Reinforced Few-Shot Acquisition Function Learning for Bayesian Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Bayesian optimization (BO) conventionally relies on handcrafted acquisition functions (AFs) to sequentially determine the sample points. However, it has been widely observed in practice that the best-performing AF in terms of regret can vary significantly under different types of black-box functions. It has remained a challenge to design one AF that can attain the best performance over a wide variety of black-box functions. This paper aims to attack this challenge through the perspective of reinforced few-shot AF learning (FSAF). Specifically, we first connect the notion of AFs with Q-functions and view a deep Q-network (DQN) as a surrogate differentiable AF. While it serves as a natural idea to combine DQN and an existing few-shot learning method, we identify that such a direct combination does not perform well due to severe overfitting, which is particularly critical in BO due to the need of a versatile sampling policy. To address this, we present a Bayesian variant of DQN with the following three features: (i) It learns a distribution of Q-networks as AFs based on the Kullback-Leibler regularization framework. This inherently provides the uncertainty required in sampling for BO and mitigates overfitting. (ii) For the prior of the Bayesian DQN, we propose to use a demo policy induced by an off-the-shelf AF for better training stability. (iii) On the meta-level, we leverage the meta-loss of Bayesian model-agnostic meta-learning, which serves as a natural companion to the proposed FSAF. Moreover, with the proper design of the Q-networks, FSAF is general-purpose in that it is agnostic to the dimension and the cardinality of the input domain. Through extensive experiments, we demonstrate that the FSAF achieves comparable or better regrets than the state-of-the-art benchmarks on a wide variety of synthetic and real-world test functions.

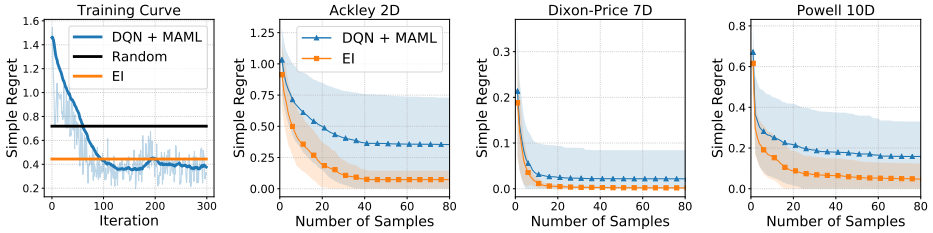
1 Introduction

Bayesian optimization (BO) has served as a powerful and popular framework for global optimization in many real-world tasks, such as hyperparameter tuning [1–4], robot control [5], automatic material design [6–8], etc. To search for global optima under a small sampling budget and potentially noisy observations, BO imposes a Gaussian process (GP) prior on the unknown black-box function and continually updates the posterior as more samples are collected. BO relies on *acquisition functions* (AFs) to determine the sample location, i.e., those with larger AF values are prioritized than those with smaller ones. AFs are often designed to capture the trade-off between exploration and exploitation of the global optima. The design of AFs has been extensively studied from various perspectives, such as optimism in the face of uncertainty (e.g., GP-UCB [9]), optimizing information-theoretic metrics (e.g., entropy search methods [10–12]), and maximizing one-step improvement (e.g., expected improvement or EI [13, 14]). As a result, AFs are often handcrafted according to different perspectives of the trade-off, and the best-performing AFs can vary significantly under different types of black-box

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

functions [11]. This phenomenon is repeatedly observed in our experiments in Section 4. Therefore, one critical issue in BO is to design an AF that can adapt to a variety of black-box functions.

To achieve better adaptability to new tasks in BO, recent works propose to leverage *meta-data*, the data previously collected from similar tasks [15–17]. For example, in the context of hyperparameter optimization under a specific dataset, the meta-data could come from the evaluation of previous hyperparameter configurations for the same learning model over any other related dataset. In [15], meta-data is used to fine-tune the initialization of the GP parameters and thereby achieves better GP model selection for each specific task. However, the potential benefit of using meta-data for more efficient exploration via AFs is not explored. On the other hand, in [16, 17], meta-data is split into multiple subsets and then used to construct a transferable acquisition function based on some off-the-shelf AF (e.g. EI) and an ensemble of GP models. Each of the GP models is learned over a separate subset of the meta-data. However, to achieve effective knowledge transfer, this approach would require a sufficiently large amount of meta-data, which significantly limits its practical use. As a result, there remains a critical unexplored challenge in BO: *how to design an AF that can effectively adapt to a wide variety of back-box functions given only a small amount of meta-data?*



(a) (b) (c) (d)

Figure 1: An illustration of the overfitting issue of DQN+MAML trained with GP functions: (a) Training curves of DQN+MAML (As EI and Random do not require any training, their lines are flat); (b)-(d) Average simple regrets of DQN+MAML and EI in testing under benchmark functions.

To tackle this challenge, we propose to rethink the use of meta-data in BO through the lens of *few-shot acquisition function (FSAF) learning*. Specifically, our goal is to learn an initial AF model that allows few-shot fast adaptation to each specific task during evaluation. Inspired by the similarity between AFs and Q-functions, we use a deep Q-network (DQN) as a surrogate differentiable AF, i.e., the Q-network would output an indicator for each candidate sample point given its posterior mean and variance as well as other related information. Given the parametric nature and the differentiability of a Q-network, it is natural to leverage optimization-based few-shot learning approaches, such as model-agnostic meta-learning (MAML) [18], for the training of few-shot AFs. Despite this natural connection, we find that a direct combination of standard DQN and MAML (DQN+MAML) is prone to overfitting, as illustrated by the comparison of training and testing results. Note that in Figure 1 (with detailed configuration in Appendix B), DQN+MAML achieves better regret than EI on the training set but suffers from much higher regret during testing. We hypothesize this is because both DQN and MAML are prone to overfitting [19–22]. This issue could be particularly critical in BO due to the uncertainty required in adaptive sampling. Based on our findings and inspired by [23], we propose a Bayesian variant of DQN with the following three salient features: (i) The Bayesian DQN learns a distribution of parameters of DQN based on the Kullback-Leibler (KL) regularization framework. Through this, the problem of minimizing the Q-learning temporal-difference (TD) error in DQN is converted into a Bayesian inference problem. (ii) For the prior of the Bayesian DQN, we propose to use a demonstration policy induced by an off-the-shelf AF to further stabilize the training process; (iii) We then use the chaser meta-loss in [20], which serves as a natural companion to the proposed Bayesian DQN. As shown by the experimental results in Section 4, the proposed design effectively mitigates overfitting and achieves good generalization under various black-box functions. Moreover, with the proper design of the Q-networks, the proposed FSAF is general-purpose in the sense that it is agnostic to both the dimension and the cardinality of the input domain.

The main contributions of this paper can be summarized as follows:

- We consider a novel setting of few-shot acquisition function learning for BO and present the first few-shot acquisition function that can use a small amount of meta-data to achieve better task-specific exploration and thereby effectively adapt to a wide variety of black-box functions.

- Inspired by the similarity between AFs and Q-functions, we view DQN as a parametric and differentiable AF and use it as the base of our FSAF. We identify the important overfitting issue in the direct combination of DQN and MAML and thereafter present a Bayesian variant of DQN that mitigates overfitting and enjoys stable training through a demo-based prior.
- We extensively evaluate the proposed FSAF in a variety of tasks, including optimization benchmark functions, real-world datasets, and synthetic GP functions. We show that the proposed FSAF can indeed effectively adapt to a variety of tasks and outperform both the conventional benchmark AFs as well as the recent state-of-the-art meta-learning BO methods.

2 Preliminaries

Our goal is to design a sampling policy to optimize a black-box function $f : \mathbb{X} \to \mathbb{R}$, where $\mathbb{X} \subset \mathbb{R}^d$ denotes the compact domain of f. f is black-box in the sense that there are no special structural properties (e.g., concavity or linearity) or derivatives (e.g., gradient or Hessian) about f available to the sampling policy. In each step t, the policy selects $x_t \in \mathbb{X}$ and obtains a noisy observation $y_t = f(x_t) + \varepsilon_t$, where ε_t are i.i.d. zero-mean Gaussian noises. To evaluate a sampling policy, we define the *simple regret* as $\text{Regret}(t) := \max_{x \in \mathbb{X}} f(x^*) - \max_{1 \le s \le t} f(x_s)$, which quantifies the best sample up to t. For convenience, we let $\mathcal{F}_t := \{(x_i, y_i)\}_{i=1}^{t-1}$ denote the observations up to step t.

Bayesian optimization. To optimize f in a sample-efficient manner, BO first imposes on the space of objective functions a GP prior, which is fully characterized by a mean function and a covariance function, and then determines the next sample based on the resulting posterior [24, 25]. In each step t, given $\mathcal{F}_t$, the posterior predictive distribution of each $x \in \mathbb{X}$ is $\mathcal{N}(\mu_t(x), \sigma_t^2(x))$, where $\mu_t(x) := \mathbb{E}[f(x) | \mathcal{F}_t]$ and $\sigma_t(x) := (\mathbb{V}[f(x) | \mathcal{F}_t])^{\frac{1}{2}}$ can be derived in closed form. In this way, BO can be viewed as a sequential decision making problem. However, it is typically difficult to obtain the exact optimal policy due to the curse of dimensionality [24]. To obtain tractable policies, BO algorithms construct AFs $\Psi(x; \mathcal{F}_t)$, which resort to maximizing one-step look-ahead objectives based on $\mathcal{F}_t$ and the posterior [24]. For example, EI chooses the sample location based on the improvement made by the immediate next sample in expectation, i.e., $\Psi_{\text{EI}}(x; \mathcal{F}_t) = \mathbb{E}[f(x) - \max_{1 \le i \le t-1} f(x_i) | \mathcal{F}_t]$, which enjoys a closed form in $\mu_t(x)$, $\sigma_t(x)$, and $\max_{1 \le i \le t-1} f(x_i)$.

Meta-learning with few-shot fast adaptation. Meta-learning is a generic paradigm for generalizing the knowledge acquired during training to solving unseen tasks in the testing phase. In the few-shot setting, meta-learning is typically achieved via a bi-level framework: (i) On the upper level, the training algorithm is meant to determine a proper initial model with an aim to facilitating subsequent task-specific adaptation; (ii) On the lower level, given the initial model and the task of interest, a fast adaptation subroutine is configured to fine-tune the initial model based on a small amount of task-specific data. Specifically, during training, the learner finds a model parameterized by θ based on a collection of tasks $\mathcal{T}$, where each task $\tau \in \mathcal{T}$ is associated with a training set $\mathcal{D}_\tau^{\text{tr}}$ and a validation set $\mathcal{D}_\tau^{\text{val}}$. For any initial model parameters θ and training set $\mathcal{D}^{\text{tr}}$ of a task τ, let $\mathcal{M}(\theta, \mathcal{D}_\tau^{\text{tr}})$ be an algorithm that outputs the adapted model parameters by applying few-shot fast adaptation to θ based on $\mathcal{D}_\tau^{\text{tr}}$. The performance of the adapted model is evaluated on $\mathcal{D}_\tau^{\text{val}}$ by a meta-loss function $\mathcal{L}(\mathcal{M}(\theta, \mathcal{D}_\tau^{\text{tr}}), \mathcal{D}_\tau^{\text{val}})$. Accordingly, the overall training can be viewed as solving the following optimization problem:

$$\theta^* := \operatorname*{argmin}_{\theta} \sum_{\tau \in \mathcal{T}} \mathcal{L}(\mathcal{M}(\theta, \mathcal{D}_\tau^{\text{tr}}), \mathcal{D}_\tau^{\text{val}}). \tag{1}$$

By properly configuring the loss function, the formulation in (1) is readily applicable to various learning problems, including supervised learning and RL. Note that the adaptation subroutine is typically chosen as taking one or a few gradient steps with respect to $\mathcal{L}(\cdot, \cdot)$ or some relevant loss function. For example, under the celebrated MAML [18], the adaptation subroutine is $\mathcal{M}(\theta, \mathcal{D}_\tau^{\text{tr}}) \equiv \theta - \eta \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_\tau^{\text{tr}})$, where η denotes the learning rate.

Reinforcement learning and Q-function. Following the conventions of RL, we use s_t, a_t, and r_t to denote the state, action, and reward obtained at each step t. Let R and γ be the reward function and the discount factor. The goal is to find a stationary randomized policy that maximizes the total expected discounted reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$. To achieve this, given a policy π, a helper function termed Q-function is defined as $Q^\pi(s, a) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_o = s, a_0 = a; \pi]$. The optimal Q-function can then be defined as $Q^*(s, a) := \max_\pi Q^\pi(s, a)$, for each state-action pair. One fundamental property of the optimal Q-function is the *Bellman optimality equation*, i.e.,

$Q^*(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') \middle| s_t = a, a_t = a\right]$. Then, the Bellman optimality operator can be defined by $[\mathcal{B}Q](s,a) := R(s,a) + \gamma\, \mathbb{E}_{s'}[\max_{a' \in \mathcal{A}} Q(s', a')]$. It is known that Q^* is the unique fixed point of $\mathcal{B}$. We leverage this fact to describe the proposed FSAF in Section 3.2.

3 Few-Shot Acquisition Function

3.1 Deep Q-Network as a Differentiable Parametric Acquisition Function

Based on the conceptual similarity between acquisition functions and Q-functions, in this section we present how to cast a deep Q-network as a parametric and differentiable instance of acquisition function. To begin with, we consider the Q-network architecture with state and action representations as the input, as typically adopted by Q-learning for large action spaces [26].

State-action representation. In BO, an action corresponds to choosing one location to sample from the input domain $\mathbb{X}$, and the state at each step t can be fully captured by the collection of sampled points $\{(x_i, y_i)\}_{i=1}^{t-1}$. However, this raw state representation appears problematic as its dimension depends on the number of observed sample points. Inspired by the acquisition functions, we leverage the posterior mean and variance as the *joint state-action representation* for each candidate sample location. In addition, we include the best observation so far (defined as $y_t^* := \operatorname{argmax}_{1 \le i \le t-1} y_i$) and the ratio between the current timestamp and total sampling budget T, which reflects the sampling progress in BO. In summary, at each step t, the state-action representation of each $x \in \mathbb{X}$ is designed to be a 4-tuple $(\mu_t(x), \sigma_t(x), y_t^*, \frac{t}{T})$, which is agnostic to the dimension and cardinality of $\mathbb{X}$.

Reward signal. To reflect the sampling progress, we define the reward r_t as a function of the simple regret, i.e., $r_t = g(\text{Regret}(t))$, where $g : \mathbb{R}_+ \to \mathbb{R}$ is a strictly decreasing function. Practical examples include $g(z) = -z$ and $g(z) = -\log z$.

Remark 1 One popular approach to construct a representation of fixed dimension is through embedding. From this viewpoint, the posterior mean and variance can be viewed as a natural embedding generated by GP inference in the context of BO. It is an interesting direction to extend the proposed design by constructing more general state and action representations via embedding techniques.

3.2 A Bayesian Perspective of Deep Q-Learning for Bayesian Optimization

One major challenge in designing an acquisition function for BO is to address the wide variability of black-box functions and accordingly achieve a favorable explore-exploit trade-off for each task. To address this by deep Q-learning as described in Section 3.2, instead of learning a single Q-network as in the standard DQN [27], we propose to learn a distribution of Q-network parameters from a Bayesian inference perspective to achieve more robust exploration and more stable training. Inspired by [23], we adapt the regularized minimization problem to Q-learning in order to connect Q-learning and Bayesian inference as follows. Let $C(\theta)$ be the cost function that depends on the model parameter θ. Instead of finding a single model, the Bayesian approach finds a distribution $q(\theta)$ over θ that minimizes the cost function augmented with a KL-regularization penalty, i.e.,

$$\min_{q(\theta)} \Big\{ \mathbb{E}_{\theta \sim q(\theta)}[C(\theta)] + \alpha D_{\text{KL}}(q \parallel q_0) \Big\}, \tag{2}$$

where q_0 denotes a prior distribution over θ and α is a weight factor of the penalty and $D_{\text{KL}}(\cdot \parallel \cdot)$ denotes the Kullback-Leibler (KL) divergence between two distributions. Note that $q(\theta)$ essentially induces a distribution over the sampling policies π_θ, and accordingly q_0 can be interpreted as constructing a prior over the policies. By setting the derivative of the objective in (2) with respect to the measure induced by q to be zero, one can verify that the optimal solution to (2) is

$$q^*(\theta) = \frac{1}{Z} \exp\Big(\frac{-C(\theta)}{\alpha}\Big) q_0(\theta), \tag{3}$$

where Z is the normalizing factor. One immediate interpretation of (3) is that $q^*(\theta)$ can be viewed as the posterior distribution under the prior distribution $q_0(\theta)$ and the likelihood $\exp(-C(\theta)/\alpha)$. To adapt the KL-regularized minimization framework to value-based RL for BO, the proposed FSAF algorithm is built on the following design principles for $C(\theta)$ and $q_0(\theta)$:

- **Use mean-squared TD error as the cost function:** Recall from Section 2 that the optimal Q-function is the fixed point of the Bellman optimality backup operation. Hence, we have $\mathcal{B}Q = Q$

if and only if Q is the optimal Q-function. Based on this observation, one principled choice of $C(\theta)$ is the squared TD error under the operator $\mathcal{B}$, i.e., $\|\mathcal{B}Q - Q\|_2^2$. Moreover, in practice, DQN typically incorporates a replay buffer $\mathcal{R}_Q$ (termed the Q-replay buffer) as well as a target Q-network to achieve better training stability [27]. Therefore, we choose the cost function as

$$C(\theta) = \mathbb{E}_{(s,a,s',r)\sim\rho}\Big[\Big(\big(r + \gamma \max_{a'\in\mathcal{A}} Q(s', a'; \theta^-)\big) - Q(s, a; \theta)\Big)^2\Big], \tag{4}$$

where ρ denotes the underlying sample distribution of the replay buffer and θ^- is the parameter of the target Q-network[1]. In practice, the cost $C(\theta)$ is estimated by the empirical average over a mini-batch $\mathcal{D}$ of samples (s, a, r, s') drawn from the replay buffer, i.e., $C(\theta) \approx \hat{C}(\theta) = \frac{1}{|\mathcal{D}|}\sum_{(s,a,r,s')\in\mathcal{D}}((r + \gamma \max_{a'\in\mathcal{A}} Q(s', a'; \theta^-)) - Q(s, a; \theta))^2$.

- **Construct an informative prior with the help of the existing acquisition functions:** In (2), the KL-penalty with respect to a prior distribution is meant to encode prior domain knowledge as well as provide regularization that prevents the learned parameter from collapsing into a point estimate. One commonly-used choice is a uniform prior (i.e., $q(\theta) = c$ for some positive constant c), under which the KL-penalty reduces to the negative entropy of q. Given that BO is designed to optimize expensive-to-evaluate functions, it is therefore preferred to use a more informative prior for better sample efficiency. Based on the above, we propose to construct a prior with the help of a demo policy π_{D} induced by existing popular AFs (e.g., EI or PI), which inherently capture critical information structure of the GP posterior. Define a similarity indicator $\delta(\pi_\theta, \pi_{\mathrm{D}})$ of π_θ and π_{D} as

$$\delta(\pi_\theta, \tilde{\pi}) := \mathbb{E}_{s\sim\rho, a\sim\pi_{\mathrm{D}}(\cdot|s)}\big[\log(\pi_\theta(s, a))\big], \tag{5}$$

where we slightly abuse the notation and let ρ denote the state distribution induced by the replay buffers. Since the term $\log(\pi_\theta(s, a))$ in (5) is the log-likelihood of that the action of π_θ matches that of π_{D} at a state s, $\delta(\pi_\theta, \pi_{\mathrm{D}})$ reflects how similar the two policies are on average (with respect to the state visitation distribution of π_θ). Accordingly, we propose to design the prior $q_0(\theta)$ to be

$$q_o(\theta) \propto \exp\big(\delta(\pi_\theta, \pi_{\mathrm{D}})\big). \tag{6}$$

As it is typically difficult to directly evaluate $\delta(\pi_\theta, \pi_{\mathrm{D}})$ in practice, we construct another replay buffer $\mathcal{R}_{\mathrm{D}}$ (termed the *demo* replay buffer), which stores the state-action pairs produced under the state distribution ρ and the demo policy π_{D}, and estimate $\delta(\pi_\theta, \pi_{\mathrm{D}})$ by the empirical average over a mini-batch $\mathcal{D}'$ of state-action pairs drawn from the demo replay buffer, i.e., $\delta(\pi_\theta, \pi_{\mathrm{D}}) \approx \hat{\delta}(\pi_\theta, \pi_{\mathrm{D}}) = \frac{1}{|\mathcal{D}'|}\sum_{(s,a)\in\mathcal{D}'} \log(\pi_\theta(s, a))$.

- **Update the Q-networks by Stein variational gradient descent:** The solution in (3) is typically intractable to evaluate and hence approximation is required. We leverage the Stein variational gradient descent (SVGD), which is a general-purpose approach for Bayesian inference. Specifically, we build up N instances of Q-networks (also called *particles* in the context of the variational methods) and update the parameters via Bayesian inference. Let $\theta^{(n)}$ denote the parameters of the n-th Q-network and use Θ as a shorthand of $\{\theta^{(n)}\}_{n=1}^N$. Under SVGD [28] and the prior described in (6), the Stein variational gradient of each particle can be derived as

$$g^{(n)}(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta^{(i)}}\Big(\frac{-1}{\alpha}C(\theta^{(i)}) + \delta(\pi_{\theta^{(i)}}, \pi_{\mathrm{D}})\Big)k(\theta^{(i)}, \theta^{(n)}) + \nabla_{\theta^{(i)}}k(\theta^{(i)}, \theta^{(n)}), \tag{7}$$

where $k(\cdot, \cdot)$ is a kernel function. As mentioned above, in practice $C(\theta)$ and $\delta(\pi_\theta, \pi_{\mathrm{D}})$ are estimated by the corresponding empirical $\hat{C}(\theta)$ and $\hat{\delta}(\pi_\theta, \pi_{\mathrm{D}})$, respectively. Let $\hat{g}^{(n)}(\Theta)$ denote the estimated Stein variational gradient based on $\hat{C}(\theta)$ and $\hat{\delta}(\pi_\theta, \pi_{\mathrm{D}})$. Accordingly, the parameters of each Q-network are updated iteratively by SVGD as

$$\theta^{(n)} \leftarrow \theta^{(n)} + \eta \cdot \hat{g}^{(n)}(\Theta), \tag{8}$$

where η is the learning rate. For ease of notation, we use $\mathcal{M}_{\mathrm{SVGD}}(\Theta, \mathcal{D})$ to denote the subroutine that applies one SVGD update of (8) to all Q-networks parameterized by Θ based on some dataset $\mathcal{D}$. Note that the update scheme in (8) serves as a natural candidate for the few-shot adaptation subroutine of the meta-learning framework described in Section 2. In Section 3.3, we will put everything together and describe the full meta-learning algorithm of FSAF.

[1] In (4), the cost function depends implicitly on the target network parameterized by θ^-. Despite this, as the target network is updated periodically from $Q(\cdot, \cdot; \theta)$, for notational convenience we do not make explicit the dependence of the cost function on θ^- in the notation $C(\theta)$.

Remark 2 The presented Bayesian DQN bears some high-level resemblance to the prior works [29–32], which are inspired by the classic principle of Thompson sampling for exploration. In [30, 31], the Q-function is assumed to be linearly parameterized with a Gaussian prior on the parameters such that the posterior can be computed in closed form. On the other hand, without imposing the linearity assumption, [31] approximates the intractable posterior by maintaining an ensemble of Q-networks as a practical heuristic of Thompson sampling to DQN. Different from [29–31], we approach Bayesian DQN through the principled framework of KL regularization for parametric Bayesian inference and solve it via SVGD, without any linearity assumption. [32] starts from an entropy-regularized formulation for Q-learning and assumes that the target Q-value is drawn from a Gaussian model to obtain a tractable posterior from the perspective of variational inference. By contrast, we do not rely on the Gaussian assumption and directly find the posterior by SVGD, and for more efficient training we consider a prior induced by an acquisition function. More importantly, the presented Bayesian DQN naturally helps substantiate the few-shot learning framework described in Section 2 for BO.

Remark 3 The regularized formulation in (2) has been extensively applied in the class of policy-based methods in RL. For example, the entropy-regularized policy optimization [33] has been applied to enable a "soft version" of policy iteration, which gives rise to the popular soft Q-learning [34] and soft actor-critic algorithms [35]. Another example is the Stein variational policy gradient method [23], which connects the policy gradient methods with Bayesian inference. Different from the prior works, we take a different path to connect the value-based RL approach with Bayesian inference.

Remark 4 The similarity indicator defined in (5) has a similar form as the loss term in some of the classic imitation learning algorithms. For example, given an expert policy π_e, DAgger [36] is designed to find a policy π' that minimizes a surrogate loss $\mathbb{E}_s[\ell(s, \pi_e)]$, where $\ell(\cdot, \cdot)$ is some loss function (e.g., 0-1 loss or hinge loss) that reflects the dissimilarity between π' and π_e. Despite this high-level resemblance, one fundamental difference between (5) and imitation learning is that the demo policy π_{D} is not a true expert in the sense that mimicking the behavior of π_{D} is not the ultimate goal of the FSAF learning process. Instead, the goal of FSAF is to learn a policy that can better adapt to new tasks and thereby outperform the existing AFs in various domains. The penalty defined in (6) is only meant to provide some prior domain knowledge to achieve more sample-efficient training. We further validate this design by providing training curves in Section 4.

3.3 Meta-Learning via Bayesian MAML

Based on the Bayesian DQN design in Section 3.2, the natural way to substantiate the meta-learning framework in (1) is to leverage the Bayesian variant of MAML [20]. In the context of BO, each task typically corresponds to optimizing some type of black-box functions (e.g., GP functions from an RBF kernel with some lengthscale). FSAF implements the bi-level framework of (1) as follows: (i) On the lower level, for each task τ, FSAF enforces few-shot fast adaptation by taking K steps of SVGD as described in (8) and thereafter obtains the fast-adapted parameters denoted by $\Theta_{\tau,K}$; (ii) On the upper level, for each task τ, FSAF computes a meta-loss that reflects the dissimilarity between the approximated posterior induced by $\Theta_{\tau,K}$ and the true posterior distribution in (6). As the true posterior is not available, one practical solution is to approximate the true posterior by taking S additional SVGD gradient steps based on $\Theta_{\tau,K}$ and obtaining a surrogate denoted by $\Theta^*_{\tau,S}$ [20]. This design can be justified by the fact that $\Theta^*_{\tau,S}$ becomes a better approximation for the true posterior as S increases due to the nature of SVGD. For any two collections of particles $\Theta' \equiv \{\theta'^{(n)}\}$ and $\Theta'' \equiv \{\theta''^{(n)}\}$, define $D(\Theta', \Theta'') := \sum_{n=1}^{N} \|\theta'^{(n)} - \theta''^{(n)}\|_2^2$ (called *chaser loss* in [20]). Then, for any task τ, the meta-loss of FSAF is computed as

$$\mathcal{L}_{\text{meta}}(\Theta; \tau) = D\big(\Theta_{\tau,K}, \text{stopgrad}(\Theta^*_{\tau,S})\big), \tag{9}$$

As will be shown by the experiments in Section 4, using small K and S empirically leads to favorable performance. The pseudo code of the training procedure of FSAF is provided in Appendix C.

Remark 5 This paper focuses on value-based methods for training a few-shot acquisition function. Based on the proposed training framework, it is also possible to extend the idea to design an actor-critic counterpart of our FSAF. We believe this is an interesting direction for future work.

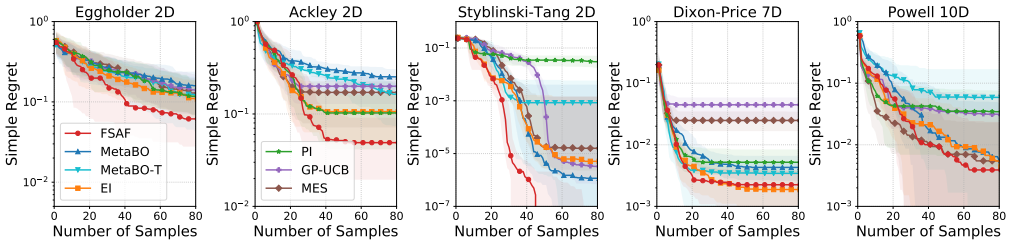
4 Experimental Results

We demonstrate the effectiveness of FSAF on a wide variety of classes of black-box functions and discuss how FSAF addresses the critical challenges described in Section 1. Unless stated otherwise,

we report the median of simple regrets over 100 evaluation trials along with the 25% and 75% percentiles (shown via shaded areas or tables in Appendix D for clarity).

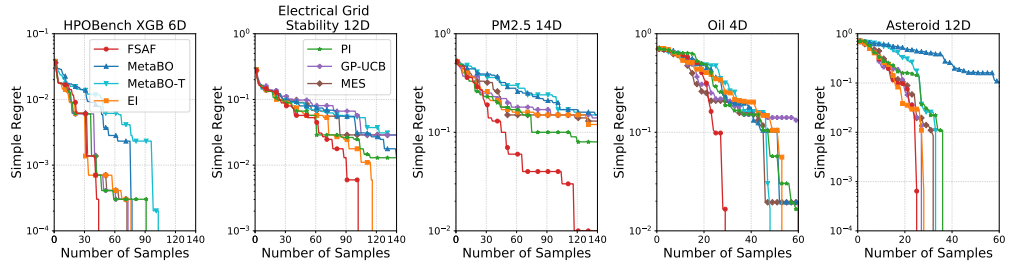
Popular benchmark methods. We evaluate FSAF against various popular benchmark methods, including EI [13], PI [37], GP-UCB [9], MES [12], and MetaBO [38]. The configuration and hyperparameters of the above methods are as follows. GP-UCB, EI, and PI are classic general-purpose AFs that have been shown to achieve good regret performance for black-box functions drawn from GP. For GP-UCB, we tune its exploration parameter δ by a grid search between 10^{-1} and 10^{-6}. Among the family of entropy search methods, MES is a strong and computationally efficient benchmark method that achieves superior performance for several global optimization benchmark functions and GP functions [12]. For MES, we use Gumbel sampling and take one sample for y_*, as suggested by the original paper [12]. MetaBO is a neural AF trained via policy-based RL and recently achieves superior performance in BO. For fair and reproducible evaluations, we use the pre-trained model of MetaBO provided by [38]. As the original MetaBO does not address the use of meta-data, for a more comprehensive comparison, we further consider a few-shot variant of MetaBO (termed MetaBO-T), which is obtained by performing 100 more training iterations on the pre-trained MetaBO model using the meta-data for few-shot fine-tuning.

Configuration of FSAF. For training, we construct a collection of training tasks, each of which is a class of GP functions with either an RBF, Matern-3/2, or a spectral mixture kernel with different parameters (e.g., lengthscale and periods). We take $N = 5$, $K = 5$, and $S = 1$ given the memory limitation of GPUs. Despite the small values of N, K, S, these choices already provide superior performance. For the reward design of FSAF, we use $g(z) = -\log z$ to encourage high-accuracy solutions. For testing, we use the model with the best average total return during training as our initial model, which is later fine-tuned via few-shot fast adaptation for each task. For a fair comparison, we ensure that FSAF and MetaBO-T use the same amount of meta-data in each experiment.



(a) (b) (c) (d) (e)

Figure 2: Simple regrets of FSAF and the benchmark methods for optimization benchmark functions. The line and the shaded areas show the median and the 25%/75% percentiles, respectively.



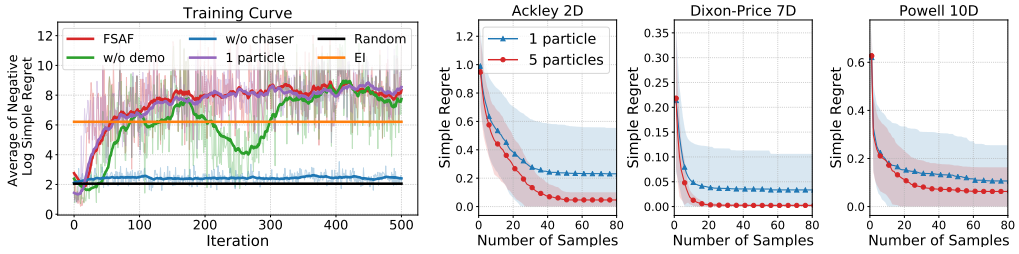
(a) (b) (c) (d) (e)

Figure 3: Median simple regrets of FSAF and other benchmark methods for real-world test functions. Other percentiles are provided in Appendix D for visual clarity.

Does FSAF adapt effectively to a wide variety of black-box functions? To answer this, we first evaluate FSAF and the benchmark methods on several types of standard optimization benchmark functions, including: (i) *Ackley* and *Eggholder*, which are functions with a large number of local optima but with different symmetry structures; (ii) *Dixon-Price*, a valley-shaped function; (iii) *Styblinski-Tang*, a smooth function with a couple of local optima; (iv) *Powell*, a 10-dimensional function (the highest input dimension among the five functions). As the y values of these functions

can be one or more orders of magnitude different from each other, for ease of comparison, we scale all the values of the functions to $[-2, 2]$. Such scaling still preserves the salient structure and variations of each test function. To construct the training and testing sets, we apply random translations and re-scalings of up to +/- 10% to x and y values, respectively. In this case, we consider 5-shot adaptation for FSAF and use the same amount of meta-data for MetaBO-T. From Figure 2, we observe that FSAF is constantly the best or among the best of all the methods under all the test functions. We observe that MetaBO performs poorly under functions with many local optima but performs better under smooth functions (e.g., Styblinski-Tang and Powell). This might be due to the fact that MetaBO was trained with smooth GP functions and lacks the ability to adapt to functions with more local variations. We also find that MetaBO-T benefits from meta-data and improves upon MetaBO in some functions. While this manifests the potential benefits of using meta-data for MetaBO, this also suggests that brute-force fine-tuning is not effective and a more careful design like FSAF is needed. Moreover, among the 6 benchmark methods, the best-performing AF indeed varies under different types of functions. This corroborates the commonly-seen phenomenon and our motivation.

We proceed to evaluate FSAF on test functions obtained from five open-source real-world tasks in different application domains. Based on the smoothness characteristics[2], the datasets can be categorized as: (i) Smooth in all dimensions: Asteroid size prediction ($d = 12$); (ii) Smooth in all but one dimension: hyperparameter optimization for XGBoost ($d = 6$) and air quality prediction in PM 2.5 ($d = 14$); (iii) Smooth in about half of the dimensions: maximization of electric grid stability ($d = 12$); (iv) Non-smooth in all dimensions: location selection for oil wells ($d = 4$). The detailed description of the datasets is in Appendix B. In this setting, we consider 1-shot adaptation for FSAF, a rather sample-efficient scenario of few-shot learning. From Figure 3, we observe that FSAF remains the best or among the best for all the five real-world test functions, despite the salient structural differences of the datasets. For a more comprehensive comparison, we also evaluate all the AFs on synthetic functions from GP and observe the similar behavior. Due to space limitation, the results of GP functions are provided in Appendix D. Based on the above discussion, we confirm that FSAF indeed achieves favorable adaptability.



(a) (b) (c) (d)

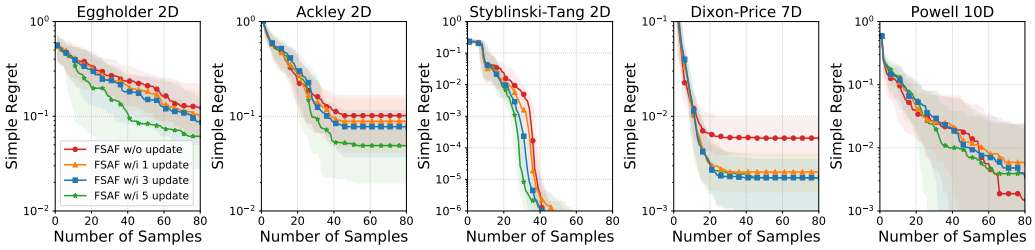
Figure 4: Ablation study for FSAF: (a) Training curves of FSAF and its ablations (EI and Random as baselines); (b)-(d) Testing performance of FSAF with 1 and 5 particles of Q-networks.

Does FSAF mitigate the overfitting issue? To answer this, we show the training curves of FSAF and its ablations, including: (i) FSAF without using the demo replay buffer (termed "w/o demo"); (ii) FSAF with the chaser meta-loss replaced by TD loss (termed "w/o chaser"); (iii) FSAF with only 1 particle (termed "1 particle"), which is equivalent to DQN+MAML with an additional demo replay buffer. The training metric plotted is the negative logarithm of simple regret at $t = 30$ (averaged over episodes in each iteration), which is consistent with the rewards of FSAF and can better demonstrate the differences in training. The results of EI and random sampling are given as references.

- **The Bayesian variant of DQN does mitigate overfitting.** This is confirmed by the fact that the training curves of FSAF with 1 and 5 particles are quite close, while in Figures 4(b)-4(d) the testing performance of FSAF with 5 particles appears much better than that of 1 particle.
- **The demo-based prior helps improve the training stability.** This is verified by that without the demo replay buffer, the training progress becomes apparently slower initially and is subject to more variations throughout the training.
- **The chaser meta-loss appears effective in FSAF.** Interestingly, we find that chaser meta-loss results in stable and effective training, while the TD meta-loss can barely make any progress.

[2]To better understand the characteristics of each real-world dataset, we extract the smoothness information through marginal likelihood maximization on a surrogate GP model with an RBF kernel.

Does FSAF benefit from the few-shot gradient updates? To better understand the effect of few-shot gradient updates, Figure 5 shows the simple regrets of FSAF under different number of gradient updates (i.e., K defined in Section 3.3) for the optimization benchmark functions. We find that the adaptation effect does increase with K for small K's for most of the cases. This appears consistent with the general observations of MAML-like algorithms [18].



(a) (b) (c) (d) (e)

Figure 5: Simple regrets of FSAF vs number of gradient steps for optimization benchmark functions.

Remark 6 (FSAF and [15]) As mentioned in Section 1, [15] proposes to leverage meta-data to fine-tune the initialization of the GP kernel parameters for the off-the-shelf AFs (called FSBO in [15]). By contrast, FSAF uses meta-data for fast adaptation of an AF, which is a direction orthogonal to FSBO. Moreover, it is possible for our FSAF and FSBO to complement each other. We provide experimental results to verify this argument in Appendix D.

5 Related Work

Few-shot learning. Few-shot learning has recently attracted much attention since the data scarcity has become a bottleneck to many real-world machine learning tasks [39]. Using prior knowledge, few-shot learning can rapidly generalize a pre-trained model to new tasks given only a few examples. One of the most representative frameworks for few-shot learning is MAML [18], where an initialization of parameters is pre-trained to be close to the tasks drawn from the task distribution, and thus a few gradient steps are sufficient to adapt it to a specific task. There are several follow-up studies for MAML. For instance, in [40], vanilla MAML is shown to be sensitive to the network hyperparameters, often leading to training instability. To overcome that, learning the majority of hyperparameters end to end is proposed. In [20], a Bayesian version of MAML is proposed to learn complex uncertainty structure beyond a simple Gaussian approximation. In [41], MAML is revisited under multimodel task distributions. To better adapt to different modes, a modulation network is proposed to identify the mode of the task distribution and then customize the meta-learned prior for the identified mode.

Meta Bayesian optimization. Meta-learning for BO has been discussed in many prior studies. Some of them [42, 43] use a single GP model as the prior of all the tasks. However, in real-world applications, different tasks may have different task-specific features that fail to be captured by a single model. To mitigate the issue, several studies propose to learn an ensemble of GPs as the prior. For instance, Wistuba et al. [16] proposed to use a transferable AF that uses several GPs to evaluate the target dataset then weights expected improvement by the result of the evaluation for fitting in the current task. Feurer et al. [17] proposed to use an ensemble of GP models obtained from each subset of meta-data and use all of them for inference in the new task. But these works still need a sufficiently large amount of data for knowledge transfer to new tasks. MetaBO [38] used policy-based RL to train a neural AF that learns structural properties of a set of source tasks to enable knowledge transfer to related new tasks. FSBO [15] provides a few-shot deep kernel network for a GP surrogate that can quickly adapt its kernel parameters to unseen tasks. However, the benefits of using meta-data for more efficient exploration via few-shot fast adaptation of AFs were not explored by [38, 15].

6 Concluding Remarks

This paper tackles the critical challenge of how to effectively adapt an AF for BO to a wide variety of black-box functions through the lens of few-shot acquisition function (FSAF) learning. One potential limitation of FSAF is the need of a small amount of meta-data. Without any few-shot adaptation, the performance may not always be satisfactory, as shown in Figure 5. Despite this, through extensive experiments, we show that FSAF is indeed a promising general-purpose approach for BO.

References

[1] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, 2012.

[2] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

[3] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.

[4] Yuting Zhang, Kihyuk Sohn, Ruben Villegas, Gang Pan, and Honglak Lee. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 249–258, 2015.

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

[6] Peter I. Frazier and Jialei Wang. Bayesian optimization for materials design. *Springer Series in Materials Science*, page 45–75, Dec 2015. ISSN 2196-2812.

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

[8] 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, 2015.

[9] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of International Conference on Machine Learning*, pages 1015–1022, 2010.

[10] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

[11] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems*, 27:918–926, 2014.

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

[13] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

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

[15] Martin Wistuba and Josif Grabocka. Few-shot bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representation*, 2021.

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

[17] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for Bayesian optimization. *arXiv:1802.02219*, 2018.

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

[19] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

[20] 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.

[21] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep Q-learning algorithms. In *International Conference on Machine Learning*, pages 2021–2030, 2019.

[22] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702, 2019.

[23] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. In *Conference on Uncertainty in Artificial Intelligence*, 2017.

[24] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv:1807.02811*, 2018.

[25] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *=arXiv:1012.2599*, 2010.

[26] Tom Van de Wiele, David Warde-Farley, Andriy Mnih, and Volodymyr Mnih. Q-learning in enormous action spaces via amortized approximate maximization. *arXiv:2001.08116*, 2020.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[28] Qiang Liu and Dilin Wang. Stein variational gradient descent: a general purpose Bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, pages 2378–2386, 2016.

[29] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *International Conference on Machine Learning*, pages 2377–2386, 2016.

[30] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through Bayesian deep Q-networks. In *Information Theory and Applications Workshop (ITA)*, pages 1–9, 2018.

[31] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, pages 4026–4034, 2016.

[32] Yunhao Tang and Alp Kucukelbir. Variational deep Q network. *arXiv:1711.11225*, 2017.

[33] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

[34] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361, 2017.

[35] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

[36] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.

[37] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106, 1964.

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

[39] S. Thrun and L. Pratt. *Learning to Learn*. Springer US, 2012. ISBN 9781461555292. URL `https://books.google.com.tw/books?id=X_jpBwAAQBAJ`.

[40] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *International Conference on Learning Representations*, 2018.

[41] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J. Lim. Multimodal model-agnostic meta-learning via task-aware modulation, 2019.

[42] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. *Advances in Neural Information Processing Systems*, 26:2004–2012, 2013.

[43] Dani Yogatama and Gideon Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1077–1085, 2014.

[44] Vadim Arzamasov, Klemens Böhm, and Patrick Jochem. Towards concise models of grid stability. In *IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6, 2018.

[45] Victor Basu. Prediction of asteroid diameter with the help of multi-layer perceptron regressor., 2019.

[46] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv:1803.02999*, 2018.

[47] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33, 2020.

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] The abstract states that the main contribution is the design of a few-shot acquisition function. Please see Section 3 for the design and see Section 4 for the experimental results.
   (b) Did you describe the limitations of your work? [Yes] Please see Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (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? [N/A]
   (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] Please see the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see Section 4 and Appendices A-B for details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Median and percentiles are provided either using shaded areas or tables. Please see the results in Section 4 and Appendix D.
   (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)? Please see Appendix A for the description of computing resources. [Yes]
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] Please see the code in the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Please see the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Please see the supplementary material.
   (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]

Appendix

A Detailed Training Configuration of FSAF

Training tasks of FSAF. To construct a diverse collection of tasks for the training of FSAF, we leverage GP functions with RBF, Matèrn-3/2, and spectral mixture kernels to capture smooth functions, functions with abrupt local variations, and functions of periodic nature, respectively. For both the RBF and the Matèrn-3/2 kernels, we consider three possible ranges of lengthscales, including $[0.07, 0.13]$, $[0.17, 0.23]$, $[0.27, 0.33]$. For the spectral mixture kernels, we consider mixtures of two Gaussian components of periods 0.3 and 0.6 and three possible ranges of the lengthscales, including $[0.27, 0.33]$, $[0.47, 0.53]$, $[0.57, 0.63]$. As a result, there are nine candidate tasks in the task collection $\mathcal{T}$. In each training iteration i, three out of the nine tasks are selected uniformly at random from the above task collection (and hence $|\mathcal{T}_i| = 3$ in Algorithm 1). The input domain of these GP functions is configured to be $[0, 1]^3$. To facilitate the training procedure, we discretize the continuous input domain by using the Sobol sequence to generate a grid on which the GP functions and the AFs are evaluated, as typically done in BO.

Demo policy for the prior of Bayesian DQN. Recall from Section 3.2 that we leverage a Bayesian variant of DQN and construct an informative prior using a demo policy induced by an off-the-shelf AF. For the demo policy, we use EI, which is computationally efficient and achieves moderate regrets in most of the cases. To control how much the demo policy involves in the training, we use a hyperparameter termed *demonstration ratio* κ, which is implemented by randomly choosing whether to use a mini-batch of transitions from the demo replay buffer with probability κ at each SVGD step. In our experiments, we find that a small demonstration ratio of $\frac{1}{128}$ is sufficiently effective.

Network architecture of each DQN particle. For all the DQN particles used in the experiments, we adopt the standard dueling network architecture [48], where one value network and an advantage network are maintained to produce the estimated Q-values. For a fair comparison between FSAF and MetaBO, both the value network and the advantage network of our FSAF are configured to have 4 fully-connected hidden layers with ReLU activation functions and 200 hidden units per layer. As described in Section 3.1, the input of an advantage network consists of a four-tuple, namely the posterior mean $\mu_t(x)$, the posterior standard deviation $\sigma_t(x)$, the best observation so far y_t^*, and the ratio between the current timestamp and total sampling budget $\frac{t}{T}$. Accordingly, the input of a value network consists of y_t^* and $\frac{t}{T}$.

Computing Resources. All the training and testing processes are run on a Linux server with (i) an Intel Xeon Gold 6136 CPU operating at a maxinum clock rate of 3.7 GHz, (ii) a total of 256 GB memory, and (iii) an RTX 3090 GPU.

Table 1 summarizes the hyperparameter configuration of the training of FSAF.

Table 1: FSAF training hyperarameters.

Description	Value
Batch size (i.e., $\lvert\mathcal{D}_{\tau,k}^{\text{tr}}\rvert$ in Algorithm 1)	128
Target update interval (in terms of iterations)	5
Lower-level learning rate (i.e., η in (8))	0.01
Upper-level learning rate (i.e., β in Algorithm 1)	0.001
Agent/demo replay buffer size	1000
Discount factor γ	0.98
Number of DQN particles	5
Total sampling budget	100
Cardinality of the Sobol grid	200

B Experiment Details

B.1 Experiment Details of Figure 1

Recall that Figure 1 illustrates the overfitting issue of DQN+MAML. Regarding the DQN used for Figure 1, we use the same design, network architecture, and training tasks as those described in

Appendix A. Regarding the vanilla MAML for Figure 1, we use the squared TD error in (4) as the loss function for both the lower-level and upper-level updates. For both training and the fast adaptation during testing, we apply 5-shot adaptation (i.e., 5 black-box functions are used for adaptation) and set the number of few-shot gradient updates to be 5. In Figures 1(b)-1(d), we report the empirical average simple regret as well as the empirical standard deviation over 100 independent trials.

B.2 Experiment Details of Figure 2

Recall that Figure 2 shows the simple regret performance of the AFs under various standard optimization benchmark functions. The input domain of each benchmark function is a hypercube in the form of $[-x_{\lim}, x_{\lim}]^d$ (e.g., $x_{\lim} = 5$ and $d = 2$ for the Ackley function). Moreover, as the function values of these benchmark functions can be one or more orders of magnitude different from each other, for ease of comparison, we scale all the values of the functions to the range $[-2, 2]$. For the posterior inference required by all the AFs, we use a GP with an RBF kernel as the surrogate model. For each benchmark function, the lengthscale parameter of the RBF kernel is estimated via marginal likelihood maximization, which is a commonly-used Bayesian model selection framework.

Validation datasets and testing datasets. As mentioned in Section 4, we construct the validation datasets (mainly for the few-shot adaptation of FSAF and the fine-tuning of MetaBO-T) and the testing datasets by applying random translations and re-scalings to the x and the y values, respectively. Specifically, the amount of translation added to each dimension of x is selected from the range $[-0.1x_{\lim}, 0.1x_{\lim}]$ uniformly at random. Similarly, the re-scaling factor applied to each y value is chosen uniformly at random from the range $[0.9, 1.1]$.

Maximization of the AFs. Recall that the input domain of each optimization benchmark function is a hypercube. To address the continuous input domains and achieve a fair comparison between FSAF and MetaBO, we leverage the hierarchical gridding method similar to that in [38] for the maximization procedure of the AFs. Specifically, we first construct a coarse Sobol grid of N_{coarse} points that span over the entire domain and then evaluate the AF on this grid. Next, we find the N_{m} maximal evaluations on this coarse grid and build a finer local Sobol grid of N_{local} points for each of the N_{m} maximal points. Then, the maximum of the AF is approximated by the maximum value among these $N_{\text{m}}N_{\text{local}}$ AF evaluations. To finish the testing process within a reasonable amount of time, we choose $N_{\text{coarse}} = 2000$, $N_{\text{m}} = 10$, and $N_{\text{local}} = 1000$.

B.3 Experiment Details of Figure 3

Recall that Figure 3 demonstrates the regret performance under the test functions obtained from a variety of real-world datasets. Below we describe these real-world datasets in more detail.

- **XGBoost Hyperparameter Optimization.** We use the HPOBench dataset[3] for the hyperparameter optimization for the XGBoost algorithm. Specifically, we use the pre-computed results of XGBoost with six tunable hyperparameters (e.g., learning rate, L_1 and L_2 regularization terms, and subsampling ratios) on 48 classification datasets, each of which is associated with 1000 randomly selected hyperparameter configurations. Hence, these 48 subsets of pre-computed results naturally provide 48 black-box test functions for BO. In the 1-shot setting, we use 1 out of the 48 black-box functions for fast adaptation of FSAF and finding the lengthscale parameter of the GP surrogate model via marginal likelihood maximization. The remaining 47 black-box functions are used only for testing.
- **Electrical Grid Stability Dataset.** This dataset corresponds to an augmented version of the *Electrical Grid Stability Simulated Dataset*[4] [44]. This dataset records total 12 features, such as the reaction times of the producer and the consumer, power balance, and price elasticity coefficient, and the objective is to maximize the grid stability. This dataset contains 60000 parameter configurations, and we divide them into 39 testing sets and 1 validation set. The validation set is used for both few-shot adaptation of FSAF as well as finding the lengthscale parameter of the GP surrogate model for posterior inference.

[3] Created by AutoML and available at `https://github.com/automl/HPOBench`.

[4] Created by Vadim Arzamasov (Karlsruher Institut für Technologie, Karlsruhe, Germany) and available at `https://www.kaggle.com/pcbreviglieri/smart-grid-stability`.

- **Air Quality Prediction.** We use the meteorological monitoring data of air quality collected in northern Taiwan in 2015[5]. We select 14 air quality features, such as the amount of Sulfur dioxide, Carbon monoxide, and ozone, and set the amount of $PM_{2.5}$ as the objective function. After cleaning up all the NaN entries and missing data entries, we get about 73000 parameter configurations and split them into 29 testing sets and 1 validation set. Again, the validation set is used for both few-shot adaptation of FSAF as well as finding the lengthscale parameter of the GP surrogate model for posterior inference.
- **Oil Well Dataset.** We use the *NYS Oil, Gas, Other Regulated Wells datasets*[6] to find the deepest drilled depth among the oil wells. For each data entry, we use the longitude and latitude of both the surface as well as the bottom of the oil well as the input features. This dataset contains about 41000 parameter configurations, and we divide the dataset into 29 subsets of testing data and 1 set of validation data for few-shot adaptation of FSAF and finding the lengthscale parameter of the GP surrogate model for posterior inference.
- **Asteroid Dataset.** This dataset[7] contains a variety features of the asteroids, and our goal is to find the maximum asteroid diameter based on all of its 12 numerical features. Since the range of the asteroid diameters is too large, we apply the commonly-used input warping technique in BO and use the logarithm of the diameter values as the objective. After cleaning up all the NaN entries and missing entries, we get about 136000 parameter configurations and split them into 39 testing sets and 1 validation set.

Regarding the maximization of AFs, since the input domains of the real-world test functions are all discrete, the maximum value of each AF can be found exactly and therefore the hierarchical gridding procedure is not required in this case.

C Pseudo Code and Architecture of the FSAF Training Algorithm

For completeness, we provide the pseudo code of the training algorithm of FSAF in the following Algorithm 1. Figure 6 further illustrates the demo policy and the replay buffers used in FSAF.

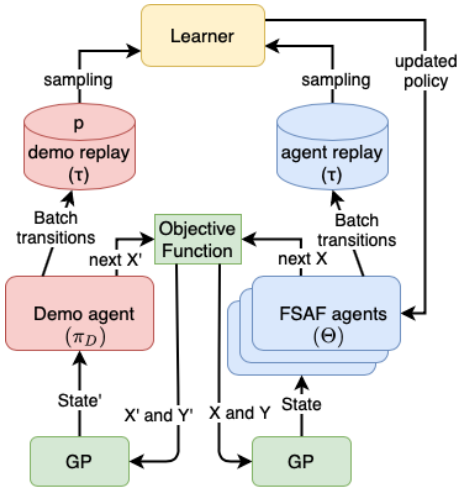


Figure 6: Architecture of the proposed FSAF.

[5]Created by Environmental Protection Administration, Executive Yuan, R.O.C. (Taiwan), available at `https://airtw.epa.gov.tw/ENG/default.aspx`

[6]Hosted by State of New York, available at `https://www.kaggle.com/new-york-state/nys-oil,-gas,-other-regulated-wells?select=oil-gas-other-regulated-wells-beginning-1860.csv`.

[7]Provided by the paper titled "Prediction of Asteroid Diameter with the help of Multi-layer Perceptron Regressor" in *International Conference on Computer Science, Industrial Electronics* in 2019 [45] and available at `https://www.kaggle.com/basu369victor/prediction-of-asteroid-diameter`.

Algorithm 1 FSAF Training Algorithm

Initialize: N instances of Q-networks with parameters $\Theta = \{\theta^{(n)}\}_{n=1}^{N}$, demo policy π_{D}, a collection of candidate tasks $\mathcal{T}$, and the parameters K, S for computing the meta-loss
for each iteration $i = 0, 1, \cdots$ **do**
  Sample a batch of tasks $\mathcal{T}_i$ from $\mathcal{T}$
  for each task $\tau \in \mathcal{T}_i$ **do**
    Generate black-box functions of task τ from GP and set $\Theta_{\tau,0} = \Theta$
    for $k = 0, \cdots, K-1$ **do**
      Collect trajectories $\{(s_0, a_1, r_1, \cdots)\}$ using $\pi_{\Theta_{\tau,k}}$ and store transitions in $\mathcal{R}_Q$
      Collect trajectories $\{(s_0, a_1, r_1, \cdots)\}$ using π_{D} and store transitions in $\mathcal{R}_{\mathrm{D}}$
      Sample a mini-batch of transitions $\mathcal{D}^{\mathrm{tr}}_{\tau,k}$ from the replay buffers $\mathcal{R}_Q, \mathcal{R}_{\mathrm{D}}$
      $\Theta_{\tau,k+1} = \mathcal{M}_{\mathrm{SVGD}}(\Theta_{\tau,k}, \mathcal{D}^{\mathrm{tr}}_{\tau,k})$
    end for
    Set $\Theta^{*}_{\tau,0} = \Theta_{\tau,K}$
    for $s = 0, \cdots, S-1$ **do**
      Sample a mini-batch of transitions $\mathcal{D}^{\mathrm{val}}_{\tau,s}$ from the replay buffers $\mathcal{R}_Q, \mathcal{R}_{\mathrm{D}}$
      $\Theta^{*}_{\tau,s+1} = \mathcal{M}_{\mathrm{SVGD}}(\Theta^{*}_{\tau,s}, \mathcal{D}^{\mathrm{val}}_{\tau,s})$
    end for
  end for
  $\Theta \leftarrow \Theta - \beta \nabla_{\Theta} \big(\sum_{\tau \in \mathcal{T}_i} \mathcal{L}_{\mathrm{meta}}(\Theta; \tau) \big)$
end for

D Additional Experimental Results

In this section, we provide additional experimental results regarding the synthetic GP test functions, the combination of FSAF and FSBO, and the percentiles associated with Figure 3 in the main text.

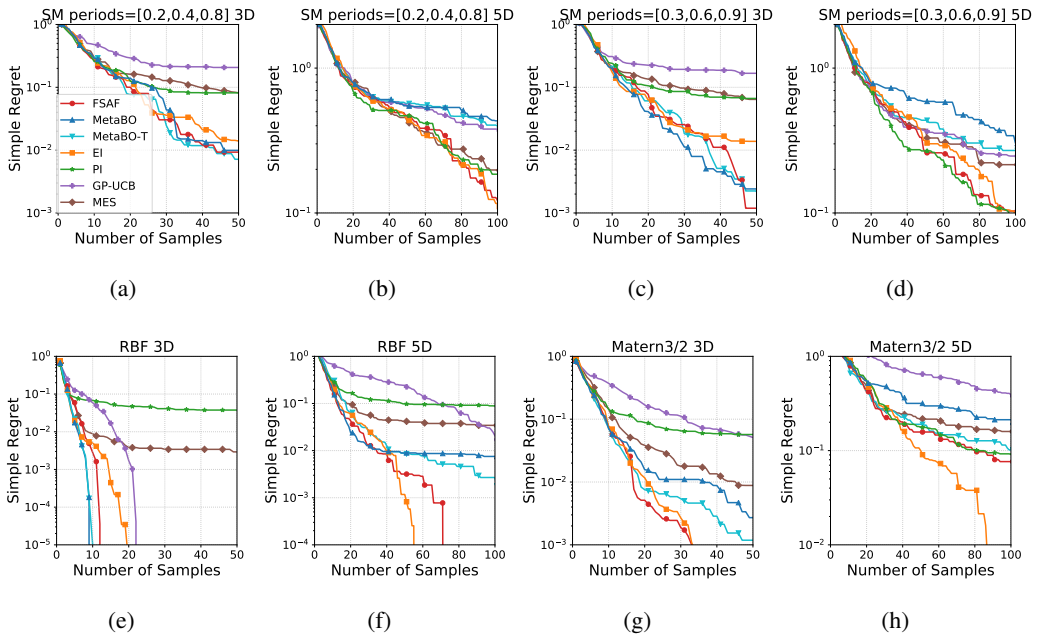


Figure 7: Median simple regrets of FSAF and other benchmark methods under synthetic GP test functions. Other percentiles are provided in Table 2 for visual clarity.

D.1 Synthetic GP Test Functions

To further validate the effectiveness of FSAF, we evaluate FSAF and the benchmark AFs on GP functions drawn from various kernels. Specifically, to generate the validation and testing datasets, we consider four different types of GP kernels for both the input domains of $[0, 1]^3$ and $[0, 1]^5$, including: (i) Spectral mixture kernel with three Gaussian components of periods 0.2, 0.4, and 0.8 as well as a lengthscale range $[0.5, 0.55]$; (ii) Spectral mixture kernel with three Gaussian components of periods 0.3, 0.6, and 0.9 as well as a lengthscale range $[0.5, 0.55]$ (iii) RBF kernel with a lengthscale range $[0.5, 0.55]$; (iv) Matèrn kernel with a lengthscale range $[0.5, 0.55]$. Notably, all the above kernels have never been seen by FSAF during training. To address the continuous input domains, we use the same hierarchical gridding method as described in Appendix B.2 to maximize the AFs. Similar to the case of Figure 2, we consider 5-shot adaptation for FSAF and use the same amount of meta-data for MetaBO-T.

Figure 7 shows the simple regrets of all the AFs under eight different types of GP functions. Again, we can see that FSAF is constantly among the best of all the methods under most of the test functions. We also observe that MetaBO and MetaBO-T perform quite well in the 3-dimensional tasks. We conjecture that this is due to the fact that the pre-trained MetaBO model was originally trained on 3-dimensional GP functions, as described in [38]. However, MetaBO and MetaBO-T do not adapt well to the 5-dimensional GP functions, as shown in Figures 7(b), 7(d), 7(f), and 7(h). By contrast, despite that FSAF is also trained on 3-dimensional GP functions, FSAF adapts more effectively to GP functions of a higher input dimension, as shown in 7(b), 7(d), 7(f). On the other hand, Figure 7(h) shows that FSAF does not catch up well with EI after 40 samples under the 5-dimensional Matèrn-2/3 functions. We conjecture that this is due to the fact that 5-dimensional Matèrn-2/3 functions have even more local variations that could not be captured well by only a small amount of meta-data. Moreover, among the benchmark methods, the best-performing AF still varies under different types of functions. This again corroborates the commonly-seen phenomenon and our motivation.

D.2 Combining FSAF with FSBO

Recall from Remark 6 that [15] proposes FSBO, which leverages meta-data to fine-tune the initialization of the GP kernel parameters for the off-the-shelf AFs. As FSBO can provide better GP kernel parameters for posterior inference than the conventional approaches (e.g., marginal likelihood maximization), it appears feasible to let FSAF and FSBO complement each other and even combine other off-the-shelf AFs with FSBO for better regret performance. In this section, we provide experimental results to validate the above argument. Since [15] did not release their source code, we need to re-implement FSBO by ourselves[8]. Specifically, for the meta-learning part of FSBO, we leverage Reptile [46] with a spectral mixture kernel as a lightweight variant of [47]. Moreover, we use cosine annealing outer-loop learning rate from 10^{-3} to 10^{-5} and set the inner-loop learning rate to be 10^{-2}. The deep kernel is represented by a neural network with two hidden layers (with 128 hidden units per layer), and the degree of few-shot deep kernel learning is configured to be 4. The spectral mixture kernel is configured to have 10 components. We test (i) the combination of FSAF and FSBO as well as (ii) the combination of EI and FSBO on the XGBoost hyperparameter optimization task described in Appendix B.3. As FSBO requires some data for obtaining an initial model, we use 5 out of the 48 subsets of the HPOBench dataset to obtain an initial FSBO model and evaluate the regret performance on the other 43 subsets. From Figure 8 and Table 3, we see that FSBO can slightly improve the regret performance of the two AFs.

D.3 Additional Percentiles for Figure 3

In this section, we provide more detailed percentiles associated with the experiments of Figure 3 in Table 4. We can see that FSAF is still among the best in terms of either lower or higher percentiles under most of the real-world test functions.

[8] We contacted the authors of [15] via email and got the reply that they still needed a bit more time before making the code publicly available. Moreover, we also confirmed the important design choices with the authors to better reproduce FSBO.

Table 2: The percentiles of simple regrets over 100 independent trials for all the AFs under synthetic GP test functions. For 3-dimensional kernel functions, we report the percentiles measured at steps 25 and 50. For 5-dimensional kernel functions, we report the percentiles measured at steps 50 and 100. All the values displayed here are scaled by 100 for more compact notations. The best of each percentile under each type of functions is highlighted in **bold and underline**.

AF	P	SM periods= [0.2,0.4,0.8] 3D	SM periods= [0.3,0.6,0.9] 3D	RBF 3D	Matèrn-3/2 3D
FSAF	25%	0.03, **0.00**	0.01, **0.00**	**0.00**, **0.00**	**0.00**, **0.00**
	50%	7.80, 0.91	3.28, **0.12**	**0.00**, **0.00**	**0.25**, 0.03
	75%	41.62, 38.81	**24.78**, 16.74	**0.00**, **0.00**	**3.87**, 0.54
	90%	**70.97**, 67.15	64.65, 62.63	**0.55**, 0.52	16.75, 9.67
MetaBO	25%	**0.00**, **0.00**	**0.00**, **0.00**	**0.00**, **0.00**	0.02, **0.00**
	50%	10.88, 0.99	**2.19**, 0.24	**0.00**, **0.00**	1.10, 0.27
	75%	46.07, 29.76	28.96, 17.16	0.19, 0.13	11.51, 5.77
	90%	81.47, 75.46	72.62, 61.25	0.92, 0.84	25.12, 22.52
MetaBO-T	25%	0.09, **0.00**	0.15, **0.00**	**0.00**, **0.00**	**0.00**, **0.00**
	50%	6.47, **0.72**	3.97, 0.22	**0.00**, **0.00**	0.65, 0.12
	75%	46.25, 16.45	41.52, 18.71	0.25, 0.02	7.04, 4.93
	90%	82.98, 51.48	78.32, 59.29	1.41, 0.84	24.48, 23.85
EI	25%	1.02, 0.50	0.93, 0.34	**0.00**, **0.00**	0.00, **0.00**
	50%	**4.57**, 1.22	2.60, 1.20	**0.00**, **0.00**	0.36, **0.00**
	75%	49.63, **6.27**	29.60, **4.19**	0.33, 0.02	5.31, **0.43**
	90%	88.47, 48.41	83.11, 38.15	1.22, 0.99	14.93, **1.38**
PI	25%	5.93, 3.96	4.55, 3.54	1.06, 1.01	3.94, 3.21
	50%	10.34, 8.16	8.76, 6.65	4.59, 3.76	7.26, 5.70
	75%	40.42, 14.74	28.49, 12.45	8.04, 6.71	12.61, 9.13
	90%	79.30, 42.57	74.08, 33.11	11.45, 8.93	19.77, 13.93
GP-UCB	25%	13.04, 11.71	11.18, 10.80	**0.00**, **0.00**	6.06, 1.79
	50%	23.83, 20.70	20.46, 16.71	**0.00**, **0.00**	13.20, 5.11
	75%	47.45, 33.06	35.04, 27.53	0.21, **0.00**	26.47, 9.42
	90%	73.17, 47.62	**59.32**, 34.97	1.11, **0.44**	41.64, 16.09
MES	25%	5.70, 3.70	5.72, 2.66	0.03, 0.02	0.69, 0.21
	50%	14.97, 8.18	10.69, 6.42	0.37, 0.29	3.05, 0.88
	75%	**37.36**, 16.17	28.17, 10.70	1.81, 1.18	7.59, 2.88
	90%	75.31, **36.15**	67.86, **17.39**	5.03, 3.75	**11.78**, 6.27

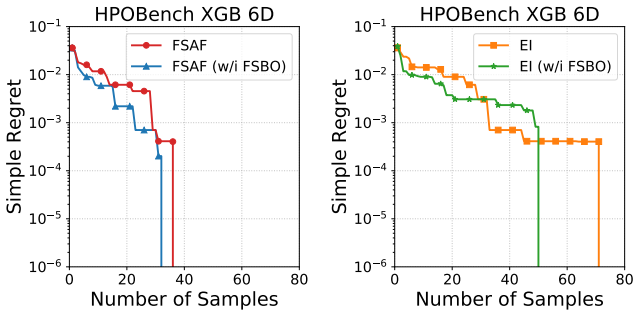
AF	P	SM periods= [0.2,0.4,0.8] 5D	SM periods= [0.3,0.6,0.9] 5D	RBF 5D	Matèrn-3/2 5D
FSAF	25%	8.10, 0.88	5.51, 1.75	**0.00**, **0.00**	2.67, **0.00**
	50%	46.04, 12.03	**25.44**, 10.08	0.31, **0.00**	15.40, 7.47
	75%	**67.89**, **46.69**	74.85, 41.58	**2.67**, **1.32**	45.85, 23.46
	90%	**104.46**, **71.09**	**98.25**, 78.89	**5.52**, **4.00**	72.67, 52.40
MetaBO	25%	22.55, 13.76	14.32, 6.84	**0.00**, **0.00**	9.18, 3.44
	50%	54.93, 43.24	58.90, 30.21	0.89, 0.75	31.69, 21.27
	75%	80.30, 69.47	101.54, 78.60	7.02, 4.92	69.30, 48.99
	90%	128.87, 96.96	156.15, 124.84	30.42, 17.31	86.73, 80.38
MetaBO-T	25%	23.23, 15.84	12.31, 5.69	**0.00**, **0.00**	4.82, 0.61
	50%	57.05, 39.99	43.67, 27.00	0.85, 0.27	19.01, 10.17
	75%	88.13, 69.88	75.39, 65.34	4.58, 3.12	55.67, 28.54
	90%	123.23, 91.40	107.24, 92.93	23.03, 8.77	77.27, 55.63
EI	25%	**6.35**, **0.00**	**3.12**, **0.00**	**0.00**, **0.00**	**0.51**, **0.00**
	50%	**38.09**, **10.12**	27.56, **8.42**	**0.10**, **0.00**	**8.68**, **0.00**
	75%	79.77, 51.62	86.52, 57.03	3.39, 1.59	**28.38**, **7.36**
	90%	143.68, 92.64	123.61, 102.67	8.09, 5.56	68.32, 30.38
PI	25%	12.74, 4.36	6.76, 2.45	5.81, 4.80	8.54, 4.42
	50%	46.90, 18.52	26.81, 9.55	9.76, 8.80	17.61, 9.13
	75%	76.23, 53.18	74.40, **40.77**	14.47, 12.75	30.70, 17.44
	90%	110.59, 89.79	107.18, **70.79**	20.94, 16.77	**53.25**, **27.73**
GP-UCB	25%	24.86, 17.37	21.25, 15.13	12.79, **0.00**	41.12, 19.84
	50%	54.77, 37.94	37.17, 24.71	24.31, 2.06	64.86, 39.35
	75%	88.69, 64.84	**71.08**, 46.74	37.09, 7.72	84.72, 56.06
	90%	127.81, 92.97	105.94, 71.43	46.44, 13.06	111.00, 68.07
MES	25%	19.37, 4.40	11.08, 5.16	0.84, 0.43	9.31, 4.46
	50%	42.62, 19.73	37.91, 21.55	4.17, 3.43	21.72, 15.95
	75%	74.29, 46.93	73.01, 50.09	11.76, 9.91	40.49, 28.11
	90%	109.83, 72.50	98.32, 85.27	18.17, 16.03	67.14, 59.04

Table 3: The percentiles of simple regrets under the HPOBench dataset at steps 30 and 60. All the values displayed here are scaled by 100 for more compact notations. The best of each percentile under each type of functions is highlighted in **bold and underline**.

Dataset	Percentile	FSAF	FSAF (w/i FSBO)
HPOBench XGB	25%	**0.00**, **0.00**	**0.00**, **0.00**
	50%	**0.07**, **0.00**	**0.07**, **0.00**
	75%	1.79, **0.90**	**0.99**, 0.99
	90%	4.51, **3.31**	**3.37**, 3.37
Dataset	Percentile	EI	EI (w/i FSBO)
HPOBench XGB	25%	**0.00**, **0.00**	**0.00**, **0.00**
	50%	**0.31**, 0.04	**0.31**, **0.00**
	75%	2.50, 1.99	**1.57**, **1.13**
	90%	6.36, 6.27	**5.18**, **4.62**

Table 4: The percentiles of simple regrets for all the AFs under different real-world test functions. For PM2.5, and Electrical Grid Stability, we report the percentiles measured at steps 60 and 120. For HPOBench XGB, Oil, and Asteroid, we report the percentiles measured at steps 30 and 60. All the values displayed here are scaled by 100 for more compact notations. The best of each percentile under each type of functions is highlighted in **bold and underline**.

AF	P	Electrical Grid Stability	PM2.5	HPOBench XGB	Oil	Asteroid
FSAF	25%	**0.00**, **0.00**	**0.00**, **0.00**	**0.00**, **0.00**	**0.00**, **0.00**	**0.00**, **0.00**
	50%	**5.28**, 0.60	**11.00**, **6.00**	**0.14**, **0.00**	**16.59**, **0.00**	2.85, **0.00**
	75%	12.16, 7.00	**19.00**, **15.00**	3.16, 2.03	40.82, 16.59	11.16, **0.00**
	90%	16.76, **12.20**	37.00, **27.60**	8.42, 6.29	50.89, **22.89**	18.90, **0.00**
MetaBO	25%	2.42, **0.00**	20.00, 9.00	**0.00**, **0.00**	**0.00**, **0.00**	16.66, **0.00**
	50%	7.43, 2.70	26.00, 17.00	1.38, 0.28	20.28, 1.96	40.58, 10.54
	75%	11.15, **6.86**	39.00, 26.00	**3.15**, **1.51**	43.34, 25.69	50.32, 19.76
	90%	17.59, 13.61	44.40, 35.00	**6.62**, 5.79	60.52, 37.25	58.37, 35.71
MetaBO-T	25%	2.81, **0.00**	23.00, 12.00	**0.00**, **0.00**	15.96, **0.00**	**0.00**, **0.00**
	50%	6.86, 3.74	30.00, 16.00	1.38, 0.61	31.71, **0.00**	2.58, **0.00**
	75%	12.17, 9.72	39.00, 28.00	3.46, 2.47	49.05, 15.75	22.93, **0.00**
	90%	14.76, 13.89	43.60, 36.00	8.27, 6.62	63.61, 31.67	32.49, **0.00**
EI	25%	0.30, **0.00**	12.00, 5.00	**0.00**, **0.00**	15.10, **0.00**	**0.00**, **0.00**
	50%	5.72, **0.00**	16.00, 15.00	0.60, 0.04	25.60, **0.00**	**0.00**, **0.00**
	75%	**9.84**, 7.12	29.00, 21.00	3.16, 2.70	43.43, **15.63**	8.73, **0.00**
	90%	**14.66**, 12.72	**36.00**, 33.60	8.03, 8.03	63.95, 25.83	21.64, **0.00**
PI	25%	**0.00**, **0.00**	6.00, **0.00**	**0.00**, **0.00**	0.27, **0.00**	**0.00**, **0.00**
	50%	5.72, 1.30	17.00, 8.00	0.60, 0.03	18.76, 1.66	2.85, **0.00**
	75%	**9.84**, 7.30	33.00, **15.00**	3.16, 2.14	29.18, 15.75	17.60, **0.00**
	90%	15.31, 12.48	37.40, 32.20	8.03, **4.92**	53.95, 27.49	23.13, **0.00**
GP-UCB	25%	1.49, **0.00**	11.00, 8.00	**0.00**, **0.00**	14.11, **0.00**	**0.00**, **0.00**
	50%	8.05, 2.87	18.00, 15.00	0.60, 0.04	21.54, 13.27	**0.00**, **0.00**
	75%	14.55, 9.84	28.00, 20.00	3.16, 2.53	30.98, 21.83	**4.84**, **0.00**
	90%	17.93, 12.83	38.20, 30.40	8.03, **4.92**	49.56, 30.39	20.66, 1.97
MES	25%	1.49, **0.00**	11.00, 8.00	**0.00**, **0.00**	1.95, **0.00**	**0.00**, **0.00**
	50%	6.81, 2.89	15.00, 14.00	0.61, 0.04	20.28, 1.95	1.36, **0.00**
	75%	13.61, 9.17	32.00, 16.00	3.16, 2.73	**28.83**, 20.93	7.62, **0.00**
	90%	16.83, 13.22	37.00, 37.00	8.03, 6.29	**33.85**, 30.42	**14.82**, 3.12



(a) (b)

Figure 8: Median simple regrets of FSAF and EI, with and without the integration with FSBO. Other percentiles are in Table 3.

E A Summary of Popular Acquisition Functions for Bayesian Optimization

Bayesian optimization via myopic AFs. In BO, AFs are usually designed to reflect the trade-off between exploration and exploitation of the global optima through maximizing some cheap-to-evaluate *myopic* objective, which typically corresponds an one-step look-ahead strategy. Such AFs have been designed from various perspectives on how to address the trade-off. For instance, under GP-UCB, AF is designed from the perspective of optimism in the face of uncertainty [9]. Under EI, AF incorporates the expected one-step improvement in terms of the best observation made so far by design [13, 14]. Similarly, under the knowledge gradient method [49], AF is chosen to be the expected improvement in the largest posterior mean over the input domain. Instead of using the expected improvement, PI [37] uses the tail probability of improvement as the AF for BO. Under entropy search methods, AF is the expected reduction in the uncertainty of the global optima [10–12]. Under Thompson sampling, AF values are directly drawn from the posterior predictive distribution [50]. As the AFs are often handcrafted according to different perspectives of the exploration-exploitation trade-off, the performance of an AF can vary significantly under different types of black-box functions [11]. Thus, designing an AF that can adapt and demonstrate outstanding performance to a wide variety of black-box functions remains a critical challenge. This is the goal of our paper: we propose a reinforced few-shot learning framework, in which an initial AF is learned with fast adaption to various black-box functions given a limited number of samples (e.g., 1-shot).

Bayesian optimization via non-myopic methods. To go beyond myopic AFs, recently there has been some interest in designing non-myopic strategies for BO from the perspective of dynamic programming (DP). For example, [51, 52] characterized the optimal non-myopic strategies for BO. To tackle the intractable DP problem, [53, 54] proposed to approximately solve the DP problem for BO by different simulation techniques. [55] focused on the two-step lookahead AFs for BO and proposed an efficient Monte-Carlo method to find such AFs. Instead of fixing the horizon in advance, [56] proposed a principled way to select the rolling horizon for approximate dynamic programming in BO. [57] proposed to achieve non-myopia by maximizing a lower bound on the multi-step expected utility. [58] proposed a tree-based AF to enable approximate multi-step lookahead in a one-shot manner by leveraging the reparameterization trick. The proposed FSAF is also non-myopic by nature as it implicitly takes multi-step effect into account by using Q-networks as AFs. Different from the above multi-step solutions, FSAF is meant to serve as an AF that can adapt to a wide variety of black-box functions given only a small amount of meta-data.

Additional References for the Appendix

[48] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1995–2003, 2016.

[49] Peter Frazier, Warren Powell, and Savas Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009.

[50] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853, 2017.

[51] Michael A Osborne, Roman Garnett, and Stephen J Roberts. Gaussian processes for global optimization. In *International Conference on Learning and Intelligent Optimization (LION3)*, pages 1–15, 2009.

[52] David Ginsbourger and Rodolphe Le Riche. Towards Gaussian process-based optimization with finite time horizon. In *Advances in Model-Oriented Design and Analysis*, pages 89–96. 2010.

[53] Javier González, Michael Osborne, and Neil Lawrence. GLASSES: Relieving the myopia of Bayesian optimisation. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 790–799, 2016.

[54] Remi R Lam, Karen E Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *Advances in Neural Information Processing Systems*, pages 883–891, 2016.

[55] Jian Wu and Peter Frazier. Practical two-step lookahead bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 9813–9823, 2019.

[56] Xubo Yue and Raed AL Kontar. Why non-myopic bayesian optimization is promising and how far should we look-ahead? a study via rollout. In *International Conference on Artificial Intelligence and Statistics*, pages 2808–2818, 2020.

[57] Shali Jiang, Henry Chai, Javier Gonzalez, and Roman Garnett. BINOCULARS for efficient, nonmyopic sequential experimental design. In *International Conference on Machine Learning*, pages 4794–4803, 2020.

[58] Shali Jiang, Daniel Jiang, Maximilian Balandat, Brian Karrer, Jacob Gardner, and Roman Garnett. Efficient nonmyopic Bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33, 2020.