Meta-Learning Reliable Priors in the Function Space

Jonas Rothfuss
ETH Zurich
jonas.rothfuss@inf.ethz.ch

Dominique Heyn
ETH Zurich
heynd@student.ethz.ch

Jinfan Chen
ETH Zurich
georgcjf@gmail.com

Andreas Krause
ETH Zurich
krausea@ethz.ch

Abstract

When data are scarce, *meta-learning* can improve a learner's accuracy by harnessing previous experience from related learning tasks. However, existing methods have unreliable uncertainty estimates which are often overconfident. Addressing these shortcomings, we introduce a novel meta-learning framework, called *F-PACOH*, that treats meta-learned priors as *stochastic processes* and performs meta-level regularization directly in the *function space*. This allows us to directly steer the probabilistic predictions of the meta-learner towards high *epistemic uncertainty* in regions of insufficient meta-training data and, thus, obtain well-calibrated uncertainty estimates. Finally, we showcase how our approach can be integrated with *sequential decision making*, where reliable uncertainty quantification is imperative. In our benchmark study on meta-learning for *Bayesian Optimization (BO)*, *F-PACOH* significantly outperforms all other meta-learners and standard baselines.

1 Introduction

Learning new concepts and skills from a small number of examples as well as adapting them quickly in face of changing circumstances is a key aspect of human intelligence. Unfortunately, our machine learning algorithms lack such adaptive capabilities. *Meta-Learning* [1, 2] has emerged as a promising avenue towards enabling systems to learn much more efficiently by harnessing experience from previous related learning tasks [3–8]. By meta-learning probabilistic prior beliefs, we not only make more accurate predictions when given a small amount of training data, but also improve the self-assessment machine learning algorithm in the form of *epistemic uncertainty* estimates [9–12]. Such uncertainty estimates are critical for sequential decision problems such as Bayesian optimization (BO) [13, 14] and reinforcement learning [15, 16] which require efficient information gathering and exploration.

However, in most practical settings, there are only few tasks available for meta-training. Hence we face the risk of overfitting to these few tasks [17] and, consequently impairing the performance on unseen tasks. To prevent this, recent work has proposed various forms of regularization on the meta-level [18, 12, 8]. While these methods are effective in preventing meta-overfitting for the mean predictions, they fail to do so for the associated uncertainty estimates, manifested in gross *overconfidence*. Such overconfidence is highly detrimental for downstream sequential decision tasks that rely on calibrated uncertainty estimates [19, 20] to perform sufficient exploration. For instance, if the global optimum of the true target function in BO lies outside the model's 95 % confidence bounds the acquisition algorithm may never query points close to the optimum and get stuck in sub-optimal solutions. We hypothesize that previous methods yield overconfident predictions since they do not meta-regularize the predictive distribution directly. Instead, they perform meta-regularization in some latent space, for example the method parameters, which is non-trivially associated with the resulting predictive distribution and thus may not have the indented regularization effects.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

To overcome the issue of overconfident predictions in meta-learning, we develop a novel approach that regularizes meta-learned priors *directly in the function space*. We build on the PAC-Bayesian PACOH framework [12] which uses a hyper-prior over the *latent* prior parameters for meta-regularization. However, we propose to define the hyper-prior as stochastic process, characterized by its marginal distributions in the *function* space, and make the associated meta-learning problem tractable by using an approximation of the KL-divergence between stochastic processes [21]. The functional KL allows us to directly steer the meta-learned prior towards high epistemic uncertainty in regions of insufficient meta-training data and, thus, obtain reliable uncertainty estimates. When instantiating our functional meta-learning framework, referred to as *F-PACOH*, with Gaussian Processes (GPs), we obtain a simple algorithm that can be seamlessly integrated into sequential decision algorithms.

In our experiments, we showcase how *F-PACOH* can facilitate transfer and life-long learning in the context of BO, and unlike previous meta-learning methods, consistently yields well-calibrated uncertainty estimates. In our benchmark study on meta-learning for BO and hyper-parameter tuning, *F-PACOH* significantly outperforms all other meta-learners and standard baselines. Finally, we consider lifelong BO, where the meta-BO algorithm faces a sequence of BO tasks and needs build-up prior knowledge iteratively. In this challenging setting, *F-PACOH* is the only method that is able to significantly improve its optimization performance as it gathers more experience. This paves the way for exciting new applications for meta-learning and transfer such as the recurring re-optimization and calibration of complex machines and systems under changing external conditions.

2 Background

In this section, we formally introduce PAC-Bayesian meta-learning, the foundation of the functional meta-learning framework that we develop in Section 4. Moreover, we provide a brief description of Bayesian Optimization, which serves as the main testing ground for our proposed method.

PAC-Bayesian Meta-Learning. Meta-learning extracts prior knowledge (i.e., inductive bias) from a set of related learning tasks to accelerate inference in light of a new task. In the context of supervised learning, the meta-learner is given n datasets $\mathcal{D}_{1,T_1}, ..., \mathcal{D}_{n,T_n}$. Each dataset $\mathcal{D}_{i,T_i} = (\mathbf{X}_i^{\mathcal{D}}, \mathbf{y}_i^{\mathcal{D}})$ consists of T_i noisy function evaluations $y_{i,t} = f_i(\mathbf{x}_{i,t}) + \epsilon$ corresponding to a function $f_i : \mathcal{X} \mapsto \mathcal{Y}$ and additive noise ϵ. In short, we write $\mathbf{X}_i^{\mathcal{D}} = (\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,T_i})^\top$ for the matrix of function inputs and $\mathbf{y}_i^{\mathcal{D}} = (y_{i,1}, ..., y_{i,T_i})^\top$ for the vector of corresponding observations. The functions $f_i \sim \mathcal{T}$ are sampled from a task distribution $\mathcal{T}$, which can be thought of as a stochastic process that governs a random function $f : \mathcal{X} \mapsto \mathcal{Y}$. In standard Bayesian inference, we exogenously presume an – often carefully *manually designed* – prior distribution $P(h)$ over learning hypotheses $h : \mathcal{X} \mapsto \mathcal{Y}, h \in \mathcal{H}$ and combine it with empirical data to form a posterior $Q(h) = P(h|\mathcal{D})$. In contrast, in meta-learning we endogenously infer the prior $P(h)$ in a *data-driven* manner, by using the provided meta-training data.

While there exist different approaches, we focus on *PAC-Bayesian meta-learning* [22, 23, 12] due to its principled foundation in statistical learning theory. The *PACOH* framework by Rothfuss et al. [12] presumes a loss function $l(h, \mathbf{x}, y)$ and a parametric family $\{P_\phi | \phi \in \Phi\}$ of priors $P_\phi(h)$ with hyperparameter space Φ. In a probabilistic setting, one typically uses the negative log-likelihood as the loss function, i.e., $l(h, \mathbf{x}, y) = -\ln p(y|h(\mathbf{x}))$. Given the meta-training data $\mathcal{D}_1, ..., \mathcal{D}_n$ and a *hyper-prior* distribution $\mathcal{P}(\phi)$ over Φ, PACOH aims to infer a *hyper-posterior* distribution $\mathcal{Q}(\phi)$ over the parameters ϕ of the prior. Using PAC-Bayesian learning theory [24], they derive a high-probability bound on the transfer error, i.e., the generalization error for posterior inference on an unseen task $f \sim \mathcal{T}$ with priors sampled from the hyper-posterior $\mathcal{Q}(\phi)$. This meta-level generalization bound also serves as an objective for the meta-learner, minimized w.r.t. $\mathcal{Q}$:

$$J(\mathcal{Q}) = -\frac{1}{n}\sum_{i=1}^{n} \frac{1}{T_i} \mathbb{E}_{\phi \sim \mathcal{Q}} \left[\ln Z_\beta(\mathcal{D}_{i,T_i}, P_\phi)\right] + \left(\frac{1}{\lambda} + \frac{1}{n\tilde{T}}\right) KL[\mathcal{Q}||\mathcal{P}] + \text{const.} \tag{1}$$

Here, $\lambda > 0$, $\tilde{T} = (T_1^{-1} + ... + T_n^{-1})^{-1}$ is the geometric mean of the dataset sizes, $\ln Z(\mathcal{D}_{i,T_i}, P) := \int_{\mathcal{H}} P(h) \exp\left(-\sum_{t=1}^{T_i} l(h, \mathbf{x}_{i,t}, y_{i,t})\right) dh$ the generalized marginal log-likelihood and $KL(\mathcal{Q}||\mathcal{P})$ the Kullback–Leibler (KL) divergence between hyper-posterior and hyper-prior. To obtain asymptotically consistent bounds, λ is typically chosen as $\lambda = \sqrt{n}$.

Bayesian Optimization. Bayesian Optimization (BO) aims to find the global maximizer $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ of a function $f : \mathcal{X} \to \mathbb{R}$ over a bounded domain $\mathcal{X}$. To obtain an estimate of $\mathbf{x}^*$, the BO algorithm iteratively chooses points $\mathbf{x}_1, ..., \mathbf{x}_T \in \mathcal{X}$ at which to query f, and observes noisy

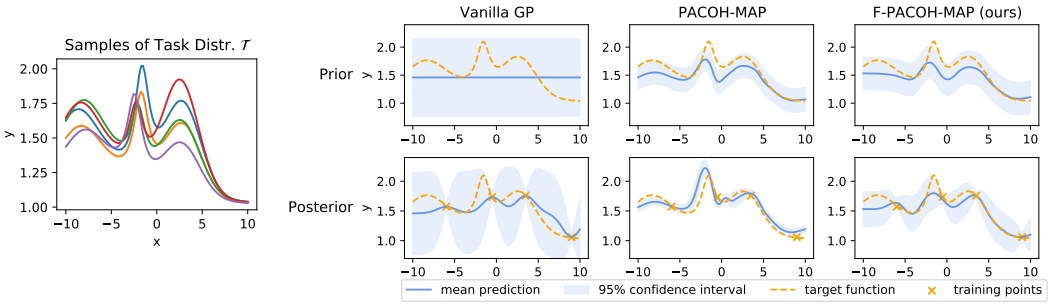

Figure 1: Prior / posterior predictions for a Vanilla GP and PACOH / F-PACOH GP meta-trained on functions from the task distribution displayed left. While PACOH yields over-confident and the Vanilla GP under-confident predictions, F-PACOH provides the well-calibrated confidence intervals.

feedback $y_1, ..., y_T \in \mathbb{R}$, e.g., via $y_t = f(\mathbf{x}_t) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2)$ with $\sigma^2 \geq$ [13, 14]. BO methods form a Bayesian *surrogate model* of the function f based on previous observations $\mathcal{D}_t = \{(\mathbf{x}_t, y_t)\}$. Typically, a *Gaussian Process (GP)* $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ with mean $m(\mathbf{x})$ and kernel function $k(\mathbf{x}, \mathbf{x}')$ is employed to form a posterior belief $p(f(\mathbf{x})|\mathcal{D}_t)$ over function values [25]. In each iteration, the next query point $\mathbf{x}_{t+1} := \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha_t(\mathbf{x})$ is chosen as maximizer of an *acquisition function* $\alpha_t(\mathbf{x})$ that is typically based on the $p(f(\mathbf{x})|\mathcal{D}_t)$ and trades-off exploration and exploitation [26–29].

3 Related Work

Meta-Learning. Common approaches in meta-learning attempt to learn a shared embedding space [3, 30–32] amortize inference by a meta-learned recurrent model [33–35] or learn the initialization of a NN so it can be quickly adapted to new tasks [6, 7, 36]. A growing body of work also uses probabilistic modeling to enable uncertainty quantification in meta-learning [9, 11, 37]. However, when only given a small number of meta-training tasks such approaches tend to overfit and fail to provide reliable uncertainty estimates. The problem of overfitting to the meta-training tasks has been brought to attention by [17, 38], followed by work that discusses potential solutions in the form of meta-regularization [8, 12, 18]. Our meta-learning framework builds on a sequence of work on PAC-Bayesian meta-learning [22, 23, 12]. However, instead of using a hyper-prior over the latent parameters of the learnable prior, we define the hyper-prior as stochastic process in the function space which gives us better control over the behavior of the predictive distributions of our meta-learned model in the absence of data.

Learning hypotheses in the function space. Recent work bridges the gap between stochastic processes and complex parametric models by developing a variational inference in the function space [21, 39]. Our framework in the function space heavily builds on these ideas; in particular, the proposed approximations of the KL-divergence between stochastic processes [21, 40, 41].

Meta-Learning and Transfer for Bayesian Optimization. To improve the sample efficiency of BO based on previous experience, a basic approach is to warm-start BO by initially evaluating configurations that have performed well on similar tasks in the past [42–44]. Another common theme is to form a global GP based model across tasks [45–48]. However, such global GPs quickly become computationally infeasible as they scale cubically in the number of tasks times the number samples per task or require hand-designed task features. Conceptually, the most similar to our work is the approach of learning a neural-network based feature map that is shared across tasks and used for BO based on a Bayesian linear regression model in the shared feature space [49, 50]. While these methods are highly scalable, they lack any form meta-level regularization and thus, in face of data-scarcity, are prone to overfitting and over-overconfident uncertainty estimates. In contrast, our method overcomes these issues thanks to its principled meta-regularization in the function space.

4 PAC-Bayesian Meta-Learning in the Function Space

In this section, we present our main contribution: a principled framework for meta-learning in the function space. First, we introduce the general framework, then we discuss particular instantiations and components of our approach and describe the resulting algorithm. Finally, we elaborate how our proposed meta-learner can be effectively applied in the context of Bayesian Optimization.

4.1 The F-PACOH Framework: Meta-Learning with Stochastic Process Hyper-Priors

We are interested in settings where the number of available meta-training tasks as well as observations may be small, and thus generalization beyond the meta-training data is challenging. In such scenarios, regularization on the meta-level plays a vital role to prevent meta-overfitting and ensure positive transfer [8, 17]. Due to its principled meta-level regularization grounded in statistical learning theory, we build on the PACOH meta-learning framework of [12] which has been introduced in Section 2.

A critical design choice when attempting to meta-learn with the PACOH objective in (1) is the hyper-prior $\mathcal{P}$ which, through the KL-divergence, becomes the dominant force that shapes the learned prior P_ϕ in the absence of sufficient meta-training data. Rothfuss et al. [12] define the hyper-prior as distribution over the prior parameters ϕ and, in particular, use a simple Gaussian $\mathcal{P}(\phi) = \mathcal{N}(\phi; 0, \sigma_{\mathcal{P}}^2 I)$. While this may act as a form of smoothness regularization on the prior, it is unclear how such a hyper-prior shapes the predictions of our meta-learned model in the function space, especially in regions where no training data is available. In such regions of data scarcity, we ideally want our meta-learned model to make conservative predictions, characterized by high epistemic uncertainty. As we can observe in Figure 1, PACOH fails to do so, yielding grossly over-confident uncertainty estimates. This is particularly problematic in sequential decision making, where data is typically non-i.i.d. and we heavily rely on well-calibrated epistemic uncertainty estimates to guide exploration.

Hyper-prior in function space. Aiming to make meta-learned models behave more predictably outside the support of the data, we devise a hyper-prior in the function space. We assume that the hyper-prior $\mathcal{P}$ is a stochastic process, indexed in $\mathcal{X}$ and taking values in $\mathcal{Y}$, i.e., a random function $h : \mathcal{X} \mapsto \mathcal{Y}$. Furthermore, we assume that for any finite measurement set $\mathbf{X} := [\mathbf{x}_1, ..., \mathbf{x}_k] \in \mathcal{X}^k, k \in \mathbb{N}$, the corresponding marginal distribution of function values $\rho(\mathbf{h}^{\mathbf{X}}) := \rho(h(\mathbf{x}_1), ...h(\mathbf{x}_k))$ exists and fulfills the exchangability and consistency conditions of the Kolmogorov Extension Theorem [51]. Similarly, we treat the prior P_ϕ as a stochastic process from which we can either sample parameters θ that correspond to functions $h_\theta : \mathcal{X} \mapsto \mathcal{Y}$ or we even have direct access to its finite marginals $p(\mathbf{h}^{\mathbf{X}}) = p(h(\mathbf{x}_1), ..., h(\mathbf{x}_k))$, e.g., multivariate normal distributions in the case of a GP. Likewise, function samples from the hyper-posterior can be obtained by hierarchical sampling ($h_\theta(\cdot)$ with $\theta \sim P_\phi, \phi \sim \mathcal{Q}$) and finite marginals by forming a mixture distribution $q(\mathbf{h}^{\mathbf{X}}) = \mathbb{E}_{\phi\sim\mathcal{Q}}\left[p_\phi(\mathbf{h}^{\mathbf{X}})\right]$.

Characterizing the prior, hyper-prior, and hyper-posterior as stochastic processes, we need to re-consider how to define and compute $KL[\mathcal{Q}||\mathcal{P}]$. For this purpose, we build on the result of Sun et al. [21] who show that the KL-divergence between two stochastic processes q and ρ can be expressed as a supremum of KL-divergences between their finite marginals:

$$KL[q, \rho] = \sup_{n\in\mathbb{N}, \mathbf{X}\in\mathcal{X}^n} KL[q(\mathbf{h}^{\mathbf{X}})||\rho(\mathbf{h}^{\mathbf{X}})] \tag{2}$$

This supremum is highly intractable, and without further specifications on the measurement set $\mathbf{X}$ it does not present a viable optimization objective [40, 41]. We thus follow a sample-based approach to computing (2), for which Sun et al. [21] provide consistency guarantees.

Enforcing the hyper-prior. A-priori, we want our meta-learned priors to match the structure of the hyper-prior both near the meta-training data and in regions of the domain where no data is available at all. Thus, for each task, we build measurement sets $\mathbf{X}_i = [\mathbf{X}_{i,s}^{\mathcal{D}}, \mathbf{X}_i^M]$ by selecting a random subset $\mathbf{X}_{i,s}^{\mathcal{D}}$ of the meta-training inputs $\mathbf{X}_i^{\mathcal{D}}$ as well as random points $\mathbf{X}_i^M \overset{iid}{\sim} \mathcal{U}(\mathcal{X})$ sampled independently and uniformly from the bounded domain $\mathcal{X}$. In expectation over these random measurement sets, we then compute the KL-divergence between the marginal distributions of the stochastic processes, giving us $\mathbb{E}_{\mathbf{X}_i}[KL\left[q(\mathbf{h}^{\mathbf{X}_i})||\rho(\mathbf{h}^{\mathbf{X}_i})\right]]$ as approximation of (2). Intuitively, to obtain a low expected KL-divergence, q must closely resemble the behavior our stochastic process ρ hyper-prior across the entire domain. Hence, in the absence of meta-training data, the hyper-prior gives us direct control over the a-priori behavior of our meta-learner in the function space. In Figure 1, we can observe how a Vanilla GP as hyper-prior shapes the predictions of our meta-learned model.

In summary, by defining the hyper-prior in the function space and using sampling-based measurement sets $\mathbf{X}_i$ to compute the functional KL divergence, we obtain the following meta-learning objective:

$$J_F(\mathcal{Q}) = \frac{1}{n}\sum_{i=1}^{n}\left(-\frac{1}{T_i}\mathbb{E}_{\phi\sim\mathcal{Q}}[\underbrace{\ln Z(\mathcal{D}_{i,T_i}, P_\phi)}_{\text{marginal log-likelihood}}] + \left(\frac{1}{\sqrt{n}} + \frac{1}{nT_i}\right)\underbrace{\mathbb{E}_{\mathbf{X}_i}\left[KL[q(\mathbf{h}^{\mathbf{X}_i})||\rho(\mathbf{h}^{\mathbf{X}_i})]\right]}_{\text{functional KL-divergence}}\right) \tag{3}$$

In here, the marginal log-likelihood forces the meta-learned prior towards a good fit of the meta-training data while the functional KL-divergence pushes the prior towards resembling the hyper-prior's behavior in the function space. Since $J_F(\mathcal{Q})$ is inspired by the PACOH framework of [12] but works with meta-learning hypotheses in the function space, we refer to algorithms that minimize $J_F(\mathcal{Q})$ as *Functional-PACOH (F-PACOH)*.

4.2 Instantiations and Components of the F-PACOH framework

In the following, we discuss various instantiations of the introduced F-PACOH framework and their implications on the two main components of the meta-learning objective in (3) — the (generalized) marginal log-likelihood and the functional KL-divergence.

Representing the hyper-posterior $\mathcal{Q}$. To optimize the functional meta-learning objective J_F w.r.t. $\mathcal{Q}$ we may choose a variational family of hyper-posteriors $\{\mathcal{Q}_\xi(\phi), \xi \in \Xi\}$ from which we can sample ϕ in a re-parameterizable manner. This allows us to obtain low-variance gradient estimates of the expectations $\nabla_\xi \mathbb{E}_{\phi\sim\mathcal{Q}_\xi}[\ln Z(\mathcal{D}_{i,T_i}, P_\phi)]$ and $\nabla_\xi \ln q_\xi(\mathbf{h}^\mathbf{X}) = \nabla_\xi \ln \mathbb{E}_{\phi\sim\mathcal{Q}_\xi}\left[p_\phi(\mathbf{h}^\mathbf{X})\right]$.

Alternatively, we can form a maximum a posteriori (MAP) estimate of the hyper-posterior which approximates $\mathcal{Q}$ by a Dirac delta function $\hat{\mathcal{Q}}(\phi) = \delta(\phi - \hat{\phi})$ in a single prior parameter $\hat{\phi}$. As a result, the corresponding expectations become trivial to to solve, turning (3) into a much simpler objective to minimize and making the overall meta-learning approach more practical. Thus, we focus on the MAP approximation in main body of the paper and discuss variational hyper-posteriors in Appx. A.

The marginal log-likelihood. In case of GPs, the marginal log-likelihood $\ln Z(\mathcal{D}_{i,T_i}, P_\phi) = \ln p(\mathbf{y}_i^\mathcal{D}|\mathbf{X}_i^\mathcal{D}, \phi)$ can be computed in closed form as (see Appx. A.1). In most other cases, e.g. when the hypothesis space $\mathcal{H} = \{h_\theta, \theta \in \Theta\}$ corresponds to the parameters θ of a neural network, we need to form an approximation of the (generalized) marginal log-likelihood $\ln p(\mathbf{y}^\mathcal{D}|\mathbf{X}^\mathcal{D}, \phi) = \ln \mathbb{E}_{\theta\sim P_\phi}\left[e^{-\sum_{t=1}^{T_i} l(h_\theta(\mathbf{x}_{i,t}), y_{i,t})}\right]$. For further discussions on this matter, we refer to [12, 52, 53].

Gradients of the KL-divergence. Generally, we only require re-parametrizable sampling of functions from our prior. Following [21], we can write the gradients of the KL-divergence as

$$\nabla_\phi KL[p(\mathbf{h}^\mathbf{X})||\rho(\mathbf{h}^\mathbf{X})] = \mathbb{E}_{\mathbf{h}^\mathbf{X}\sim p_\phi}\left[\nabla_\phi \mathbf{h}^\mathbf{X}\left(\nabla_\mathbf{h} \ln p_\phi(\mathbf{h}^\mathbf{X}) - \nabla_\mathbf{h} \ln \rho(\mathbf{h}^\mathbf{X})\right)\right] \tag{4}$$

Here, $\nabla_\phi \mathbf{h}^\mathbf{X}$ is the Jacobian of a function sample $\mathbf{h}^\mathbf{X}$ w.r.t. the prior parameters θ. Thus, it remains to estimate the score of the prior $\nabla_\mathbf{h} \ln p_\phi(\mathbf{h}^\mathbf{X})$ and the score of hyper-prior $\nabla_\mathbf{h} \ln \rho_\phi(\mathbf{h}^\mathbf{X})$. In various scenarios, the marginal distributions $p(\mathbf{h}^\mathbf{X})$ or $\rho(\mathbf{h}^\mathbf{X})$ in the function space may be intractable and we can only sample from it. For instance, when our prior $P_\phi(\theta)$ is a known distribution over neural network (NN) parameters θ, associated with NN functions $h_\theta : \mathcal{X} \mapsto \mathcal{Y}$, its marginals $p_\phi(\mathbf{h}^\mathbf{X})$ in the function space are typically intractable. In such cases, we can use either the Spectral Stein Gradient Estimator (SSGE) [54] or sliced score matching [55] to estimate the respective score from samples. Whenever the marginal densities are available in closed form, we use automatic differentiation to compute their score. Finally, if both the prior and the hyper-prior are GPs, we use the closed-form KL-divergence between multivariate normal distributions.

4.3 The F-PACOH-MAP Algorithm

When focusing on the MAP approximation of the hyper-posterior, where we directly meta-learn the prior's parameter vector ϕ, we arrive at a simple algorithm for meta-learning reliable priors in the function space. After initializing P_ϕ, we iteratively perform stochastic gradient steps on the functional meta-learning objective $J_F(\phi)$ in (3). In each step, we iterate through all the meta-training tasks, compute the corresponding (generalized) marginal log-likelihood $\ln Z(\mathcal{D}_{i,T_i}, P_\phi)$, sample a measurement set $\mathbf{X}_i$ and estimate the gradient of the functional KL-divergence $D_{\mathrm{KL}}[p(\mathbf{h}^{\mathbf{X}_i})||\rho(\mathbf{h}^{\mathbf{X}_i})]$ based on $\mathbf{X}_i$. In accordance with (3), we compute a weighted sum of these components and average the resulting gradients over the tasks. Finally, we use our gradient estimate $\nabla_\phi J_F(\phi)$ to perform a gradient update on ϕ. The overall procedure, which we denote by F-PACOH-MAP, is summarized in Algorithm 1.

While the proposed function-space approach brings us many benefits, it also comes at a cost: Estimating the expectation over measurement sets $\mathbf{X}_i$ by uniform sampling and Monte Carlo estimation is subject to the curse of dimensionality. Hence, we do not expect it to work well for high-dimensional data (d > 50) such as images. For such purposes, future work may investigate alternative approximation / sampling schemes that take the data manifold into account.

Algorithm 1 F-PACOH-MAP: Meta-Learning Reliable Priors

Input: Datasets $\mathcal{D}_{1,T_1}, ..., \mathcal{D}_{n,T_n}$, parametric family $\{P_\phi | \phi \in \Phi\}$ of priors, learning rate α
Input: Stochastic process hyper-prior with marginals $\rho(\cdot)$
Initialize the parameters ϕ of prior P_ϕ
while not converged **do**
  for $i = 1, ..., n$ **do** ▷ Iterate over meta-training tasks
    $\mathbf{X}_i = [\mathbf{X}_{i,s}^{\mathcal{D}}, \mathbf{X}_i^M]$, where $\mathbf{X}_{i,s}^{\mathcal{D}} \subseteq \mathbf{X}_i^{\mathcal{D}}, \mathbf{X}_i^M \overset{iid}{\sim} \mathcal{U}(\mathcal{X})$ ▷ Sample measurement set
    Estimate or compute $\nabla_\phi \ln Z(\mathbf{X}_i^{\mathcal{D}}, P_\phi)$ and $\nabla_\phi KL[p_\phi(\mathbf{h}^{\mathbf{X}_i}) || \rho(\mathbf{h}^{\mathbf{X}_i})]$
    $\nabla_\phi J_{F,i} = -\frac{1}{T_i} \nabla_\phi \ln Z(\mathcal{D}_{i,T_i}, P_\phi) + \left(\frac{1}{\sqrt{n}} + \frac{1}{nT_i} \right) \nabla_\phi KL[p(\mathbf{h}^{\mathbf{X}_i}) || \rho(\mathbf{h}^{\mathbf{X}_i})]$
  end for
  $\phi \leftarrow \phi - \alpha \, \frac{1}{n} \sum_{i=1}^n \nabla_\phi J_{F,i}$ ▷ Update prior parameter
end while

4.4 Application: Meta-Learning Reliable GP Priors for Bayesian Optimization

To harness the reliable uncertainty estimates of our proposed meta-learner towards improving sequential-decision making, we employ it in the context of BO. We assume that we either face a sequence of related BO problems corresponding to n target functions $f_1, ..., f_n \sim \mathcal{T}$ or have access to the function evaluations from multiple such optimization runs. We use the previously collected function evaluations for meta-training with F-PACOH. Then we employ the UCB algorithm [26, 27] together with our meta-learned model to perform BO on a new target function $f \sim \mathcal{T}$.

To be able to extract sufficient prior knowledge from the data of previous BO runs, we need to choose a sufficiently rich parametrization of the GP prior $P_\phi(h) = \mathcal{GP}\left(h | m_\phi(x), k_\phi(x, x')\right)$. Hence, following [56, 38], we instantiate m_ϕ and k_ϕ as neural networks (NN), where the parameter vector ϕ corresponds to the weights and biases of the NN. To ensure the positive-definiteness of the kernel, we use the neural network as feature map $\Phi_\phi(x)$ on top of which we apply a squared exponential kernel.

In the standard BO setting, we would usually use a Vanilla GP with constant mean function and a conservative SE or Matérn kernel. Hence, in the absence of sufficient meta-training data, we ideally want to fall back on the behavior of such a Vanilla GP. For this reason, we use a GP prior with zero-mean and SE kernel with a small lengthscale as hyper-prior[1]. Correspondingly, the marginals $p_\phi(\mathbf{h}^{\mathbf{X}}) = \mathcal{N}(\mathbf{m}_{\mathbf{X},\phi}, \mathbf{K}_{\mathbf{X},\phi})$ of the prior and the hyper-prior $\rho(\mathbf{h}^{\mathbf{X}}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{X},SE})$ are multivariate normal distributions. Thus, the marginal log-likelihood (see Section A.1) and the KL-divergence are available in closed form. We provide more details in Appx. A.4 and Algorithm 3. Due to the closed-form computations, the resulting FPACOH-MAP algorithm for GPs has an asymptotic runtime complexity of $\mathcal{O}(nT^3 + nL^3)$ per iteration. In that, $T = \max_i T_i$ is the maximal number of observations in one of the meta-training datasets and $L = \max_i |\mathbf{X}_i|$ is the number of points per measurement set which can be chosen freely. In our experiment, we use $L = 20$. However, BO is most relevant when function evaluations are costly and data is scarce, i.e., both T and n are small. Thus, the cubic runtime is hardly a concern in practice. Alternatively, sparse GP approximations can be used to reduce the runtime complexity [57, 58]

5 Experiments

First, we investigate the effect of our functional meta-regularization on the uncertainty estimates of the meta-learned model. To assess the utility of the uncertainty for sequential decision making, we then evaluate our F-PACOH approach in meta-learning for BO as well as a challenging life-long BO setting. Details on the experiments can be found in Appendix B.

5.1 Calibration and Uncertainty Quantification

To study and illustrate our proposed meta-regularization in the function space, we use a simulated task distribution of 1-dimensional functions. Random samples from this task distribution are displayed left in Fig. 1. Using F-PACOH-MAP and PACOH-MAP [12], we meta-train a GP with $n = 10$ tasks and $T = 10$ function evaluations per task, collected with Vanilla GP-UCB [27]. PACOH-MAP corresponds to a MAP approximation of (1) with a Gaussian hyper-prior on the prior parameters ϕ.

[1]Note that we standardize the inputs $\mathbf{x}$ and observations y based on the empirical mean and variance of the meta-training data. The GP prior is applied in the standardized data space

Fig. 1 displays the prior and posterior predictions of the meta-learned models along those of a Vanilla GP with zero mean and SE kernel. We observe that the 95 % confidence intervals of the PACOH posterior are strongly concentrated around the mean predictions, even far away from the training points. Unlike a Vanilla GP whose confidence regions contract locally around the training points, the uncertainty estimates of PACOH contract uniformly across the domain, even far away from the training data. The fact that the true target function lies mostly outside of the confidence intervals reflects the over-confidence of PACOH-MAP. In contrast, while also reflecting useful meta-learned prior knowledge, the *predictions of the F-PACOH model exhibit the behavior we desire*, i.e., small uncertainty close to the training points, higher uncertainty far away from the data.

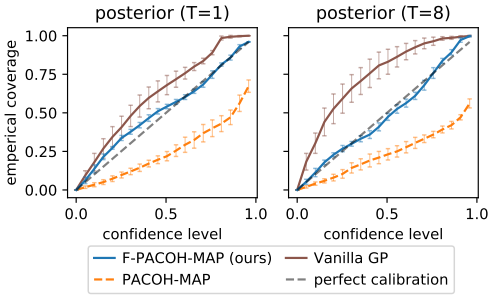

Figure 2: Calibration plots of posterior predictions corresponding to $T = 1$ and 8 training points. Only F-PACOH yields well-calibrated predictions.

How reliable uncertainty estimates are can be quantified by the concept of *calibration* [19, 20]. We say that a probabilistic predictor is calibrated, if, in expectation, its $\alpha\%$ confidence intervals contain $\alpha\%$ of the true function values. Fig. 2 visualizes this by plotting the fraction of the true function values covered by the posterior's confidence intervals at varying levels of α. Here, the Vanilla-GP model is consistently under-confident, while PACOH-MAP makes grossly over-confident predictions. In contrast, F-PACOH yields well-calibrated uncertainty estimates across the entire range of confidence levels. For a more thorough analysis, Table 5 in Appendix B.5.1 reports the calibration error for all the methods and BO environments, presented Section 5.2. There, F-PACOH yields significantly lower calibration errors than the other methods in the majority of the environments. All in all, this empirically supports our claim that, through its meta-regularization in the function space, *F-PACOH is able to meta-learn priors that yield reliable uncertainty estimates*.

5.2 Meta-Learning for Bayesian Optimization: Setup of the Benchmark Study

We present a comprehensive benchmark study on meta-learned priors and multi-task methods for BO in which we compare our proposed method F-PACOH-MAP against various related approaches. We use the UCB aquisition algorithm [26, 27] across all the models.

Baselines. We compare against approaches that also meta-learn a GP prior. *PACOH-MAP* [12] is the most similar to our approach as it meta-learns a neural-network based GP prior. However, it uses a hyper-prior over prior parameters ϕ instead of our stochastic process formulation. Similarly, *ABLR* [50] meta-learns the feature map and prior of a Bayesian linear regression model. As a simple baseline, we meta-train a GP prior with constant mean and SE kernel by maximizing the sum of marginal log-likelihoods across tasks *(Learned GP)*. We also compare with neural processes *(NP)* [11] and rank-weighted GPs *(RWGP)* [47]. Finally, we use a *Vanilla GP* as a baseline that does not perform transfer across tasks and, if the regret definition permits[2], also report the performance of *Random Search*.

Simulated Benchmark Environments. We use three simulated function environments as well as three hyper-parameter optimization environments as benchmarks. Two of the simulated environments are based on the well-known *Branin* and *Hartmann6* functions for global optimization [59]. Following [60], we replace the function parameters with distributions over them in order to obtain a task distribution. Another simulated function is based on the *Camelback* function [61] which we overlay with a product of sinusoids of varying amplitude, phase and shifts along the two dimensions.

Hyper-Parameter Optimization for machine learning algorithms. As practical application of meta-learning for BO, we consider hyper-parameter tuning of machine learning algorithms on different datasets. In this setting, the domain $\mathcal{X}$ is an algorithm's hyper-parameter space and the target function $f(\mathbf{x})$ represents the test performance under the hyper-parameter configuration $\mathbf{x}$. The different functions $f_i \sim \mathcal{T}$ correspond to training and testing the machine learning algorithm on different datasets. The goal is to gather knowledge from hyper-parameter optimizations on different datasets so that tuning of the same algorithm on a new dataset can be done more efficiently.

[2]Since random search performs no model-based inference, we can only report its simple regret.

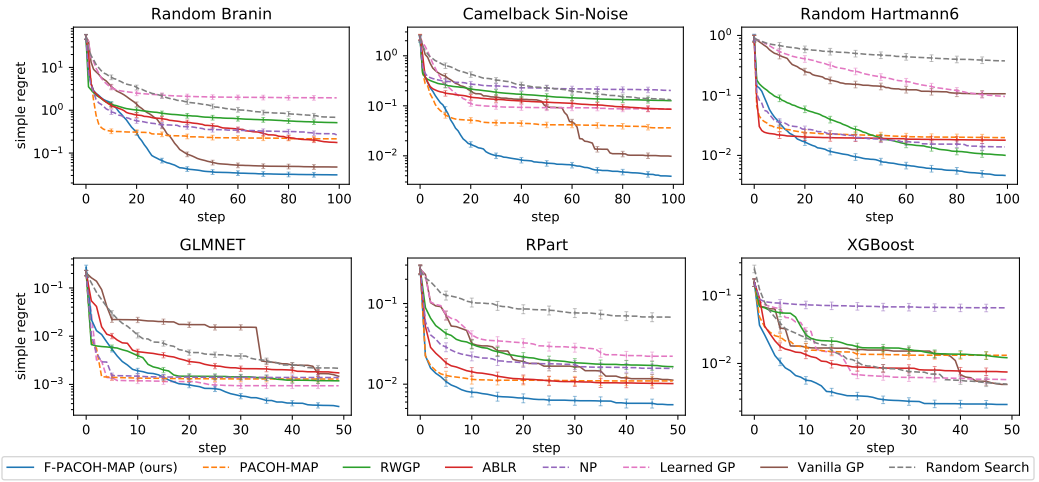

Figure 3: Performance of BO with meta-learned models on simulated function environments (top) and hyper-parameter tuning (bottom). Reported is the simple regret, averaged over seeds and function samples, alongside 95% confidence intervals. While other methods improve slowly or saturate in sub-optimal solutions, BO based on F-PACOH consistently finds near-optimal solutions quickly.

In particular, we consider three machine learning algorithms for this purpose: Generalized linear models with elastic NET regularization *(GLMNET)* [62], recursively partitioning trees *(RPart)* [63, 64] and *XGBoost* [65]. Following previous work [e.g. 50, 60], we replace the costly training and evaluation step by a cheap table lookup based on a large number of hyper-parameter evaluations [66] on 38 classification datasets from the OpenML platform [67]. In these hyper-parameter experiments, we use the area under the ROC curve (AUROC) as test metric to optimize.

5.3 Meta-Learning for BO: Offline Meta-Training

The first scenario we consider is where we have offline access to data of previous BO runs. In particular, using Vanilla GP-UCB, we collect T_i function evaluations on n tasks sampled from the task distribution $\mathcal{T}$. Depending on the dimensionality of $\mathcal{X}$ we vary n and T_i across the task distributions (see Tab. 1 in Appx. B). With this meta-training data $\mathcal{D}_{1,T_i}, ..., \mathcal{D}_{1,T_n}$, we meta-train F-PACOH and the considered baselines. To obtain statistically robust results, we perform independent BO runs on 10 unseen target functions / tasks and we repeat the whole meta-training & -testing process for 25 random seeds to initialize the meta-learner. To assess the performance, we report the simple regret $r_{f,t} = f(\mathbf{x}^*) - \max_{t' \leq t} f(\mathbf{x}_{t'})$ as well as the inference regret $\hat{r}_{f,t} = f(\mathbf{x}^*) - f(\hat{\mathbf{x}}_t^*)$, wherein $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ is the global optimum, $\mathbf{x}_t$ the point the BO algorithm chooses to evaluate in iteration t and $\hat{\mathbf{x}}_t^*$ is the optimum predicted by the model at time t.

Fig. 3 displays the results. The inference regret is reported in Fig. 5 in Appx. B and reflects the same patterns. While PACOH-MAP yields relatively low regret after few iterations, it saturates in performance very early on and gets stuck in sub-optimal solutions. This supports our hypothesis that meta-learning with a hyper-prior in the parameter space leads to unreliable uncertainty estimates that result in poor BO solutions. Similar early saturation behavior can be observed for the other the other meta-learners, i.e., ABLR, NPs and Learned GPs. F-PACOH also yields good solutions quickly, but then *continues to improve* throughout the course of the optimization. Across all the environments, it yields *significantly lower regret* and *better final solutions* than the other methods. This demonstrates that, due to our novel meta-level regularization in the function space, F-PACOH achieves strongly positive transfer in BO without loosing the reliability and long-run performance of well-established algorithms like GP-UCB.

5.4 Lifelong Bayesian Optimization

Finally, we consider the scenario of lifelong BO where we face a sequence $f_0, ..., f_{N-1} \sim \mathcal{T}$ of related target functions which we want to optimize efficiently, one after another. This is a common problem setup when complex systems / machines such as the Swiss Free-Electron laser (SwissFEL) [68] that are subject to external and internal effects (e.g., drift, hysteresis). As a result, the system / machine responds differently to parameter configurations over time and needs to be

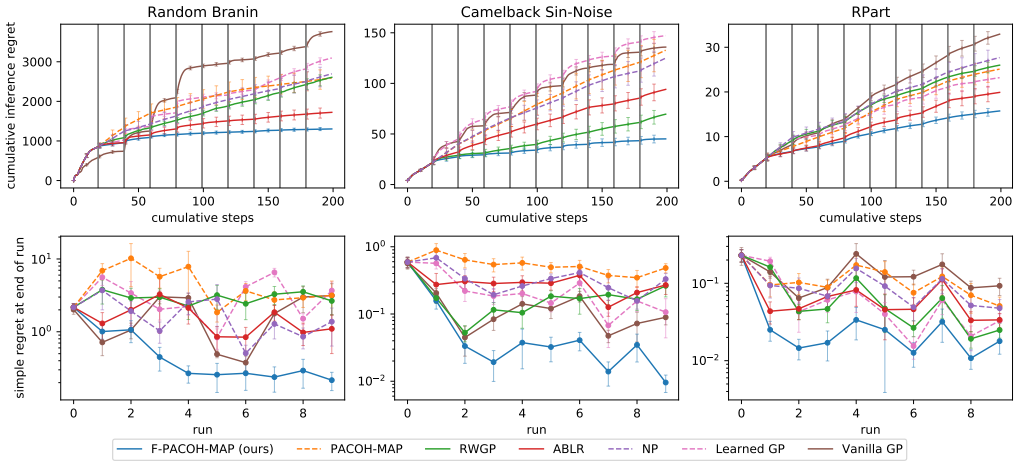

Figure 4: Lifelong BO performance on two simulated function environments and one hyper-parameter tuning benchmark (RPart). The reported results are averages over seeds and random sequences of tasks alongside 95% confidence intervals. While other meta-learners struggle to achieve positive transfer, F-PACOH is able to significantly improve the BO perfomance as it gathers more experience.

re-calibrated/optimized on a frequent basis. Our aim is to incrementally improve our performance from run to run by meta-learning with data collected in previous BO runs. This setup is more challenging than gathering meta-training offline for two reasons: 1) in the beginning, the meta-learner has much fewer meta-training tasks and 2) the meta-learned models and the collected meta-train data become interdependent, resulting in a feedback-loop between predictions and data collection on the task level. In short, this can be seen as an *exploration-exploitation trade-off on the meta-level*, which our calibrated uncertainty helps to effectively navigate.

Overall, we sequentially conduct $n = 10$ BO runs with $T_i = 20$ steps each. In the initial BO run ($i = 0$), we start without meta-training data, i.e. $\mathcal{M}_0 = \emptyset$ and thus use Vanilla GP-UCB. After each run, we add the collected function evaluations to the meta-training data, i.e., $\mathcal{M}_{i+1} = \mathcal{M}_t \cup \{D_{i,T}\}$. For the following runs ($i > 0$), we first perform meta-training with $\mathcal{M}_i$ and then run BO with the meta-learned model. As performance metric, we compute the cumulative inference regret, i.e., the sum of inference regrets of all the previous steps and runs as well as the simple regret $r_{f_i,20}$ at the end of each run. We repeat the experiment for 5 random sequences of target functions and 5 random model seeds each.

Figure 4 displays the results of our lifelong BO study for the Random Branin, Camelback Sin-Noise and RPart environment. Similar results for more environments can be found in Appx. B. At first glance, meta-learning seems to result in lower cumulative inference regret overall. However, this is mainly due to the fact that the meta-learned models start with better initial predictions which, in case of most meta-learning baselines, hardly improve within a run. Similarly, we observe that the majority of meta-learning baselines fail to consistently find better solution than a Vanilla GP. F-PACOH in stark contrast is able to *significantly improve the BO perfomance* as it gathers more experience, and *finds better solutions* by the end of a run than the other methods. This further highlights the reliability of our proposed method. Finally, the fact that F-PACOH shows strong positive transfer despite this challenging setting is highly promising, as many real-world applications may benefit from it.

6 Conclusion

We have introduced a novel meta-learning framework that treats meta-learned priors as stochastic processes and performs meta-level regularization directly in the function space. This gives us much better control over the behavior of predictive distribution of the meta-learned model beyond the training data. Our experiments empirically confirm that the resulting *F-PACOH* meta-learning method alleviates the major issue of over-confidence plaguing prior work, and yields well-calibrated confidence intervals. Our extensive experiments on lifelong learning for Bayesian Optimization demonstrate that *F-PACOH* is able to facilitate strong positive transfer in challenging sequential decision problems – well beyond what existing meta-learning methods are able to do. This opens many new avenues for exciting applications such as continually improving the efficiency with which we re-calibrate/-optimize complex machines and systems.

Broader Impact

Our work focuses on meta-learning reliable priors with a small number of meta-tasks and thus a the potential to implact applications that can be cast in such as setting. Since it ensures more reliable uncertainty estimates, the proposed method is of particular interest for interactive machine learning systems that actively gather information, e.g., active learning, Bayesian optimization and reinforcement learning. For instance, the meta-learning for BO method featured in Section 5.3 and 5.4 could be applied in robotics for tuning controllers more efficiently, or in biochemistry / molecular medicine to develop novel therapeutics through protein optimization. Other potential applications include recommender systems and targeted advertisement. While improving medical applications and the control of robots promises positive impact, misuse can never be avoided.

Acknowledgments and Disclosure of Funding

This project received funding from the Swiss National Science Foundation under NCCR Automation under grant agreement 51NF40 180545, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program grant agreement no. 815943, and was supported by Oracle Cloud Services. Moreover, we thank Sebastian Curi and Lars Lorch for their valuable feedback.

References

[1] Juergen Schmidhuber. *Evolutionary principles in self-referential learning. On learning how to learn: The meta-meta-... hook*. PhD thesis, Technische Universitaet Munchen, 1987.

[2] Sebastian Thrun and Lorien Pratt, editors. *Learning to Learn*. Kluwer Academic Publishers, 1998.

[3] Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 2000.

[4] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[5] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, Timothy Lillicrap, and Google Deepmind. Meta-Learning with Memory-Augmented Neural Networks. In *International Conference on Machine Learning*, 2016.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

[7] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. ProMP: Proximal Meta-Policy Search. In *International Conference on Learning Representations*, 2019.

[8] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *International Conference on Learning Representations*, 2020.

[9] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 2018.

[10] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pages 7332–7342, 2018.

[11] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, volume 80, pages 1704–1713, 2018.

[12] Jonas Rothfuss, Vincent Fortuin, Martin Josifoski, and Andreas Krause. PACOH: Bayes-optimal meta-learning with PAC-guarantees. In *International Conference for Machine Learning (ICML)*, 2021.

[13] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[14] Peter I. Frazier. A Tutorial on Bayesian Optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[15] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[16] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Machine Learning Research*, 4:237–285, 1996.

[17] Yunxiao Qin, Weiguo Zhang, Chenxu Zhao, Zezheng Wang, Hailin Shi, Guojun Qi, Jingping Shi, and Zhen Lei. Rethink and redesign meta learning. *arXiv*, 2018.

[18] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. MetaReg: Towards Domain Generalization using Meta-Regularization. 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., 2018.

[19] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

[20] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate Uncertainties for Deep Learning Using Calibrated Regression. In *International Conference on Machine Learning*, 2018.

[21] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. In *International Conference on Learning Representations*, 2019.

[22] Anastasia Pentina and Christoph Lampert. A PAC-Bayesian bound for lifelong learning. In *International Conference on Machine Learning*, 2014.

[23] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *International Conference on Machine Learning*, 2018.

[24] Benjamin Guedj. A primer on PAC-Bayesian learning. In *2nd Congress of the French Mathematical Society*, 2019.

[25] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes in machine learning*. MIT Press, 2006.

[26] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[27] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *International Conference on Machine Learning*, pages 1015–1022, 07 2010.

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

[29] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

[30] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[31] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

[32] James Harrison, Apoorva Sharma, and Marco Pavone. Meta-learning priors for efficient online bayesian regression. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 318–337. Springer, 2018.

[33] Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning To Learn Using Gradient Descent. In *International Conference on Artificial Neural Networks*, 2001.

[34] Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *arXiv*, 2016.

[35] 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 *International Conference on Machine Learning*, 2017.

[36] Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv*, 2018.

[37] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 2018.

[38] Vincent Fortuin and Gunnar Rätsch. Deep mean functions for meta-learning in gaussian processes. *arXiv preprint arXiv:1901.08098*, 2019.

[39] Chao Ma, Yingzhen Li, and José Miguel Hernández-Lobato. Variational implicit processes. In *International Conference on Machine Learning*, pages 4222–4233. PMLR, 2019.

[40] Alexander Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the kullback- leibler divergence between stochastic processes. *Journal of Machine Learning Research*, 51:231–239, 2016.

[41] David R Burt, Sebastian W Ober, Adrià Garriga-Alonso, and Mark van der Wilk. Understanding variational inference in function-space. *arXiv preprint arXiv:2011.09421*, 2020.

[42] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Using meta-learning to initialize bayesian optimization of hyperparameters. In *MetaSel@ ECAI*, pages 3–10. Citeseer, 2014.

[43] Marius Lindauer and Frank Hutter. Warmstarting of model-based algorithm configuration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[44] Y. Wei, P. Zhao, Huaxiu Yao, and J. Huang. Transferable Neural Processes for Hyperparameter Optimization. *arXiv preprint arXiv:1909.03209*, 2019.

[45] 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.

[46] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google Vizier: A Service for Black-Box Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 1487–1495, New York, NY, USA, 2017. Association for Computing Machinery.

[47] 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.

[48] Ho Chung Leon Law, Peilin Zhao, Lucian Chan, Junzhou Huang, and Dino Sejdinovic. Hyperparameter learning via distributional transfer. In *Advances in Neural Information Processing Systems*, 2019.

[49] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust Bayesian neural networks. *Advances in Neural Information Processing Systems*, 29:4134–4142, 2016.

[50] 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, 2018.

[51] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*, volume 82. 01 2000.

[52] Anne-Marie Lyne, Mark Girolami, Yves Atchadé, Heiko Strathmann, and Daniel Simpson. On Russian Roulette Estimates for Bayesian Inference with Doubly-Intractable Likelihoods. *Statistical Science*, 30(4):443 – 467, 2015.

[53] Yucen Luo, Alex Beatson, Mohammad Norouzi, Jun Zhu, David Duvenaud, Ryan P Adams, and Ricky TQ Chen. SUMO: Unbiased estimation of log marginal probability for latent variable models. In *International Conference on Learning Representations*, 2020.

[54] Jiaxin Shi, Shengyang Sun, and Jun Zhu. A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pages 4644–4653. PMLR, 2018.

[55] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[56] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, pages 370–378. PMLR, 2016.

[57] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005.

[58] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse gaussian process approximations. In *Advances in Neural Information Processing Systems*, 2016.

[59] L. C. W. Dixon and G. P. Szego. The Global Optimization Problem: An Introduction. In *Towards Global Optimisation 2*. North-Holland Pub. Co, 1978.

[60] Felix Berkenkamp, Anna Eivazi, Lukas Grossberger, Kathrin Skubch, Jonathan Spitz, Christian Daniel, and Stefan Falkner. Probabilistic Meta-Learning for Bayesian Optimization. `https://openreview.net/pdf?id=fdZvTFn8Yq`, 2021.

[61] Marcin Molga and Czesław Smutnicki. Test functions for optimization needs. *Test functions for optimization needs*, 101:48, 2005.

[62] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[63] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[64] Terry Therneau, Beth Atkinson, Brian Ripley, and Maintainer Brian Ripley. rpart: Recursive partitioning and regression trees, 2018.

[65] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[66] Daniel Kühn, Philipp Probst, Janek Thomas, and Bernd Bischl. Automatic exploration of machine learning experiments on openml. *arXiv preprint arXiv:1806.10961*, 2018.

[67] B. Bischl, Giuseppe Casalicchio, Matthias Feurer, F. Hutter, Michel Lang, R. Mantovani, J. N. Rijn, and J. Vanschoren. Openml benchmarking suites and the openml100. *arXiv preprint arXiv:1708.03731*, 2017.

[68] Christopher J Milne, Thomas Schietinger, Masamitsu Aiba, Arturo Alarcon, Jürgen Alex, Alexander Anghel, Vladimir Arsov, Carl Beard, Paul Beaud, Simona Bettoni, et al. Swissfel: the swiss x-ray free electron laser. *Applied Sciences*, 2017.

[69] Georg Ch Pflug. *Optimization of stochastic models: the interface between simulation and optimization*, volume 373. Springer Science & Business Media, 2012.

[70] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. 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., 2018.

[71] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[72] Jonas Rothfuss, Fabio Ferreira, Simon Boehm, Simon Walther, Maxim Ulrich, Tamim Asfour, and Andreas Krause. Noise Regularization for Conditional Density Estimation. *arXiv preprint arXiv:1907.08982*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our work introduces a new meta-learning approach which is very general and can be applied in many different scenarios. Our results suggest that is makes the uncertainty estimates of meta-learned models more reliable and robust which promises overall positive impact. However, misuse and potentially harmful applications of our method can never be ruled out.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix B.3.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section B.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A F-PACOH: Details and Additions

A.1 Estimating / Computing the Marginal Log-Likelihood

In case of GPs with Gaussian likelihood, the marginal log-likelihood $\ln Z(\mathcal{D}_{i,T_i}, P_\phi) = \ln p(\mathbf{y}_i^{\mathcal{D}} | \mathbf{X}_i^{\mathcal{D}}, \phi)$ can be computed in closed form as

$$\ln p(\mathbf{y}^{\mathcal{D}} | \mathbf{X}^{\mathcal{D}}, \phi) = -\frac{1}{2} \left(\mathbf{y}^{\mathcal{D}} - \mathbf{m}_{\mathbf{X}^{\mathcal{D}},\phi}\right)^\top \tilde{\mathbf{K}}_{\mathbf{X}^{\mathcal{D}},\phi}^{-1} \left(\mathbf{y}^{\mathcal{D}} - \mathbf{m}_{\mathbf{X}^{\mathcal{D}},\phi}\right) - \frac{1}{2} \ln |\tilde{K}_{\mathbf{X}^{\mathcal{D}},\phi}| - \frac{T}{2} \ln 2\pi \quad (5)$$

where $\tilde{\mathbf{K}}_{\mathbf{X}^{\mathcal{D}},\phi} = \mathbf{K}_{\mathbf{X}^{\mathcal{D}},\phi} + \sigma^2 I$, with kernel matrix $\mathbf{K}_{\mathbf{X}^{\mathcal{D}},\phi} = [k_\phi(\mathbf{x}_l, \mathbf{x}_k)]_{l,k=1}^{T_i}$, likelihood variance σ^2, and mean vector $\mathbf{m}_{\mathbf{X}^{\mathcal{D}},\phi} = [m_\phi(\mathbf{x}_1), ..., m_\phi(\mathbf{x}_{T_i})]^\top$. In most other cases, e.g. when the hypothesis space $\mathcal{H} = \{h_\theta, \theta \in \Theta\}$ corresponds to the parameters θ of a neural network, we need to form an approximation of the (generalized) marginal log-likelihood $\ln p(\mathbf{y}^{\mathcal{D}} | \mathbf{X}^{\mathcal{D}}, \phi) = \ln \mathbb{E}_{\theta \sim P_\phi} \left[e^{-\sum_{t=1}^{T_i} l(h_\theta(\mathbf{x}_{i,t}), y_{i,t})} \right]$. For a more detailed discussion on this matter, we refer to [12, 52, 53].

A.2 Gradients of the KL-divergence

For notational brevity, we assume a MAP approximation of the hyper-posterior so that the finite marginals of the hyper-posterior coincide with those of the prior P_ϕ, i.e., $q(\mathbf{h}^{\mathbf{X}_i}) = p_\phi(\mathbf{h}^{\mathbf{X}_i})$. However, the concepts discussed in the remainder straightforwardly apply to the case when $\mathcal{Q}$ is a full and non-Dirac posterior distribution. In the most general case, we only require that we can sample functions from our prior in a re-parametrizable manner, i.e., there exists a map φ and a noise distribution $p(\epsilon)$ such that for $\epsilon \sim p(\epsilon)$ we have $\mathbf{h}^{\mathbf{X}} = \varphi(\mathbf{X}, \phi, \epsilon) \sim p(\mathbf{h}^{\mathbf{X}})$. Following [21], we can derive the gradients of the KL-divergence as

$$\nabla_\phi KL[p(\mathbf{h}^{\mathbf{X}}) || \rho(\mathbf{h}^{\mathbf{X}})] = \underbrace{\mathbb{E}_{\mathbf{h}^{\mathbf{X}} \sim p_\phi} \left[\nabla_\phi \ln p_\phi(\mathbf{h}^{\mathbf{X}}) \right]}_{=0} + \mathbb{E}_\epsilon \left[\nabla_\phi \mathbf{h}^{\mathbf{X}} \left(\nabla_{\mathbf{h}} \ln p_\phi(\mathbf{h}^{\mathbf{X}}) - \nabla_{\mathbf{h}} \ln \rho(\mathbf{h}^{\mathbf{X}}) \right) \right]$$

Here, the expected score of p_ϕ is zero and $\nabla_\phi \mathbf{h}^{\mathbf{X}} = \nabla_\phi \varphi(\mathbf{X}, \phi, \epsilon)$ is the Jacobian of φ. Thus, it remains to estimate the score of the prior $\nabla_{\mathbf{h}} \ln p_\phi(\mathbf{h}^{\mathbf{X}})$ and the score of hyper-prior $\nabla_{\mathbf{h}} \ln \rho_\phi(\mathbf{h}^{\mathbf{X}})$. In various scenarios, the marginal distributions $p(\mathbf{h}^{\mathbf{X}})$ or $\rho(\mathbf{h}^{\mathbf{X}})$ in the function space may be intractable and we can only sample from it. For instance, when our prior $P_\phi(\theta)$ is a known distribution over neural network (NN) parameters θ, associated with NN functions $h_\theta : \mathcal{X} \mapsto \mathcal{Y}$, its marginals $p_\phi(\mathbf{h}^{\mathbf{X}})$ in the function space are typically intractable since the NN map is not injective w.r.t. to θ. In such cases, we can use either the Spectral Stein Gradient Estimator (SSGE) [54] or sliced score matching [55] to estimate the respective score from samples. Whenever the marginal densities are available in closed form, we use automatic differentiation to compute their score. Finally, if both the prior and the hyper-prior are GPs, we use the closed-form KL-divergence between multivariate normal distributions.

A.3 F-PACOH-VI

While we mainly focus on a maximum a-posteriori approximation of the hyper-posterior $\mathcal{Q}$ in the main paper (see Section 4.3), we now discuss the more general case of a variational approximation of the hyper-posterior.

IN this case, we presume a parametric variational family $\{\hat{\mathcal{Q}}_\xi(\phi), \xi \in \Xi\}$ of hyper-posterior distributions that are supported on the parameter space Φ of the prior. Moreover, we require that we can sample ϕ in a re-parameterizable manner, i.e., there exists a differentiable function g and a noise distribution $p(\epsilon)$ such that $g(\xi, \epsilon) \sim \hat{\mathcal{Q}}_\xi$ for $\epsilon \sim p(\epsilon)$. An example of such variational family are Gaussians $\mathcal{N}(\phi; \mu_{\mathcal{Q}}, \text{diag}(\sigma_{\mathcal{Q}}^2))$ with diagonal covariance matrices such that the variational parameters $\xi = (\mu_{\mathcal{Q}}, \sigma_{\mathcal{Q}}^2)$ coincide with the mean and the variance of the distribution. We can obtain re-parametrized samples with $\phi = g(\xi, \epsilon) = \mu_{\mathcal{Q}} + \sigma_{\mathcal{Q}} \epsilon$ and $p(\epsilon) = \mathcal{N}(0, I)$. Overall, re-parametrizable sampling allows us to obtain low-variance pathwise stochastic gradient estimators [69] of the expectation $\nabla_\xi \mathbb{E}_{\phi \sim \mathcal{Q}_\xi}[\ln Z(\mathcal{D}_{i,T_i}, P_\phi)]$ in (3).

Algorithm 2 F-PACOH-VI: Meta-Training with GP priors

Input: Datasets $\mathcal{D}_{1,T_1}, ..., \mathcal{D}_{n,T_n}$, parametric family $\{P_\phi | \phi \in \Phi\}$ of GP priors, learning rate α
Input: Parametric family of hyper-posteriors $\{\hat{\mathcal{Q}}_\xi(\phi), \xi \in \Xi\}$
Input: Stochastic process hyper-prior with marginals $\rho(\cdot)$
Initialize the parameters ξ of the hyper-posterior $\mathcal{Q}_\xi(\phi)$
while not converged **do**
  for $i = 1, ..., n$ **do** ▷ Iterate over meta-training tasks
    $\mathbf{X}_i = [\mathbf{X}_{i,s}^{\mathcal{D}}, \mathbf{X}_i^M]$, where $\mathbf{X}_{i,s}^{\mathcal{D}} \subseteq \mathbf{X}_i^{\mathcal{D}}, \mathbf{X}_i^M \overset{iid}{\sim} \mathcal{U}(\mathcal{X})$ ▷ Sample measurement set
    $\phi \leftarrow g(\xi, \epsilon_{\mathcal{Q}})$ with $\epsilon_{\mathcal{Q}} \sim p(\epsilon_{\mathcal{Q}})$ ▷ Sample prior parameters from $\hat{\mathcal{Q}}_\xi$
    Compute $\nabla_\phi \ln Z(\mathbf{X}_i^{\mathcal{D}}, P_\phi)$ in closed form ▷ cf. (5)
    $\nabla_{\mathbf{h}} \ln q_\phi(\mathbf{h}^{\mathbf{X}}) \leftarrow \text{SSGE}(q_\phi, \mathbf{h}^{\mathbf{X}})$ ▷ estimate hyper-posterior score
    $\mathbf{h}^{\mathbf{X}_i} \leftarrow \mathbf{m}_{\mathbf{X}_i,\phi} + \mathbf{K}_{\mathbf{X}_i,\phi}^{\frac{1}{2}} \epsilon_P$ with $\epsilon_P \sim \mathcal{N}(0, I)$ ▷ Sample function values from GP prior
    $\nabla_\xi KL[q_\xi(\mathbf{h}^{\mathbf{X}_i}) || \rho(\mathbf{h}^{\mathbf{X}_i})] \leftarrow \nabla_\xi \mathbf{h}^{\mathbf{X}_i} \left(\nabla_{\mathbf{h}} \ln q_\phi(\mathbf{h}^{\mathbf{X}_i}) - \nabla_{\mathbf{h}} \ln \rho(\mathbf{h}^{\mathbf{X}_i}) \right)$
    $\nabla_\xi J_{F,i} = \frac{1}{T_i} \nabla_\xi g(\xi, \epsilon_{\mathcal{Q}}) \nabla_\phi \ln Z(\mathcal{D}_{i,T_i}, P_\phi) + \left(\frac{1}{\sqrt{n}} + \frac{1}{nT_i} \right) \nabla_\xi KL[q(\mathbf{h}^{\mathbf{X}_i}) || \rho(\mathbf{h}^{\mathbf{X}_i})]$
  end for
  $\xi \leftarrow \xi - \alpha \frac{1}{n} \sum_{i=1}^n \nabla_\xi J_{F,i}$ ▷ Update hyper-posterior parameter
end while

A bigger challenge becomes estimating the gradient of the functional KL divergence w.r.t. the hyper-posterior parameters ξ, i.e.,

$$\nabla_\xi \mathbb{E}_{\mathbf{X}} \left[KL[q_\xi(\mathbf{h}^{\mathbf{X}}) || \rho(\mathbf{h}^{\mathbf{X}})] \right] = \mathbb{E}_{\mathbf{X}} \left[\nabla_\xi \mathbb{E}_{\mathbf{h}^{\mathbf{X}} \sim q_\xi} \left[\ln q_\xi(\mathbf{h}^{\mathbf{X}}) - \ln \rho(\mathbf{h}^{\mathbf{X}}) \right] \right] \tag{6}$$

$$= \mathbb{E}_{\mathbf{X}} \left[\nabla_\xi \underbrace{\mathbb{E}_{\mathbf{h}^{\mathbf{X}} \sim q_\xi} \left[\ln q_\xi(\mathbf{h}^{\mathbf{X}}) \right]}_{-\text{ entropy}} - \nabla_\xi \underbrace{\mathbb{E}_{\mathbf{h}^{\mathbf{X}} \sim q_\xi} \left[\ln \rho(\mathbf{h}^{\mathbf{X}}) \right]}_{-\text{ cross-entropy}} \right] \tag{7}$$

In particular, since we now have a full distribution over priors, the hyper-posterior marginals

$$q(\mathbf{h}^{\mathbf{X}}) = \mathbb{E}_{\phi \sim \hat{\mathcal{Q}}_\xi} \left[p_\phi(\mathbf{h}^{\mathbf{X}}) \right] \tag{8}$$

are generally intractable mixing distributions, even if the prior is a GP and its marginals are tractable multivariate normal distributions. While we can still get an unbiased pathwise gradient estimator of the cross-entropy, a simple Monte Carlo gradient estimate of the negative entropy will be biased due to the concavity of the logarithm outside the expectation in $\nabla_\xi \ln q_\xi(\mathbf{h}^{\mathbf{X}}) = \nabla_\xi \ln \mathbb{E}_{\phi \sim \mathcal{Q}_\xi} \left[p_\phi(\mathbf{h}^{\mathbf{X}}) \right]$ [53].

As discussed in Section 4.2, we may resort to score estimation techniques such as [54, 55] to obtain an estimate of KL-divergence. In particular, we rewrite the gradient of the KL-divergence as

$$\nabla_\xi KL[q_\xi(\mathbf{h}^{\mathbf{X}}) || \rho(\mathbf{h}^{\mathbf{X}})] = \mathbb{E}_{\epsilon_{\mathcal{Q}}, \epsilon_P} \left[\nabla_\xi \mathbf{h}^{\mathbf{X}} \left(\nabla_{\mathbf{h}} \ln q_\phi(\mathbf{h}^{\mathbf{X}}) - \nabla_{\mathbf{h}} \ln \rho(\mathbf{h}^{\mathbf{X}}) \right) \right] , \tag{9}$$

wherein $\nabla_\xi \mathbf{h}^{\mathbf{X}} = \nabla_\xi \varphi(g(\xi, \epsilon_{\mathcal{Q}}), \epsilon_P)$ is the Jacobian for the concatenation $g \circ \varphi$ of the re-parametrization maps for the prior and the hyper-posterior. Finally, we propose to use SSGE [54] for obtaining a sampling based estimate of $\nabla_{\mathbf{h}} \ln q_\phi(\mathbf{h}^{\mathbf{X}})$. If we use a GP as hyper-prior, $\nabla_{\mathbf{h}} \ln \rho(\mathbf{h}^{\mathbf{X}})$ is the score of a multivariate normal distribution and thus available in closed form. Algorithm 2 summarizes the resulting meta-training procedure for GP priors.

A.4 Implementation Details for F-PACOH-MAP with GPs

In the following, we provide details on our implementation of the the F-PACOH-MAP algorithm which has been introduced in Section 4.3.

The NN-based GP prior Following [38, 56], we parameterize the GP prior $P_\phi(h) = \mathcal{GP}(h | m_\phi(\mathbf{x}), k_\phi(\mathbf{x}, \mathbf{x}'))$, particularly the mean m_ϕ and kernel function k_ϕ, as neural networks (NN). Here, the parameter vector ϕ corresponds to the weights and biases of the NN. To ensure the positive-definiteness of the kernel, we use the neural network as feature map $\Phi_\phi(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}^d$ that

Algorithm 3 F-PACOH-MAP: Meta-Learning GP Priors

Input: Datasets $\mathcal{D}_{1,T_1}, ..., \mathcal{D}_{n,T_n}$, parametric family $\{P_\phi | \phi \in \Phi\}$ of GP priors, learning rate α
Input: GP hyper-prior with marginals $\rho(\mathbf{h}^{\mathbf{X}}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{X},\mathcal{P}})$
Initialize the parameters ϕ of the GP prior $P_\phi(h) = \mathcal{GP}\left(h | m_\phi(\mathbf{x}), k_\phi(\mathbf{x}, \mathbf{x}')\right)$
while not converged **do**
  Sample batch $I_{batch} \subseteq \{1, ..., n\}$ of H tasks
  for $i \in I_{batch}$ **do** ▷ Iterate over meta-training tasks
    $\mathbf{X}_i = [\mathbf{X}_{i,s}^{\mathcal{D}}, \mathbf{X}_i^M]$, where $\mathbf{X}_{i,s}^{\mathcal{D}} \subseteq \mathbf{X}_i^{\mathcal{D}}, \mathbf{X}_i^M \overset{iid}{\sim} \mathcal{U}(\mathcal{X})$ ▷ Sample measurement set
    $\ln Z_{i,\phi} \leftarrow -\frac{1}{2}\left(\mathbf{y}_i^{\mathcal{D}} - \mathbf{m}_{\mathbf{X}_i^{\mathcal{D}},\phi}\right)^\top \tilde{\mathbf{K}}_{\mathbf{X}_i^{\mathcal{D}},\phi}^{-1}\left(\mathbf{y}_i^{\mathcal{D}} - \mathbf{m}_{\mathbf{X}_i^{\mathcal{D}},\phi}\right) - \frac{1}{2}\ln|\tilde{K}_{\mathbf{X}_i^{\mathcal{D}},\phi}| - \frac{T}{2}\ln 2\pi$
    $KL_{i,\phi} \leftarrow \frac{1}{2}\left(\text{tr}\left(\mathbf{K}_{\mathbf{X}_i,\mathcal{P}}^{-1}\mathbf{K}_{\mathbf{X}_i,\phi}\right) + \mathbf{m}_{\mathbf{X}_i,\phi}^\top \mathbf{K}_{\mathbf{X}_i,\mathcal{P}}^{-1}\mathbf{m}_{\mathbf{X}_i,\phi} - L + \ln\frac{|\mathbf{K}_{\mathbf{X}_i,\mathcal{P}}|}{|\mathbf{K}_{\mathbf{X}_i,\phi}|}\right)$
    $\nabla_\phi J_{F,i} = -\frac{1}{T_i}\nabla_\phi \ln Z(\mathcal{D}_{i,T_i}, P_\phi) + \left(\frac{\kappa}{\sqrt{n}} + \frac{\kappa}{nT_i}\right)\nabla_\phi KL_{i,\phi}$
  end for
  $\phi \leftarrow \texttt{AdamOptimizer}(\phi, \alpha, \frac{1}{H}\sum_{i \in I_{batch}} \nabla_\phi J_{F,i})$ ▷ Update prior parameter
end while
Output: Meta-learned GP prior $P_\phi(h)$

maps to a d-dimensional real-values feature space in which we apply a squared exponential kernel. Accordingly, the parametric kernel reads as

$$k_\phi(x, x') = \nu_P \exp\left(-||\Phi_\phi(\mathbf{x}) - \Phi_\phi(\mathbf{x}')||^2/(2l_P)\right) \ . \tag{10}$$

Both $m_\phi(\mathbf{x})$ and $\Phi_\phi(\mathbf{x})$ are fully-connected neural networks with 3 layers with each 32 neurons and $\tanh$ non-linearities. The kernel variance ν_P and lengthscale l_P as well as the Gaussian likelihood variance σ_P^2 $p(y|h(\mathbf{x})) = \mathcal{N}(y; h(\mathbf{x}), \sigma_P^2)$ are also learnable parameters which are appended to the NN parameters ϕ. Since l_P and σ_P^2 need to be positive, we represent and optimize them in log-space.

The hyper-prior We use a Vanilla GP $\mathcal{GP}(0, k_{\mathcal{P}}(x, x'))$ as hyper-prior stochastic process. In that,

$$k_{\mathcal{P}}(x, x') = \nu_{\mathcal{P}} \exp\left(-||x - x'||^2/(2l_{\mathcal{P}})\right) \tag{11}$$

is a SE kernel with variance $\nu_{\mathcal{P}}$ and lengthscale $l_{\mathcal{P}}$. Both are treated as hyper-parameters. Correspondingly, the finite marginals of the hyper-prior $\rho(\mathbf{h}^{\mathbf{X}}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{X},\mathcal{P}})$ are multivariate normal distributions.

Minimizing the functional meta-learning objective In case of the MAP approximation, we aim to minimize the functional meta-learning objective in (12) directly w.r.t. the prior parameters ϕ. To minimize the objective we use mini-batching on the task level, i.e., in each iteration we sample a random subset $I_{batch} \subset \{1, ..., n\}$ with $H = |I_{batch}|$ task indices and only compute the average over the mini-batch of tasks. This stochastic estimate is unbiased and much faster to optimize than the entire sum over tasks. Since the weighting term $(1/\sqrt{n} + 1/(nT_i))$ in front of the KL divergence originates from conservative worst-case bounds on transfer error, it may be sub-optimal in expectation. Thus, following [23, 12] we add a scalar weight $\kappa > 0$ in front of it and treat it as hyper-parameter. As described in Algorithm 1, we need to also sample random measurement sets $\mathbf{X}_i$ in each iteration and for each of the tasks in the batch. In particular, we sample 10 random points $\mathbf{X}_{i,s}^{\mathcal{D}}$ without replacement from the inputs $\mathbf{X}_i^{\mathcal{D}}$ corresponding to task i and sample another 10 points $\mathbf{X}_i^M \overset{iid}{\sim} \mathcal{U}(\mathcal{X})$ uniformly and independently from the bounded domain $\mathcal{X}$. The final measurement set $\mathbf{X}_i = [\mathbf{X}_{i,s}^{\mathcal{D}}, \mathbf{X}_i^M]$ is the concatenation of both sets and thus contains $L = 20$ points. Then we compute the sample-based objective

$$J_F^{MAP}(\phi) = -\frac{1}{H}\sum_{i \in I_{batch}}\left(\frac{1}{T_i}\ln Z(\mathcal{D}_{i,T_i}, P_\phi) + \left(\frac{\kappa}{\sqrt{n}} + \frac{\kappa}{nT_i}\right) KL[q(\mathbf{h}^{\mathbf{X}_i})||\rho(\mathbf{h}^{\mathbf{X}_i})]\right) \tag{12}$$

wherein the marginal log-likelihood (see A.1) and the KL-divergence $KL[p_\theta(\mathbf{h}^{\mathbf{X}})|\rho(\mathbf{h}^{\mathbf{X}})]$ are available in closed form. In particular, we use GPyTorch [70] to perfom these computations numerically stable and automatic differentiation to compute the gradients $\nabla_\phi J_F^{MAP}(\phi)$. To perform gradient updates to ϕ, we use the adaptive learning rate method AdamW [71] with learning rate α and weight

$\mathcal{T}$	$dim(\mathcal{X})$	n	T_i
Random Mixture 1d	1	10	10
Random Branin	2	20	20
Camelback Sin-Noise	2	20	20
Random Hartmann6	6	30	100
GLMNET	2	20	10
RPart	4	20	20
XGBoost	10	20	50

Table 1: Summary of meta-BO benchmark environments

decay ω. In addition, we decay the learning rate every 1000 iterations by a factor $\eta \in (0, 1)$. Both α, ω and eta as well as the overall number of iterations are treated are hyper-parameters. Algorithm 3 summarizes the F-PACOH-MAP meta-learning procedure for GPs.

B Experiment Details and Further Results

B.1 Benchmark Environments

In the following, we provide further details on benchmark environments that were used in the experiments in Section 5. Table 1 displays a summary of the environments, specifying the dimensionality of the domain, the number of meta-training tasks n and the number of evaluation points T_i per task used in the experiments of Section 5.1 and Section 5.3.

B.1.1 Simulated Benchmarks

Random Mixture Environment (1D) The environment corresponds to an affine combination of un-normalized Cauchy and Gaussian probability density functions:

$$p_1(x) = \frac{1}{\pi(1 + ||x - \mu_1||^2)}, \quad p_2(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{||x-\mu_2||^2}{8}}, \quad p_3(x) = \frac{1}{\pi\left(1 + \frac{||x-\mu_3||^2}{4}\right)} . \tag{13}$$

The target function follows as

$$f(x) = 2 \cdot w_1 \cdot p_1(x) + 1.5 \cdot w_2 \cdot p_2(x) + 1.8 \cdot w_3 \cdot p_2(x) + 1 \tag{14}$$

wherein the mixing weights w_1, w_2, w_3 are sampled independently from $\mathcal{U}(0.6, 1.4)$ and the location parameters are sampled from the Gaussians

$$\mu_1 \sim \mathcal{N}(-2, 0.3^2), \quad \mu \sim \mathcal{N}(3, 0.3^2), \quad \mu_3 \sim \mathcal{N}(-8, 0.3^2) . \tag{15}$$

The domain is the one dimensional interval $\mathcal{X} = [-10, 10]^\top$. Function samples from the environment are illustrated in Fig. 1.

Random Branin The environment corresponds to random Branin functions [59] with the 2-dimensional cube $\mathcal{X} = [-5, 10] \times [0, 15]$ as domain. We denote $\mathbf{x} = (x_1, x_2)^\top$. Since we phrase BO as maximization problem, we used the negative Branin function:

$$f(x_1, x_2) = -\left(a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t)\cos(x_1) + s\right) \tag{16}$$

In that, the parameters a, b, c, r, s, t are sampled from uniform distributions, in particular,

$$\begin{aligned} a \sim \mathcal{U}(0.5, 1.5), \quad b \sim \mathcal{U}(0.1, 0.15), \quad c \sim \mathcal{U}(1, 2), \\ r \sim \mathcal{U}(5, 7), \quad s \sim \mathcal{U}(8, 12), \quad t \sim \mathcal{U}(0.03, 0.05) . \end{aligned} \tag{17}$$

Camelback Sin-Noise The environment corresponds to a Camelback function [61]

$$g(x_1, x_2) = \max\left(-(4 - 2.1 \cdot x_1^2 + x_1^4/3) * x_1^2 - x_1 x_2 - (4 \cdot x_2^2 - 4) * x_2^2, \ -2.5\right) . \tag{18}$$

plus random sinusoid functions, defined over the 2-dimensional cube $\mathcal{X} = [-2, 2] \times [-1, 2]$ as domain. Specifically, the target function is defined as

$$f(x_1, x_2) = g(x_1, x_2) + a \sin(\omega_1 * (x_1 - \rho_1)) \sin(\omega_2 * (x_2 - \rho_2)) \tag{19}$$

wherein the parameters are sampled independently as

$$a \sim \mathcal{U}(0.3, 0.5), \quad \omega_1, \omega_2 \sim \mathcal{U}(0.5, 1.0), \quad \rho_1, \rho_2 \sim \mathcal{N}(0, 0.3^2) . \tag{20}$$

Random Hartmann6 The environment corresponds to a negated and randomized version Hartmann-6D function [59] with the hyper-cube $\mathcal{X} = [0, 1]^6$ as domain. In particular, the target function is defined as

$$f(\mathbf{x}) = \frac{1}{3.322368} \sum_{i=1}^{4} \alpha_i \exp\left(- \sum_{j=1}^{6} A_{i,j}(x_j - P_{i,j})^2 \right) \text{, where} \tag{21}$$

$$\mathbf{A} = \begin{pmatrix} 10.00 & 3.00 & 17.00 & 3.50 & 1.70 & 8.00 \\ 0.05 & 10.00 & 17.00 & 0.10 & 8.00 & 14.00 \\ 3.00 & 3.50 & 1.70 & 10.00 & 17.00 & 8.00 \\ 17.00 & 8.00 & 0.05 & 10.00 & 0.10 & 14.00 \end{pmatrix} \text{ and} \tag{22}$$

$$\mathbf{P} = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix} . \tag{23}$$

The parameters $\alpha_1, ..., \alpha_4$ are sampled independently from uniform distributions

$$\alpha_1 \sim \mathcal{U}(0.5, 1.5), \quad \alpha_2 \sim \mathcal{U}(0.6, 1.4), \quad \alpha_3 \sim \mathcal{U}(2.0, 3.0), \quad \alpha_4 \sim \mathcal{U}(2.8, 3.6). \tag{24}$$

B.1.2 Hyper-Parameter Tuning on OpenML Datasets

In our BO empirical benchmark studies, we consider use case of hyper-parameter tuning for machine learning algorithm. In particular, we consider three machine learning algorithms for this purpose:

- Generalized linear models with elastic NET regularization *(GLMNET)* [62]
- Recursively partitioning trees *(RPart)* [63, 64]
- Gradient boosting *(XGBoost)* [65]

Following previous work [e.g. 50, 60], we replace the costly training and evaluation step by a cheap table lookup based on a large number of hyper-parameter evaluations [66] on 38 classification datasets from the OpenML platform [67]. The hyper-parameter evaluations are available under a Creative Commons BY 4.0 license and can be downloaded here[3]. In effect, $\mathcal{X}$ is a finite set, corresponding to 10000-30000 random evaluations hyper-parameter evaluations per dataset and machine learning algorithm. Since the sampling is quite dense, for the purpose of empirically evaluating the meta-learned models towards BO, this finite domain can be treated like a continuous domain. All datasets correspond to binary classification. The target function we aim to optimize is the area under the ROC curve (AUROC) on a test split of the respective dataset.

We randomly split the available tasks (i.e. train/test evaluations on a specific dataset) into a set of meta-train and meta-test tasks. In the following, we list the corresponding OpenML dataset identifiers:

- meta-train tasks: 3, 1036, 1038, 1043, 1046, 151, 1176, 1049, 1050, 31, 1570, 37, 4134, 1063, 1067, 44, 1068, 50, 1461, 1462
- meta-test tasks: 335, 1489, 1486, 1494, 1504, 1120, 1510, 1479, 1480, 333, 1485, 1487, 334

Since some of machine learning algorithm's hyper-parameters were sampled by [66] in log-space, we transform the respective hyper-parameters accordingly and also adjust them to standard normal values ranges such that we can expect a reasonably good performance of a Vanilla GP with SE kernel. The hyper-parameters and transformations are listed in Table 2.

B.2 Evaluation Methodology and Metrics

B.2.1 Supervised Learning: Regression

Methodology In the following, we describe our experimental methodology and provide details on how the empirical results reported in Table 4 and Table 5 were generated. Overall, evaluating a

[3] `https://doi.org/10.6084/m9.figshare.5882230.v2`

algorithm	hyper-parameter	type	transformation
GLMNET	alpha	numeric	-
	lambda	numeric	$t(x) = \log_2(x)/10$
RPart	cp	numeric	$t(x) = 4x$
	maxdepth	integer	$t(x) = x/10$
	minbucket	integer	$t(x) = x/20$
	minsplit	integer	$t(x) = x/20$
XGBoost	nrounds	integer	$t(x) = (x - 2000)/1000$
	eta	numeric	$t(x) = (\log_2(x) + 5)/2$
	lambda	numeric	$t(x) = \log_2(x)/5$
	alpha	numeric	$t(x) = \log_2(x)/5$
	subsample	numeric	$t(x) = (x - 0.5)/2$
	booster	$\{-1, 1\}$	-1 for 'linear' and 1 for 'tree'
	max_depth	integer	-
	min_child_weight	numeric	$t(x) = (x - 50)/20$
	colsample_bytree	numeric	-
	colsample_bylevel	numeric	-

Table 2: Hyper-parameters and corresponding parameter transformations for the three machine learning algorithms considered for our hyper-parameter tuning benchmark

meta-learner consists of two phases, *meta-training* and *meta-testing*. In meta-training, we perform meta=learning based on a set of datasets $\{\mathcal{D}_1, ..., \mathcal{D}_n\}$ corresponding to tasks sampled from the task distribution $\mathcal{T}$. In meta-learned model receives multiple of unseen test tasks consisting of each a train set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$ that both correspond to the same function $f \sim \mathcal{T}$. The train set $\mathcal{D}_{train}$ is used to perform inference / training with the model. Then the following evaluation metrics are computed on $\mathcal{D}_{test}$.

Log-Likelihood Following [72], we report the average predictive log-likelihood of test points:

$$\text{LL} = \frac{1}{|\mathcal{D}_{test}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{test}} \ln \hat{p}(y|\mathbf{x}, \mathcal{D}_{train}) \tag{25}$$

In that, $\hat{p}(\cdot|\mathbf{x})$ denotes is the predictive distribution of the respective (meta-learned) model, trained on $\mathcal{D}_{train}$ at meta-test time.

Calibration Error The concept of calibration applies to probabilistic predictors that, given a new target input $\mathbf{x}_j$, produce a probability distribution $\hat{p}(y_j|\mathbf{x}_j)$ over predicted target values y_j [19, 20].

Corresponding to the predictive density, we denote a predictor's cumulative density function (CDF) as $\hat{F}(y_j|\mathbf{x}_j) = \int_{-\infty}^{y_j} \hat{p}(y|\mathbf{x}_i)dy$. For confidence levels $0 \leq q_h < ... < q_H \leq 1$, we can compute the corresponding empirical frequency

$$\hat{q}_h = \frac{|\{y_j \mid \hat{F}(y_j|\mathbf{x}_j) \leq q_h, j = 1, ..., m\}|}{m} , \tag{26}$$

based on the test dataset $\mathcal{D}_{test} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ of m samples. If we have calibrated predictions we would expect that $\hat{q}_h \to q_h$ as $m \to \infty$. Similar to [20], we can define the calibration error as a function of residuals $\hat{q}_h - q_h$, in particular,

$$\text{calib-err} = \sqrt{\frac{1}{H} \sum_{h=1}^{H} (\hat{q}_h - q_h)^2} . \tag{27}$$

Note that we while [20] reports the average of squared residuals $|\hat{q}_h - q_h|^2$, we report its square root in order to preserve the units and keep the calibration error easier to interpret. In our experiments, we compute (27) with $M = 20$ equally spaced confidence levels between 0 and 1.

B.2.2 Offline Meta-Learning for BO

In this section, we describe the experimental methodology of the meta-learning for BO experiments in Section 5.3.

Unlike previous work [e.g. 50, 60] which collects meta-training data by uniformly sampling data from the domain, we collect meta-training data by running Vanilla GP-UCB on a the respective meta-training tasks. This is a more realistic setup since in real-wold applications we most likely only have access to non-i.i.d. data that originates from previous optimization attempts. The number of tasks n and evaluations per task T_i are specified in Table 1. In case of the simulated experiments, the tasks are sampled i.i.d. from the task distribution while in hyper-parameter optimization study they corresponds to the meta-train tasks listed in Appendix B.1.2.

With the collected datasets, we perform meta-training and then employ the meta-learned model towards BO with the UCB acquisition function

$$\alpha_t(\mathbf{x}) = \hat{\mu}_{t-1}(\mathbf{x}) + 2\hat{\sigma}_{t-1}(\mathbf{x}) \tag{28}$$

on unseen meta-test tasks, i.e. new functions $f \sim \mathcal{T}$. In that, $\hat{\mu}_{t-1}(\mathbf{x})$ and $\hat{\sigma}_{t-1}(\mathbf{x})$ denote the mean and standard deviation of predictive distribution $\hat{p}(y|\mathbf{x}, \{(\mathbf{x}_{t'}, y_{t'}\}_{t'=1}^{t-1})$ of the meta-learned model, given the previous BO evaluations $\{(\mathbf{x}_{t'}, y_{t'}\}_{t'=1}^{t-1}$ in this run as training data. To obtain statistically robust results, we evaluate the BO performance on 10 test tasks and repeat the entire meta-training and BO process for 25 model seeds.

To assess the BO performance, we report the *simple regret*

$$r_{f,t} = f(\mathbf{x}^*) - \max_{t' \leq t} f(\mathbf{x}_{t'}) \ , \tag{29}$$

wherein $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ is the global optimum of the target function, $\mathbf{x}_t$ the point the BO algorithm chooses to evaluate in iteration t. Moreover, we report the *inference regret*

$$\hat{r}_{f,t} = f(\mathbf{x}^*) - f(\hat{\mathbf{x}}_t^*) \ , \tag{30}$$

where $\hat{\mathbf{x}}_t^* = \arg\max_{\mathbf{x} \in \mathcal{X}} \hat{\mu}_{t-1}(\mathbf{x})$ the predicted maximum at time t.

B.2.3 Lifelong BO

Unlike in the offline meta-learning setting, in the lifelong BO experiment of Section 5.4, $n = 10$ BO runs with $T_i = 20$ steps each are performed sequentially and meta-training happens online fashion after every BO run. Since, initially, there is no meta-training data available, i.e. $\mathcal{M}_0 = \emptyset$, we use a Vanilla GP as model in the first BO run. After each run, we add the collected function evaluations to the meta-training data, i.e., $\mathcal{M}_{i+1} = \mathcal{M}_t \cup \{D_{i,T}\}$. For the following runs ($i > 0$), we first perform meta-training with $\mathcal{M}_i$ and then run BO with the UCB acquisition function and the meta-learned model. In case of the simulated environments, the 10 tasks are sampled i.i.d from $\mathcal{T}$, whereas in the hyper-parameter tuning setting, we use randomly shuffled sequences of tasks from the meta-test tasks listed in Appendix B.1.2. We repeat the whole lifelong BO process for 5 random task sequences and and 5 model seeds each.

To assess the overall performance of the meta-learned to the end of lifelong BO, we compute the *cumulative inference regret*

$$R_{i,t} = \sum_{i'<i} \sum_{t'=0}^{T_{i'}} \hat{r}_{f_{i'},t'} + \sum_{t'=0}^{t} \hat{r}_{f_i,t'} \ , \tag{31}$$

that is, is the sum of inference regrets of all the previous steps and runs. In addition, we report the *simple regret $r_{f_i,20}$ at the end of each BO run*. While the former metric gives us a good picture of the respective meta-learner throughout the course of the lifelong Bayesian Optimization, the latter metric tells us what is the best point found per run and how this end-of-run solution develops as the meta-learned collects more meta-training tasks.

B.3 Open Source Code, Experiment Data and Compute Resources

We provide source code which includes an implementation of F-PACOH, the baselines, the environments as well as the experiment scripts to reproduce the presented empirical results. The source code is part of our code and data repository which can be accessed via:

https://www.dropbox.com/sh/n2thesjq87sh66j/AACg-HKMl1NhQpaMOHvvUEOfa?dl=0

The repository also includes the meta-training data which has been collected with GP-UCB as well as detailed recordings of our experiments that were the basis for the plots and tables presented in this paper.

All experiments for this work, especially the hyper-parameter sweeps for F-PACOH and the baselines were conduced on CPU-only machines on Oracle Cloud. Overall, we have used 192,428 OCPU hours, an equivalent of 125 days on a 64-core machine.

B.4 Hyper-Parameter Selection for F-PACOH-MAP and the baselines

	symbol	sampling type	value range / choices
learning rate	α	loguniform	[0.0001, 0.005]
learning rate decay	η	loguniform	[0.8, 1.0]
weight decay	ω	loguniform	[0.00001, 0.1]
task batch size	H	choice	{4, 10}
number of meta-training iterations	-	choice	{2000, 4000, 8000}
hyper-prior lengthscale	$l_{\mathcal{P}}$	loguniform	[0.1, 1.0]
hyper-prior factor	κ	loguniform	[0.0001, 0.5]
kernel feature dimensionality	d	choice	{2, 6}

Table 3: Hyper-parameter search ranges and uniform sampling types for F-PACOH-MAP

The choose the hyper-parameters of F-PACOH-MAP and the considered baselines we use random search. The hyper-parameters are either sampled uniformly from a finite set of choices or sampled log-uniformly over a range. The particular choice sets and ranges for F-PACOH-MAP are listed in Table 3. We draw 128 random hyper-parameter samples, employ the respective method on a specific environment and select the hyper-parameters corresponding to the best hyper-parameter settings employed on three validation tasks that are distinct from the meta-test tasks. Specifically, for the offline meta-learning experiment, we select the best hyper-parameters based on the last simple regret $r_{f,T}$ and the average inference regret during that last 50 iterations $\frac{1}{50}\sum_{t=T-50}^{T} \hat{r}_{f,T}$. In the life-long BO setting, we use the average inference regret over the last 5 steps per run as well as the last simple regret per run, averaged over all runs, as performance metrics. In particular, we rank the 128 hyper-parameter runs for each of the two metrics and choose the hyper-parameter setting with the highest average ranking. We perform this random hyper-parameter search for all the baselines and all the environments individually.

The resulting hyper-parameter configurations for all the methods and environments are reported in our experiment repository and can be accessed / downloaded under the following link: https://www.dropbox.com/sh/n2thesjq87sh66j/AACg-HKMl1NhQpaMOHvvUEOfa?dl=0

B.5 Further Experiment Results

B.5.1 Supervised Meta-Learning Experiments

Following the methodology described in Appendix B.2.1, we present a meta-learning benchmark study that evaluates the F-PACOH method as well as multiple other baselines based on their supervised learning predictions. Unlike the empirical studies in Section 5.3 and 5.4 which evaluate the BO performance of the meta-learned, this study only considers regression. We use the task data which was collected using GP-UCB as part of the experiment in Section 5.3 for meta-training and meta-testing. See Table 1 for a summary of the number of tasks and data points per environment. In particular, we use half of the n tasks for meta-training and the other half for meta-testing. For the meta-test tasks, we use 50% of the points for inference and the other 50% for computing the test metrics.

Table 4 reports the average test log-likelihood and Table 5 lists the corresponding calibration error. Overall, we observe that, alongside ABLR, F-PACOH achieves the best test log-likelihood across the environments. In terms of the calibrations of its uncertainty estimates, F-PACOH significantly outperforms the other methods in the majority of the environments. This is consistent with our

	Rand. Branin	Camelb. Sin-Noise	Rand. Hartmann6	GLMNET	RPart	XGBoost
FPACOH-MAP	$\mathbf{-1.854 \pm 0.015}$	-0.235 ± 0.044	$\mathbf{1.448 \pm 0.044}$	$\mathbf{1.692 \pm 0.041}$	1.596 ± 0.087	1.051 ± 0.079
PACOH-MAP	-2.507 ± 0.267	-0.716 ± 0.029	1.337 ± 0.048	1.369 ± 0.025	-0.808 ± 0.217	0.916 ± 0.027
ABLR	-3.684 ± 0.006	-0.738 ± 0.008	1.358 ± 0.059	1.233 ± 0.012	-0.471 ± 0.209	0.949 ± 0.061
NP	-4.621 ± 0.232	-0.888 ± 0.031	1.288 ± 0.099	0.875 ± 0.523	-0.566 ± 0.496	0.421 ± 0.144
Learned GP	-3.738 ± 0.003	$\mathbf{0.395 \pm 0.004}$	1.305 ± 0.001	1.412 ± 0.002	$\mathbf{1.611 \pm 0.002}$	$\mathbf{1.587 \pm 0.004}$
Vanilla GP	-3.027 ± 0.000	0.170 ± 0.000	1.351 ± 0.041	0.831 ± 0.000	1.380 ± 0.000	0.872 ± 0.000

Table 4: Average test log-likelihood of various meta-learned models as well as a Vanilla GP on meta-test tasks generated by uniform sampling from the meta-BO benchmark environments.

	Rand. Branin	Camelb. Sin-Noise	Rand. Hartmann6	GLMNET	RPart	XGBoost
FPACOH-MAP	$\mathbf{0.095 \pm 0.006}$	$\mathbf{0.046 \pm 0.002}$	0.049 ± 0.003	0.124 ± 0.010	$\mathbf{0.125 \pm 0.006}$	$\mathbf{0.077 \pm 0.001}$
PACOH-MAP	0.105 ± 0.009	0.054 ± 0.005	0.085 ± 0.003	0.175 ± 0.004	0.151 ± 0.006	0.084 ± 0.003
ABLR	0.180 ± 0.004	0.049 ± 0.002	$\mathbf{0.044 \pm 0.005}$	0.220 ± 0.005	0.158 ± 0.010	0.097 ± 0.004
NP	0.146 ± 0.006	0.053 ± 0.010	0.063 ± 0.009	0.202 ± 0.009	0.176 ± 0.013	0.197 ± 0.038
Learned GP	0.112 ± 0.000	0.069 ± 0.001	0.062 ± 0.000	0.125 ± 0.000	0.137 ± 0.001	0.107 ± 0.001
Vanilla GP	0.123 ± 0.000	0.085 ± 0.000	0.089 ± 0.003	$\mathbf{0.123 \pm 0.000}$	0.150 ± 0.000	0.182 ± 0.000

Table 5: Calibration error of various meta-learned models as well as a Vanilla GP on meta-test tasks generated by uniform sampling from the meta-BO benchmark environments.

experimental findings in Section 5.1 and the results of the BO benchmark studies where F-PACOH performs significantly better than the baselines.

B.5.2 Offline Meta-Learning

In addition to the simple regret results (see Figure 3), we also provide plots of the inference regret in Figure 5. Note that, since random search does not maintain a machine learning model of the target function, the concept of inference regret does not apply to it. Thus it is not included here.

Overall, the inference regret results in Figure 5 show the same patterns as the simple regret results - F-PACOH significantly outperforms the other methods across all the environments.

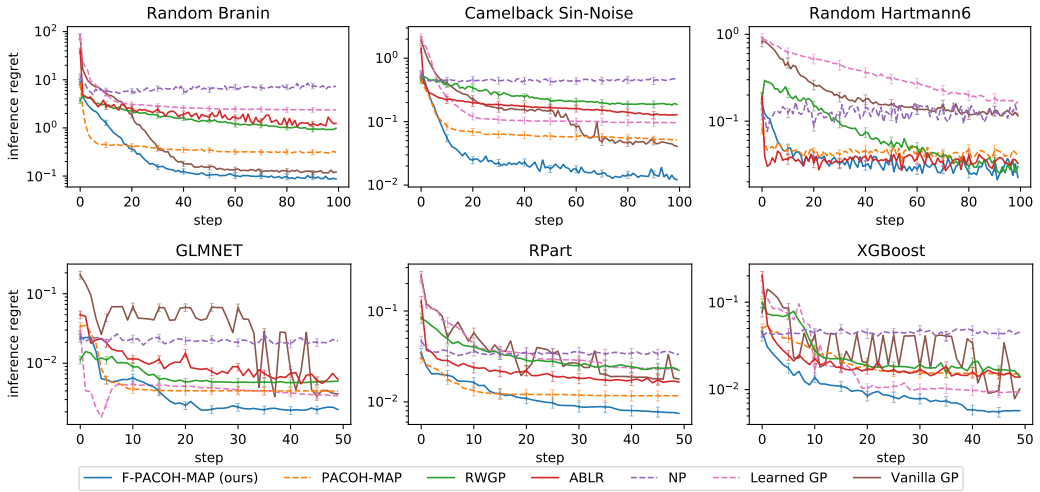

Figure 5: Performance of BO with meta-learned models on simulated function environments (top) and hyper-parameter tuning (bottom). Reported is the inference regret in log-scale, averaged over seeds and function samples, alongside 95% confidence intervals. Consistent with the simple regret, displayed Figure 3, FPACOH significantly outperforms the other methods across all environments.

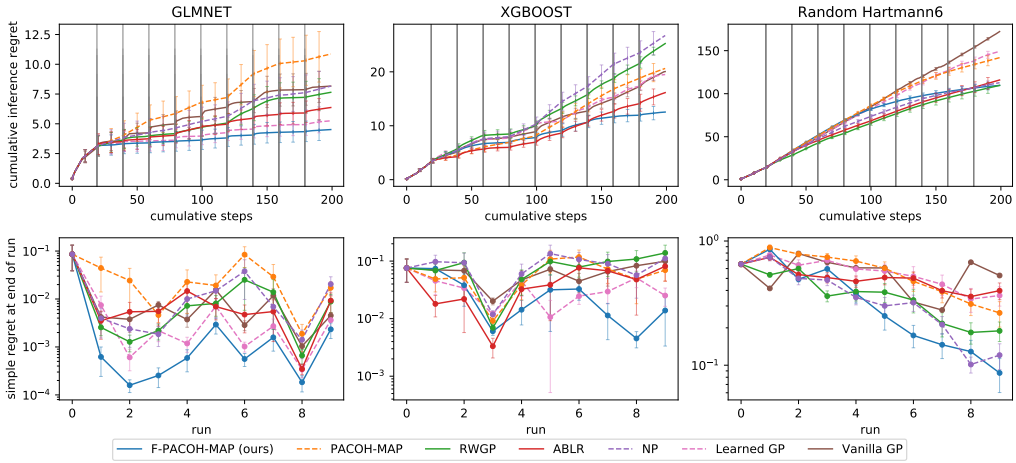

Figure 6: Lifelong BO performance on two simulated function environments and one hyper-parameter tuning benchmark (RPart). The reported results are averages over seeds and random sequences of tasks alongside 95% confidence intervals. While other meta-learners struggle to achieve positive transfer, F-PACOH is able to significantly improve the BO perfomance as it gathers more experience.

B.5.3 Lifelong Bayesian Optimization

In addition to Figure 4 which displays the the results of our lifelong BO study for three environments, we provide analogous plots for the remaining three environments GLMNET, XGBOOST and Random Hartmann6 in Figure 6. Similar to the previous results, at any time throughout the course of the lifelong BO episode performs among the best and, unlike other methods, keeps improving across the later BO runs.