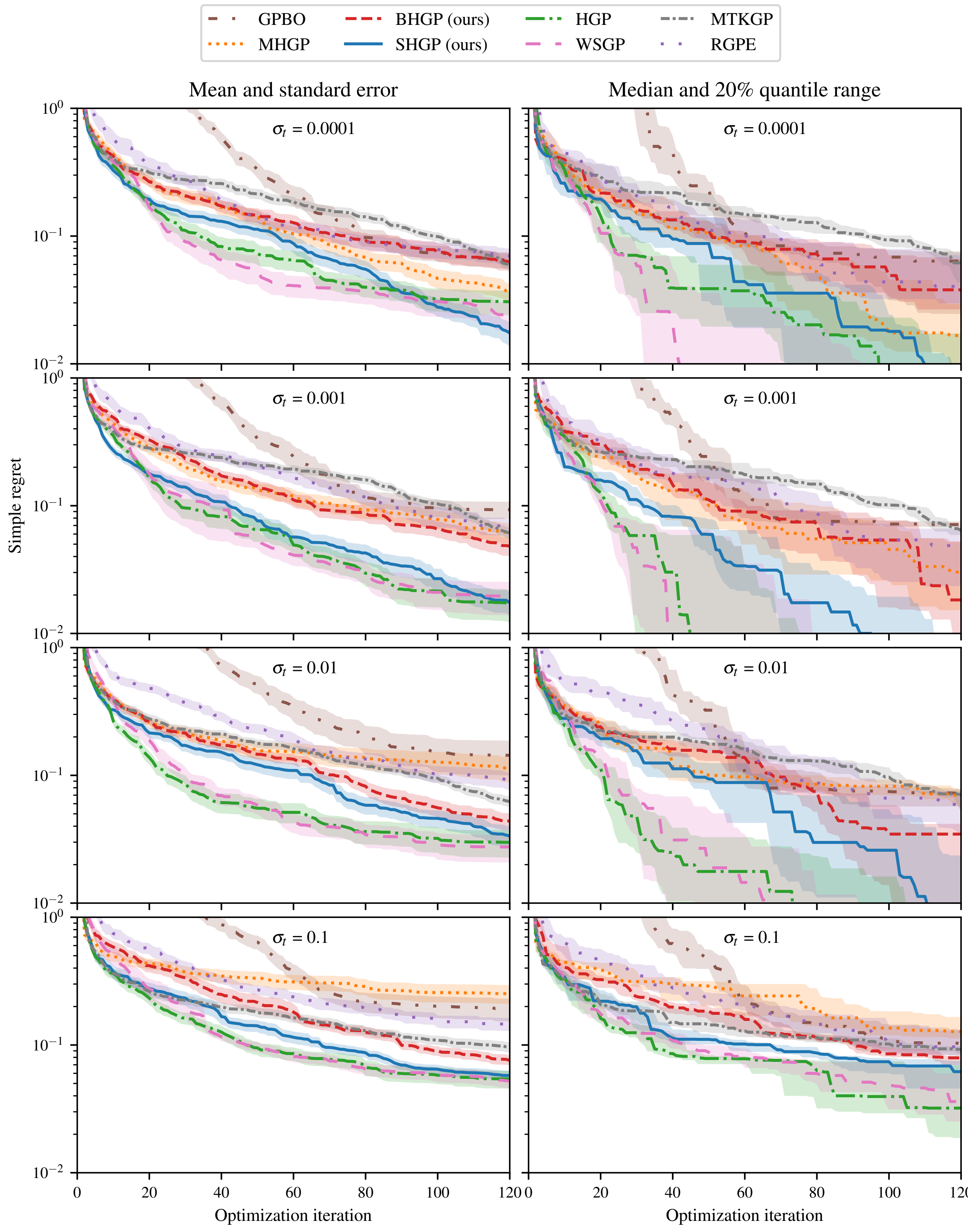

Figure 9: Performance on the single-source Hartmann6 function family for 120 points in the source data set. We vary the amount of observational noise, $\sigma_s = \sigma_t$, from 0.001 to 0.1. The left panel depicts the mean $\pm$ standard error of the mean. The right panel shows the median and 20% quantile range.

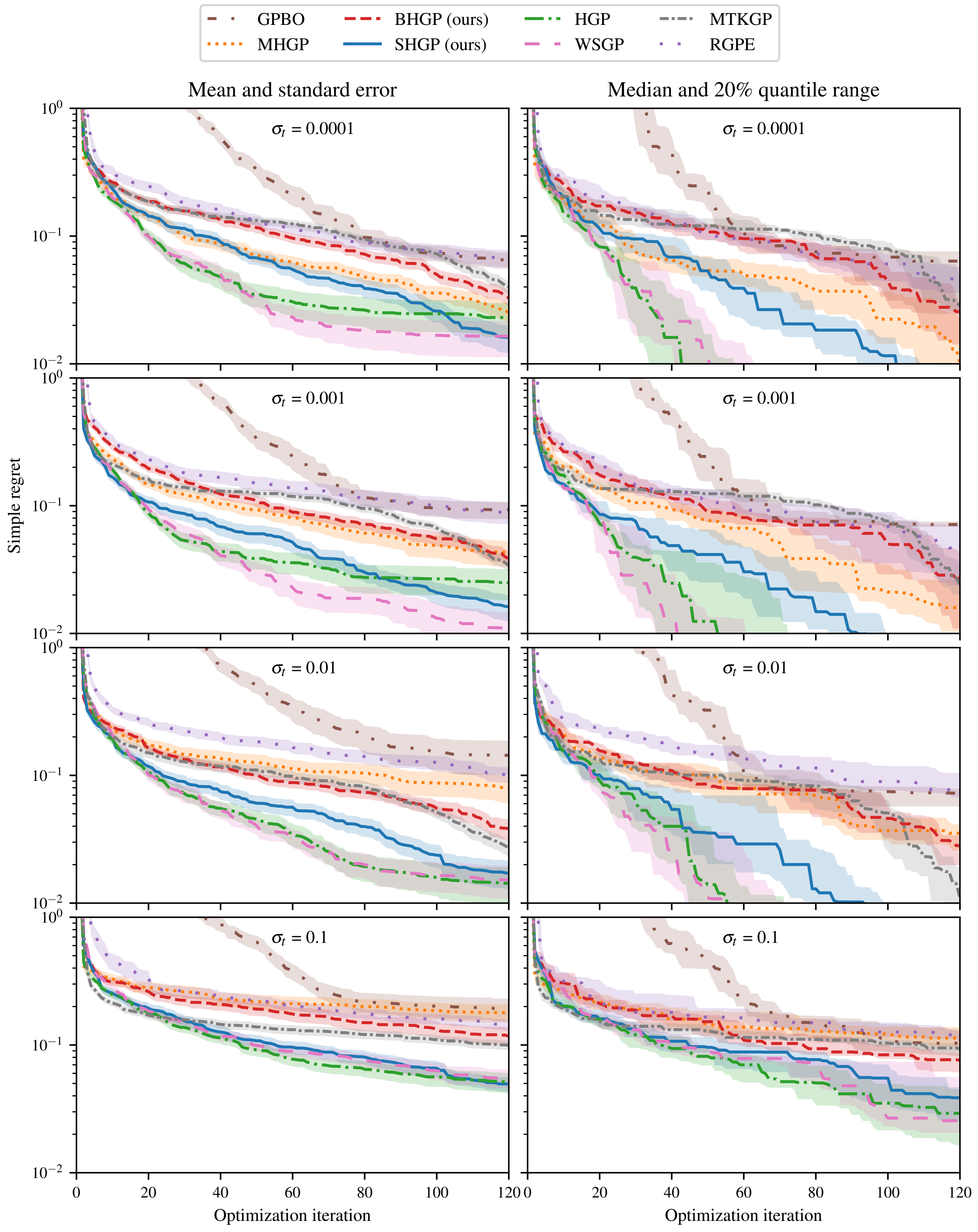

Figure 10: Performance on the single-source Hartmann6 function family for 250 points in the source data set. We vary the amount of observational noise, $\sigma_s = \sigma_t$, from 0.001 to 0.1. The left panel depicts the mean ± standard error of the mean. The right panel shows the median and 20% quantile range.

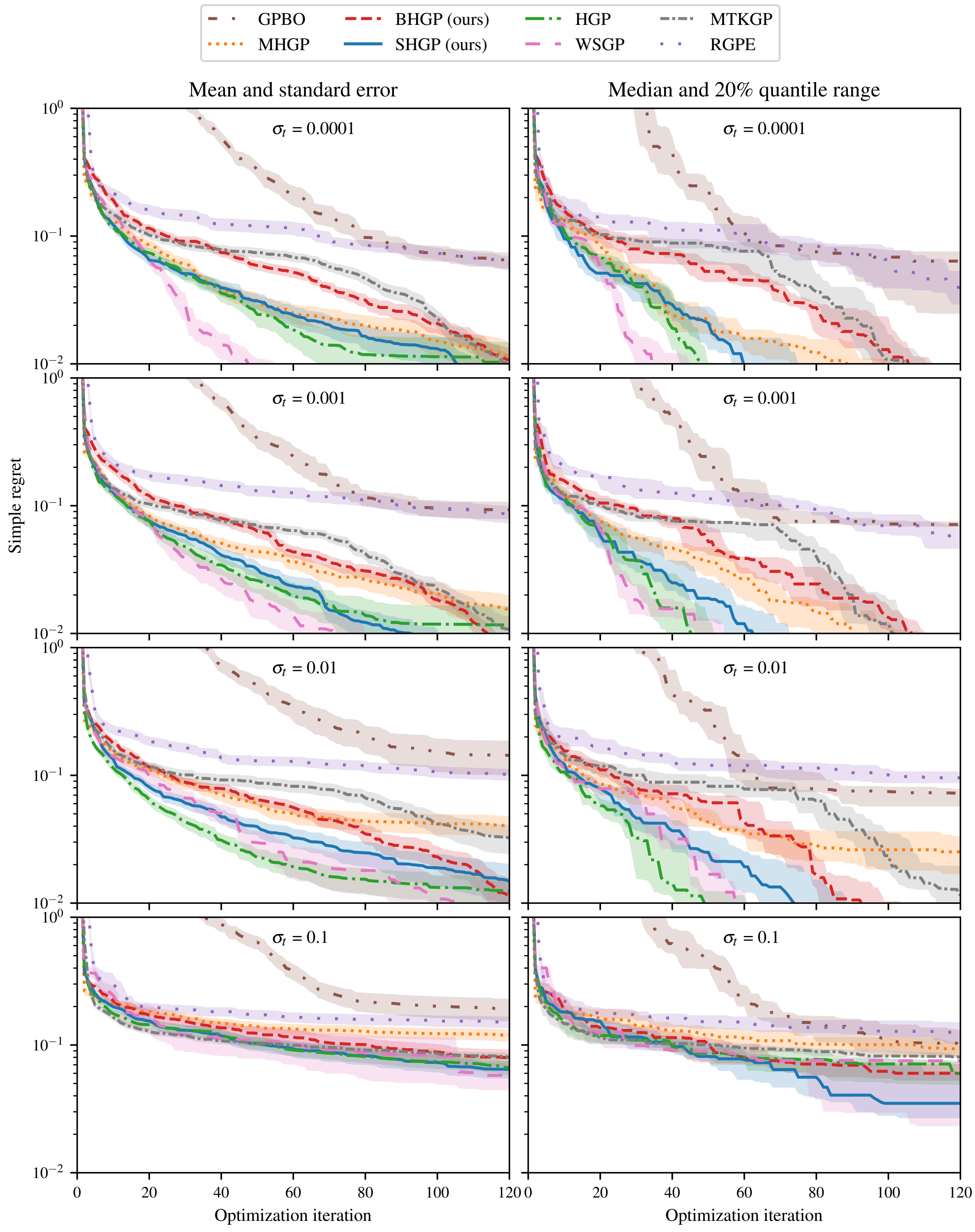

Figure 11: Performance on the single-source Hartmann6 function family for 500 points in the source data set. We vary the amount of observational noise, $\sigma_s = \sigma_t$, from 0.001 to 0.1. The left panel depicts the mean ± standard error of the mean. The right panel shows the median and 20% quantile range.

E IMPLEMENTATION AND COMPUTATIONAL RESOURCES

All experiments were run on a HPC cluster, where each individual experiment used a single Intel Xeon CPU. All experiments (including early debugging and evaluations) amounted to a total of 154353 hours, which corresponds to roughly 17.6 years if the jobs ran sequentially. Most of this compute was required to ensure reproducibility (multiple random seeds per job and ablation studies over the effects of parameters). The cluster is part of a carbon-neutral infrastructure and does not leave a carbon footprint.