WORK WHILE THEY SLEEP: EXPLOITING EVALUATION LATENCY FOR FULLY BAYESIAN OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Black-box optimization problems are ubiquitous across science and engineering, often dealing with expensive objective functions. This objective latency has two consequences during optimization: (i) the objective evaluation dominates execution time, and (ii) sample-efficient algorithms are crucial to accelerate development and avoid wasting resources. Bayesian optimization (BO) methods are the *de facto* choice of planners for suggesting the next point to try. Standard BO fits the surrogate model's hyperparameters with a point estimate. Alternatively, a fully Bayesian approach uses model averaging to account for uncertainty over the hyperparameters, leading to better uncertainty estimates—useful in the low-data regime that is pervasive in BO. However, it is often prohibitively expensive and thus rarely used. In this work, we propose ELF-BO, an algorithm that uses the objective evaluation latency to headstart the computation of the next suggestion, allowing for fully Bayesian optimization without incurring substantial decision-time costs. This is done by sampling from the hyperparameter posterior *while* the objective is being evaluated, only requiring reweighting of the samples once the objective value is observed. Across synthetic functions and real-world applications, we show that ELF-BO matches the performance of fully Bayesian methods while only incurring decision latency on par with or better than standard BO. Thus, ELF-BO makes fully Bayesian optimization practical in real-world use cases.

1 INTRODUCTION

Bayesian optimization (BO; Močkus, 1975) often acts as a planner (suggesting the next query point to evaluate) for optimizing black-box functions commonly encountered in science (Greenaway et al., 2023; Muthyala et al., 2026) and engineering (Lam et al., 2018; Ament et al., 2023b). Across domains, most applications of BO deal with expensive objective functions, a property that motivates the development of sample-efficient methods. The inherent high evaluation latency makes it the main bottleneck in the optimization run. Meanwhile, after suggesting a query location, the planner waits idly for the evaluation results before continuing. Crucially, expensive function evaluations also limit applications to cases with restricted data availability, both in initial dataset size and in the number of evaluations allowed in the optimization loop (Jones et al., 1998). The former increases the difficulty of achieving the latter, as a misspecified model can lead to suboptimal queries that waste precious evaluation budget.

Standard BO methods utilize Gaussian processes (GP; Rasmussen & Williams, 2006) as surrogate models and use a point estimate approach—either maximum likelihood estimation (MLE) or maximum a posteriori (MAP)—to find the GP's hyperparameters (e.g., lengthscale, output scale, deep-kernel parameters, and noise variance). However, relying on a single estimate of the hyperparameters can drastically underestimate the uncertainty over the true hyperparameters, and therefore the uncertainty over the objective values (De Ath et al., 2021b). Importantly, this is a bigger problem in the low-data regimes where well-calibrated models are required for sample-efficient decision-making. The same trade-off arises beyond GPs: the high computational overhead of Bayesian neural networks (BNNs; Li et al., 2024; Kristiadi et al., 2024) makes them similarly expensive at decision time.

An alternative to point estimates is the use of a fully Bayesian optimization (Osborne et al., 2009; Benassi et al., 2011), in which the hyperparameters are averaged out according to their posterior dis-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

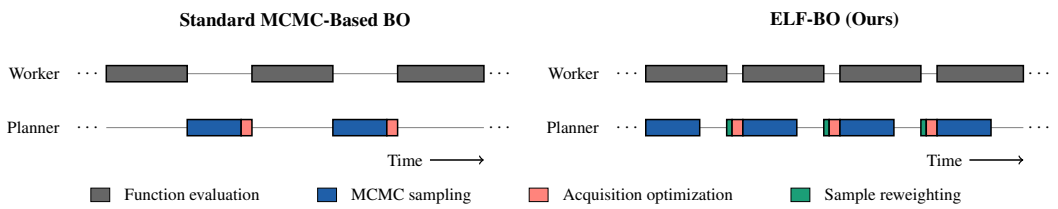

Figure 1: **Where the time goes in a sequential BO campaign.** Each timeline shows the *worker* (the experiment or simulator evaluating f) and the *planner* (the BO algorithm); thick segments denote busy time. **Left:** In standard fully Bayesian BO, the two alternate: the planner waits while f is evaluated, and the worker waits while the planner runs MCMC and optimizes the acquisition function. **Right:** ELF-BO runs MCMC *while* f is being evaluated, so once y_t arrives, only a cheap importance reweighting and the acquisition optimization remain, greatly reducing decision latency.

tribution. In practice, this is done by using Markov chain Monte Carlo (MCMC; Neal, 2011) methods to obtain samples from the posterior distribution and computing the average of the acquisition function over the samples. As usual in Bayesian inference, the price is paid with more computation and time, mostly due to the expensive MCMC sampling, rendering fully Bayesian methods prohibitively expensive for many practitioners. This expensive decision-making step between objective function evaluations is undesirable in various applications; e.g., in automated materials discovery, machine time is valuable, so minimal downtime is desired (MacLeod et al., 2020).

In this work, we show that most of the expensive steps needed in a fully Bayesian approach can be performed while the expensive objective function is being evaluated. We propose **E**valuation **L**atency **F**ully **B**ayesian **O**ptimization (**ELF-BO**), which uses the BO planner's otherwise idle time to reduce the footprint of fully Bayesian optimization after the evaluation result becomes available—see Fig. 1. This is accomplished by showing that an expectation over the next hyperparameter posterior can be computed using a weighted expectation with the posterior of the hyperparameters at the current step. These weights are proportional to the posterior predictive density of the pending evaluation results—thus only the weights are computed once the evaluation is done. Since this identity relies only on Bayes' rule, ELF-BO is not tied to GPs and can be applied to BNNs. In this work, however, we focus on GPs due to their prevalence and simplicity in BO.

We evaluate ELF-BO on 15 standard benchmarks ranging from 2 to 10 dimensions and on three real-world experimental datasets. Across these problems, ELF-BO matches the optimization performance of naïve MCMC-based fully Bayesian BO after every observation, while its decision latency—the time the experiment sits idle waiting for the next query—stays close to that of point-estimate BO. The latter amounts to a decision latency roughly $10\times$ lower than standard fully Bayesian BO with a Matérn kernel, 70–$110\times$ lower with a deep kernel, and more than two orders of magnitude lower as the dataset grows. We further use the effective sample size of the importance weights to diagnose when reweighting is reliable, and characterize the regimes in which ELF-BO yields the largest gains. Altogether, ELF-BO makes MCMC-based fully Bayesian BO practical in real-world scenarios—the more expensive the function evaluation, the more calibrated the planner will be, without high compute overhead.

2 Preliminaries

We consider the maximization problem $\boldsymbol{x}_* \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ where $f : \mathcal{X} \to \mathbb{R}$ is an expensive black-box objective function that can only be accessed via noisy observations $y_i = f(\boldsymbol{x}_i) + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$. Bayesian optimization (BO) provides a principled solution to this problem grounded in Bayesian decision theory. The key components of BO are: a surrogate model $p(f \mid \mathcal{D}_t)$ and an acquisition function $\alpha(\boldsymbol{x}; \mathcal{D}_t)$, where $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^t$ is the set of observations of the objective function up to the current time step $t \in \{1, \ldots, T\}$. The surrogate models the current belief over the objective function given the data observed so far, which the acquisition function then uses to score the value of querying each point next. The next query is chosen as $\boldsymbol{x}_{t+1} \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha(\boldsymbol{x}; \mathcal{D}_t)$, the objective is evaluated at $\boldsymbol{x}_{t+1}$, and the procedure repeats until the evaluation budget is exhausted.

2.1 Gaussian Processes

The standard surrogate choice for BO is the Gaussian process. A Gaussian process $\mathcal{GP}(\mu^{(\boldsymbol{\theta})}, k^{(\boldsymbol{\theta})})$ is a distribution over functions fully characterized by its prior mean function $\mu^{(\boldsymbol{\theta})} : \mathcal{X} \to \mathbb{R}$ and its covariance function $k^{(\boldsymbol{\theta})} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\boldsymbol{\theta}$ collects all hyperparameters of the mean and covariance functions, together with the observation noise variance σ_n^2.

Conditioned on an observation dataset $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^t$ and on the hyperparameters $\boldsymbol{\theta}$, the posterior distribution over function values is also a GP, with posterior mean and covariance given by

$$\mu_t^{(\boldsymbol{\theta})}(\boldsymbol{x}) = \mu^{(\boldsymbol{\theta})}(\boldsymbol{x}) + \kappa^{(\boldsymbol{\theta})}(\boldsymbol{x})^\top (\boldsymbol{K}^{(\boldsymbol{\theta})} + \sigma_n^2 \boldsymbol{I})^{-1} (\boldsymbol{y}_t - \mu^{(\boldsymbol{\theta})}(\boldsymbol{X}_t)) \tag{1}$$

$$k_t^{(\boldsymbol{\theta})}(\boldsymbol{x}, \boldsymbol{x}') = k^{(\boldsymbol{\theta})}(\boldsymbol{x}, \boldsymbol{x}') - \kappa^{(\boldsymbol{\theta})}(\boldsymbol{x})^\top (\boldsymbol{K}^{(\boldsymbol{\theta})} + \sigma_n^2 \boldsymbol{I})^{-1} \kappa^{(\boldsymbol{\theta})}(\boldsymbol{x}'), \tag{2}$$

where $\boldsymbol{K}^{(\boldsymbol{\theta})} = [k^{(\boldsymbol{\theta})}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i=1,j=1}^{t,t} \in \mathbb{R}^{t \times t}$ is the observation covariance matrix, $\kappa^{(\boldsymbol{\theta})}(\boldsymbol{x}) = [k^{(\boldsymbol{\theta})}(\boldsymbol{x}, \boldsymbol{x}_i)]_{i=1}^t \in \mathbb{R}^t$ is the vector of cross-covariances, $\boldsymbol{y}_t = [y_1, \ldots, y_t] \in \mathbb{R}^t$ collects the observed objective values, and $\boldsymbol{X}_t$ collects all observed points. Defining $\sigma_t^{2(\boldsymbol{\theta})}(\boldsymbol{x}) := k_t^{(\boldsymbol{\theta})}(\boldsymbol{x}, \boldsymbol{x})$, the marginal posterior at any point $\boldsymbol{x}$ is $p(f(\boldsymbol{x}) \mid \mathcal{D}_t, \boldsymbol{\theta}) = \mathcal{N}(f(\boldsymbol{x}) \mid \mu_t^{(\boldsymbol{\theta})}(\boldsymbol{x}), \sigma_t^{2(\boldsymbol{\theta})}(\boldsymbol{x}))$. We emphasize that these expressions are conditional on $\boldsymbol{\theta}$; recovering the surrogate $p(f \mid \mathcal{D}_t)$ introduced above requires marginalizing the hyperparameters out, which we discuss in Section 2.2.

2.2 Model Averaging

The posterior in Section 2.1 is conditional on a particular setting of hyperparameters $\boldsymbol{\theta}$; thus, following the prescription of Bayesian inference, we should marginalize out our uncertainty over it, calculating $p(f \mid \mathcal{D}_t) = \int p(f \mid \mathcal{D}_t, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}_t) d\boldsymbol{\theta}$. The problem, however, is that such an integral is intractable. The hyperparameter posterior $p(\boldsymbol{\theta} \mid \mathcal{D}_t) \propto p(\mathcal{D}_t \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$, given a prior $p(\boldsymbol{\theta})$ and the GP marginal likelihood, is not Gaussian, since $\boldsymbol{\theta}$ enters the likelihood nonlinearly through the kernel matrix, its inverse, and its determinant; moreover, its normalizing constant has no closed form.

As a consequence, practitioners rely on a point estimate $\hat{\boldsymbol{\theta}}$ obtained via maximum likelihood or maximum a posteriori. Such practice is the standard in BO, and is equivalent to approximating $p(\boldsymbol{\theta} \mid \mathcal{D}_t)$ with $\delta(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$. Unfortunately, this discards all uncertainty about $\boldsymbol{\theta}$, which is especially bad in low-data regimes prevalent in BO.

Monte Carlo (MC) methods offer a solution to this problem, allowing us to approximate the intractable integral as $p(f \mid \mathcal{D}_t) \approx 1/M \sum_{m=1}^M \mathcal{GP}(f \mid \mu_t^{(\boldsymbol{\theta}_m)}, k_t^{(\boldsymbol{\theta}_m)})$, where $\{\boldsymbol{\theta}^{(m)}\}_{m=1}^M \sim p(\boldsymbol{\theta} \mid \mathcal{D}_t)$. Different Markov chain Monte Carlo (MCMC) algorithms can be used to generate the samples, such as slice sampling (Neal, 2003; Murray & Adams, 2010) or Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 2011) and its adaptive variant NUTS (Hoffman & Gelman, 2014).

Under this mixture surrogate, the marginal acquisition function becomes the corresponding mixture of per-sample acquisitions,

$$\alpha(\boldsymbol{x}; \mathcal{D}_t) = \mathbb{E}_{p(\boldsymbol{\theta} \mid \mathcal{D}_t)}[\alpha(\boldsymbol{x}; \mathcal{D}_t, \boldsymbol{\theta})] \approx \frac{1}{M} \sum_{m=1}^M \alpha(\boldsymbol{x}; \mathcal{D}_t, \boldsymbol{\theta}^{(m)}), \tag{3}$$

provided α is linear in the predictive distribution, so that the marginalization over $\boldsymbol{\theta}$ and the expectation over function values commute. This holds for expected improvement (EI; Jones et al., 1998); for its numerically stable variant Log-EI (Ament et al., 2023a), which we use in this work, we average in EI space and take the logarithm of the result. Other acquisition functions can also be applied but may not reduce to a mixture of per-sample acquisitions; for instance, upper confidence bound (Auer, 2002; Srinivas et al., 2010) is instead computed from the mean and variance of the mixture.

3 Evaluation Latency Fully Bayesian Optimization

Consider a dataset $\mathcal{D}_{t-1}$ has been observed, a point $\boldsymbol{x}_t$ was just dispatched for evaluation, and y_t is pending. Our goal is to start the computations to generate the suggestion as follows:

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_t)}[\alpha(\boldsymbol{x}; \mathcal{D}_t, \boldsymbol{\theta})], \tag{4}$$

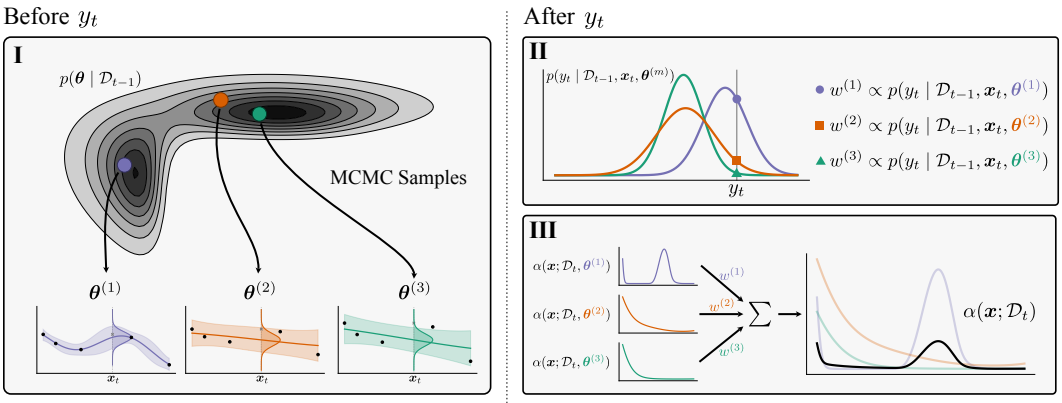

Figure 2: **Overview of ELF-BO.** Fully Bayesian BO marginalizes the acquisition function over the GP hyperparameters $\boldsymbol{\theta}$, which requires MCMC after every new observation. We instead run it while the objective is still being evaluated. *Left:* **(I)** with the candidate $\boldsymbol{x}_t$ dispatched and its value y_t still pending, we sample $\{\boldsymbol{\theta}^{(m)}\}$ from the posterior given the data collected so far, $p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})$, which is already available. Each sample gives a different GP fit, and hence a different posterior at $\boldsymbol{x}_t$. *Right:* **(II)** when y_t arrives, each sample is reweighted by how well it predicted it, $w^{(m)} \propto p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}^{(m)})$, and **(III)** the per-sample acquisition functions $\alpha(\boldsymbol{x}; \mathcal{D}_t, \boldsymbol{\theta}^{(m)})$ are combined into the marginal acquisition $\alpha(\boldsymbol{x}; \mathcal{D}_t)$ used to pick the next candidate.

where $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t, y_t)\}$ and $\boldsymbol{\theta}$ are the GP hyperparameters. The obvious challenge is that y_t, and consequently $\mathcal{D}_t$, are not known yet, so we would have to wait for it to run MCMC to obtain samples $\{\boldsymbol{\theta}^{(m)}\}_{m=1}^M$ needed to compute the acquisition.

The central idea of ELF-BO is that we can use the objective evaluation time to obtain samples $\{\boldsymbol{\theta}^{(m)}\}_{m=1}^M$ from the posterior of the previous step $p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})$ and use a reweighted average to compute (4), as illustrated in Fig. 2. The following proposition justifies this (proof in Appendix A):

Proposition 1. *Let φ be a function of $\boldsymbol{\theta}$ and $p(\boldsymbol{\theta} \mid \mathcal{D}_t)$ be the posterior of hyperparameters at step $t \in \{1, \dots, T\}$. Suppose that, for each $t \geq 1$, $p(\boldsymbol{\theta} \mid \mathcal{D}_t)$ is absolutely continuous w.r.t. $p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})$. Then, for all $t = 1, \dots, T$,*

$$\mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_t)}[\varphi(\boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})}[w(\boldsymbol{\theta})\, \varphi(\boldsymbol{\theta})], \qquad \textit{where} \quad w(\boldsymbol{\theta}) = \frac{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta})}{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t)}.$$

Proof sketch. We use the importance sampling argument with the previous posterior as the proposal, so the importance weight is the ratio of the new posterior to the old one. By Bayes' rule, this ratio is proportional to the likelihood of the new observation under the predictive distribution. □

The preceding result implies that we can start computing the expectation w.r.t. the posterior $p(\boldsymbol{\theta} \mid \mathcal{D}_t)$ at the time step t *before* we receive the actual result for y_t. In particular, we can already start sampling $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})$ of the previous time step $t-1$. This is the expensive part of this expectation computation due to expensive MCMC. Indeed, once y_t is obtained, only simple computation of $w(\boldsymbol{\theta})$, $\varphi(\boldsymbol{\theta})$, and averaging are required.

Notice that Proposition 1 casts the expectation as importance sampling. We can thus use self-normalized importance sampling (Owen, 2013) to approximate $\mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_t)}[\alpha(\boldsymbol{x}; \mathcal{D}_t, \boldsymbol{\theta})]$ using MCMC samples $\{\boldsymbol{\theta}^{(m)}\}_{m=1}^M$ from $p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})$ obtained *before* the evaluation is ready and reweighted *after* y_t is observed:

$$\alpha(\boldsymbol{x}; \mathcal{D}_t) \approx \sum_{m=1}^{M} w^{(m)}\, \alpha(\boldsymbol{x}; \mathcal{D}_t, \boldsymbol{\theta}^{(m)}), \qquad w^{(m)} = \frac{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}^{(m)})}{\sum_{m'} p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}^{(m')})}. \tag{5}$$

This moves all MCMC computations to objective evaluation time. Once the objective value is observed, the acquisition function can be computed by reweighting the sampled hyperparameters based

Algorithm 1 ELF-BO

Input: Initial dataset $\mathcal{D}_0$, acquisition α, number of samples M, evaluation budget T
Output: Final dataset $\mathcal{D}_T$

$\{\boldsymbol{\theta}^{(m)}\}_{m=1}^{M} \leftarrow \texttt{MCMC}(p(\boldsymbol{\theta} \mid \mathcal{D}_0))$
$w^{(m)} \leftarrow 1/M$
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \sum_m w^{(m)} \alpha(\boldsymbol{x}; \mathcal{D}_{t-1}, \boldsymbol{\theta}^{(m)})$
  $h \leftarrow$ **async** `compute`$(f, \boldsymbol{x}_t)$ ▷ evaluation of $f(\boldsymbol{x}_t)$ starts
  $\{\boldsymbol{\theta}^{(m)}\}_{m=1}^{M} \leftarrow \texttt{MCMC}(p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1}))$ ▷ Computed in parallel
  $y_t \leftarrow$ **await** `get_result`(h) ▷ evaluation of $f(\boldsymbol{x}_t)$ ends; obtain y_t
  $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t, y_t)\}$
  Compute $\{w^{(m)}\}$ as in (5)
end for

on how well each explains the observed value. Algorithm 1 summarizes the complete procedure. The reweighting strategy in ELF-BO may resemble sequential Monte Carlo (SMC; Chopin, 2002), but in ELF-BO, a fresh set of samples is generated at each function evaluation, whereas SMC propagates a single population across all steps.

Importantly, as in all importance sampling algorithms, the approximation is only reliable when the proposal $p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})$ adequately covers the target $p(\boldsymbol{\theta} \mid \mathcal{D}_t)$. This can be monitored through the effective sample size of the self-normalizing weights: $M_{\text{eff}} = 1/\sum_{m=1}^{M}(w^{(m)})^2$, which is available immediately after y_t is observed. Values of M_{eff} near M indicate that the value of y_t had high probability under a variety of sampled hyperparameters, so the reweighted samples are representative, while low values indicate that y_t was poorly explained by most of the samples, meaning a few samples dominate and the estimate will collapse to a single point. In Section 6 we analyze the effective sample size of the method across optimization steps.

3.1 Decision Latency Reduction

We now analyze the *decision latency*, the wall-clock time elapsed between an observation arriving and the next query being proposed. This is relevant whenever idle equipment is costly, samples degrade while waiting, or an operator must dispatch the next experiment immediately, prevalent in automated scientific discovery (MacLeod et al., 2020).

Let T_f denote the evaluation latency of the objective, T_{MC} the time to draw M posterior samples, T_α the time to optimize the expected acquisition function, and T_w the time to compute the weights in (5). The decision latencies of standard fully Bayesian optimization and ELF-BO are given by

$$L_{\text{FBO}} = T_{\text{MC}} + T_\alpha \qquad \text{and} \qquad L_{\text{ELF-BO}} = \max(0,\, T_{\text{MC}} - T_f) + T_w + T_\alpha,$$

respectively. Therefore, ELF-BO reduces decision latency by

$$\Delta L = L_{\text{FBO}} - L_{\text{ELF-BO}} = \min(T_{\text{MC}}, T_f) - T_w.$$

Note that in many real-world applications, $T_f \geq T_{\text{MC}}$, so the entire MCMC sampling procedure runs behind the objective evaluation. In such cases, the reweighting overhead is negligible in practice. We validate this analysis in Section 6.

4 Related Work

Fully Bayesian Optimization Early work on GPs for global optimization considered a fully Bayesian approach to model hyperparameters (Osborne et al., 2009; Benassi et al., 2011). For instance, the seminal work of Snoek et al. (2012) demonstrated that an MCMC approach using slice sampling (Neal, 2003; Murray & Adams, 2010) outperforms point estimate-based methods. More recently, De Ath et al. (2021b) analyzed the benefits of fully Bayesian methods in BO, demonstrating that the MCMC approach is superior to MAP and variational inference methods when using EI and

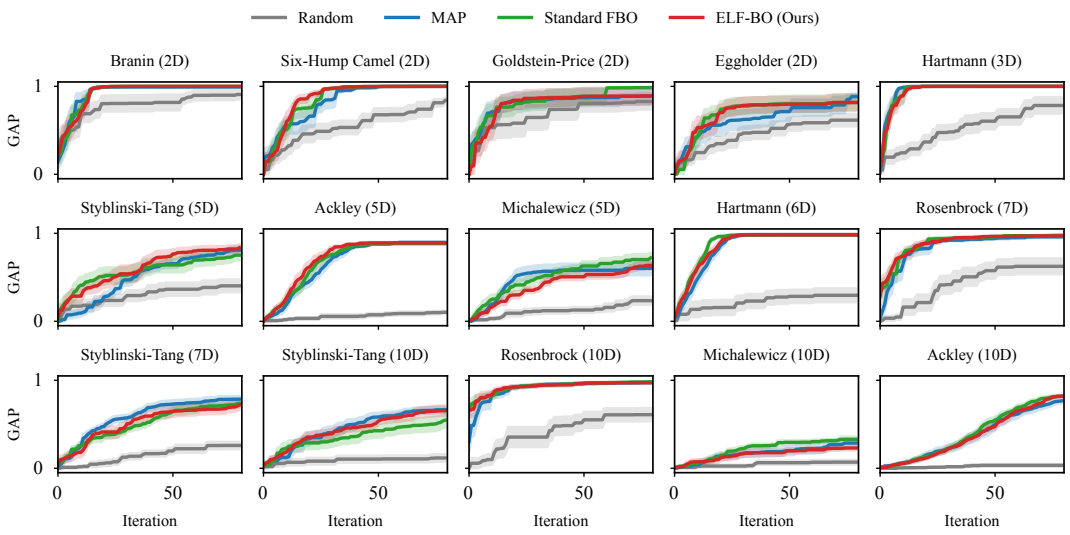

Figure 3: **Standard benchmarks.** Optimization performance of 15 synthetic functions. ELF-BO consistently follows Standard FBO on most functions, in many of which the fully Bayesian approaches achieve earlier convergence than MAP.

automatic relevance determination kernels. In the high-dimensional setting, SAASBO (Eriksson & Jankowiak, 2021) demonstrated the effectiveness of a fully Bayesian method with NUTS (Hoffman & Gelman, 2014) and priors that capture sparseness. Marginalizing out hyperparameters also provides a textbook solution to overfitting and is beneficial in the overparameterized regime of deep kernels (Ober et al., 2021). Beyond GP surrogates, fully Bayesian BO has also been performed with Bayesian neural networks (Li et al., 2024; Kristiadi et al., 2023; 2024), which could equally benefit from ELF-BO.

Asynchronous Batch Bayesian Optimization In this setting, multiple function evaluations run in parallel, and, as soon as one of the function evaluations finishes, a new query is launched without waiting for the pending results. Ginsbourger et al. (2011) was the first to work on this problem, using samples from the predictive posterior at the pending location to calculate the expected EI under different objective realizations. Alvi et al. (2019) proposed the use of local penalization around the pending point to reduce redundant selections, while De Ath et al. (2021a) used an ε-greedy algorithm to achieve the same goal. Alternatively, Kandasamy et al. (2018) proposed the use of Thompson Sampling (Thompson, 1933) in which a new sample of the posterior is taken after including only the newly revealed observation. More recently, Riegler et al. (2026) also demonstrated good results across acquisition functions without explicitly modelling the pending evaluations. On the related problem of stochastically delayed feedback, Verma et al. (2022) considered substituting the minimum function value for observations not yet revealed, a pessimistic version of the Kriging Believer method (Ginsbourger et al., 2010). ELF-BO complements this line of work by operating in settings with a single function evaluation at a time, a scenario often found in real-world applications.

5 Experiments

We now evaluate the empirical performance of ELF-BO. Our experiments aim to answer two questions: (i) Does reweighting hyperparameter samples drawn while the oracle is busy match the performance of running MCMC after each observation? (ii) What is the wall-clock time effect of moving sampling under the objective evaluation?

Baselines. We compare ELF-BO against random search and two baselines sharing the surrogate and acquisition function, so that differences between methods reflect only how the GP hyperparameters are handled and when the compute is spent. Random search samples uniformly from the

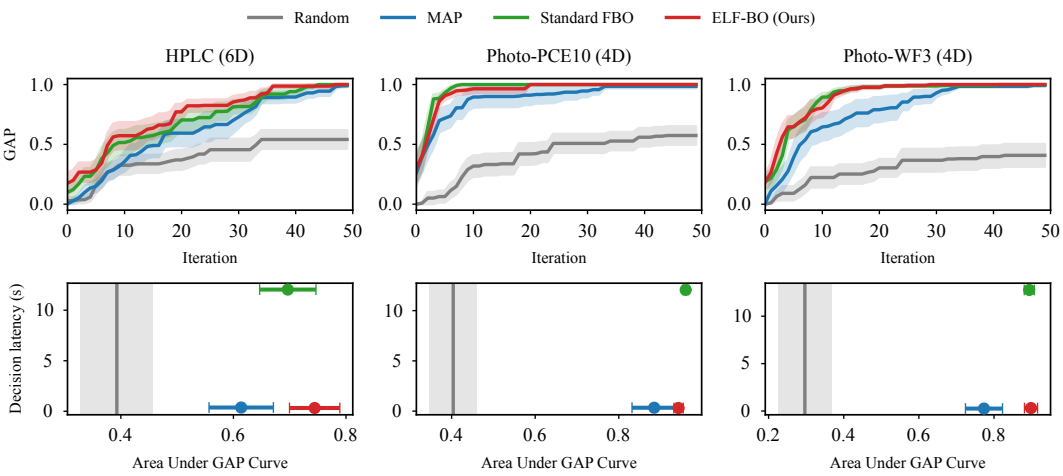

Figure 4: **Real-world tasks.** Results on three experimental datasets from the Olympus suite. *Top:* best GAP found versus iteration, with standard error shaded. ELF-BO converges as quickly and reliably as Standard FBO, while MAP lags and varies more across seeds. *Bottom:* mean decision latency versus area under GAP curve, with error bars showing the standard error over all steps. By sampling while the objective is evaluated, ELF-BO has area under GAP close to Standard FBO's at a latency similar to MAP's, more than 10× lower than Standard FBO.

domain. MAP fits the hyperparameters by maximizing the marginal likelihood plus log prior at every step, which is the default practice. Standard fully Bayesian optimization (Standard FBO) runs NUTS on the full dataset after each observation arrives; it is the fully Bayesian gold standard and pays the entire sampling cost between evaluations. ELF-BO runs the same NUTS sampler on the data available when a query is dispatched, while the oracle is evaluating, and when the observation arrives, it corrects the draws with a single importance reweighting.

Setup. All methods use a GP with a constant mean, homoskedastic Gaussian likelihood, and ARD Matérn-5/2 kernel with output scale. The lengthscale prior is scaled by dimension as proposed by Hvarfner et al. (2024). For the sampling methods, we shift the distribution so that the MAP prior's mode becomes the sampled prior's median, and we reduce the scale from $\sqrt{3}$ to 1. Unless stated otherwise, Standard FBO and ELF-BO use NUTS (Hoffman & Gelman, 2014) with 4 vectorized chains, 512 warmup steps per chain, keeping a total of M samples after thinning by 16 (total of $16M$ samples generated after warmup). We use a dense mass matrix and a max tree depth of 6. Log-EI (Ament et al., 2023a) is used for all methods. For all experiments, we report the results across 10 trials, with standard errors indicated. All methods are implemented in BoTorch (Balandat et al., 2020). Further details on the experimental setup are presented in Appendix B.

Metrics. As different objective functions have different scales, GAP (Barton, 1984) is used to evaluate performance: $\text{GAP} = (y_i^{\max} - y_0^{\max})/(y_* - y_0^{\max})$, where $y_i^{\max}$ is the best observed value at the i-th iteration, $y_0^{\max}$ is the highest initial value, and y_* is the optimal value. To evaluate the benefits of leveraging objective evaluation time, we also measure the decision latency of each algorithm.

5.1 Standard Benchmark Functions

Following De Ath et al. (2021a), we use the 15 benchmark functions[1] ranging from $d = 2$ to $d = 10$: Branin, Eggholder, Goldstein–Price and Six-Hump Camel ($d = 2$); Hartmann ($d = 3, 6$); Ackley and Michalewicz ($d = 5, 10$); Styblinski–Tang ($d = 5, 7, 10$); and Rosenbrock ($d = 7, 10$). Each run starts from $2d$ Sobol points and uses a total budget of 100 optimization steps. The methods that use MCMC take $M = 64$ hyperparameter samples per BO step. We consider noiseless observations.

The results presented in Fig. 3 demonstrate that ELF-BO tracks Standard FBO closely on most of the 15 functions, with curves largely overlapping within standard error. This agreement holds

[1] https://www.sfu.ca/~ssurjano/optimization.html

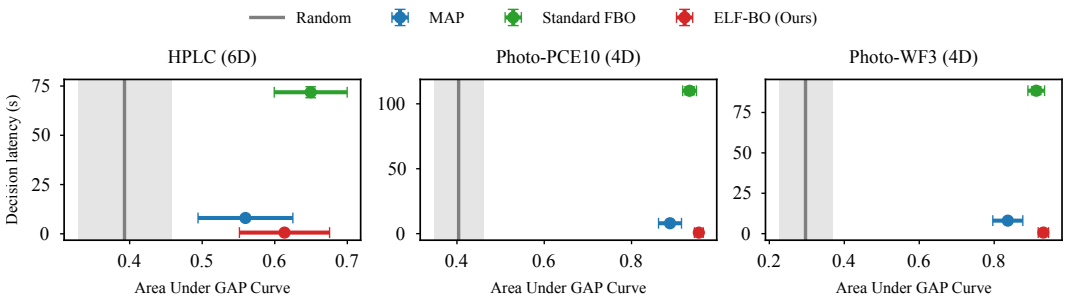

Figure 5: **Deep kernel on real-world tasks.** Results of the deep kernel on Olympus datasets. As in experiments with Matérn kernel, ELF-BO achieves Pareto optimality over Standard FBO and MAP, with an even more pronounced reduction in decision latency. See Appendix C for GAP curves.

across input dimensions and a range of landscape characteristics, including smooth functions such as Branin and more rugged, highly multimodal functions such as Eggholder. There are, however, functions for which ELF-BO and Standard FBO diverge, such as Michalewicz ($d = 5$). In Section 6 we analyze this failure through the lens of effective sample size. Finally, we observe that ELF-BO follows Standard FBO both when it outperforms MAP and when they are equivalent, showing no clear preference for cases where point estimates are "good enough". This matches the goal of the work, which is to provide a small-latency alternative to fully Bayesian optimization.

5.2 Real-World Applications

In addition, we evaluate ELF-BO on three experimental datasets curated in the Olympus benchmark suite (Häse et al., 2021). Rather than using learned emulators, we directly treat the datasets as a virtual library—thus, querying a point returns its recorded experimental measure. HPLC ($d = 6$ and $|\mathcal{X}| = 1{,}386$) tunes the sampling sequence of a robotic direct-inject HPLC system to maximize peak area (Roch et al., 2020). Photo-PCE10 and Photo-WF3 ($d = 4$ and $|\mathcal{X}| = 1{,}040$ each) optimize the composition of polymer blends to minimize[2] photodegradation (Langner et al., 2020). We choose the three Olympus datasets with more than 1000 data points to provide a large enough library for BO. GAP is computed with y_* set to the best measurement in each dataset.

All three datasets were collected through physical experiments. In a live campaign where Standard FBO chooses each experiment before it is run, every evaluation would take far longer than sampling the hyperparameter posterior, leaving the planner idle for most of the campaign—precisely the setting ELF-BO is designed for. Although the photodegradation data was collected in a high-throughput setting, we optimize all three tasks sequentially. We use an initial dataset of $2d$ Sobol points and a total budget of 50 steps for all tasks. Standard FBO and ELF-BO use $M = 128$ hyperparameter samples per BO step. More details on the tasks and datasets are provided in Appendix B.

The optimization curves presented in Fig. 4 (top) demonstrate that, across all tasks, ELF-BO closely tracks Standard FBO in performance. The results also highlight that in real-world applications with limited evaluation budget, a fully Bayesian approach can improve sample efficiency, achieving better results in earlier optimization steps. We analyze the sample efficiency using the area under the GAP curve, which is presented in Fig. 4 (bottom) alongside average decision latency. This demonstrates the Pareto dominance of ELF-BO over Standard FBO and MAP, displaying optimization performance on par with Standard FBO while providing suggestions more than $10\times$ faster.

Deep kernel. A natural use case for a fully Bayesian approach is with deep kernels (Wilson et al., 2016), as marginalizing out the network weights can mitigate the overfitting of point estimates in the overparameterized regime (Ober et al., 2021). In addition, they offer a challenging setting for ELF-BO, as the sampled parameters include neural-network weights, a step toward BNN surrogates. We repeat the real-world benchmarks, replacing the Matérn kernel with a deep kernel (two-layer neural network; see Appendix B). As shown in Fig. 5, the decision latency of Standard FBO grows 5–$9\times$ relative to Matérn, whereas ELF-BO remains under 1 s, a 70–$110\times$ reduction, while remaining competitive in sample efficiency.

[2]We negate the objective and consider it a maximization problem.

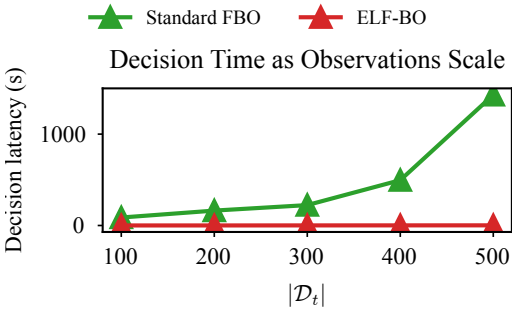

Figure 6: **Deep kernel on larger datasets.** Decision latency on Photo-WF3 ($d = 4$) as observation set grows. Standard FBO increases with dataset size while ELF-BO remains constant.

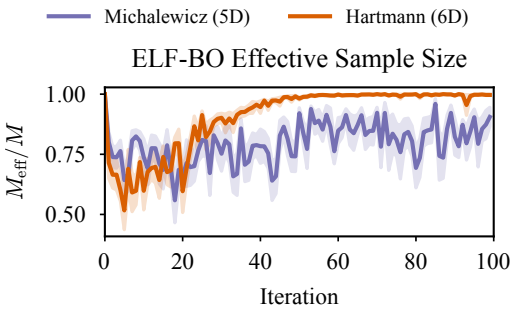

Figure 7: **Normalized effective sample size per iteration.** Michalewicz ($d = 5$) exhibits a lower effective sample size than Hartmann ($d = 6$) for most of the optimization steps.

6 Analysis

We now analyze ELF-BO under different settings, improving our understanding of the method. To this end, we study the scalability with respect to dataset size and the impact of effective sample size.

Scaling with the number of observations. The cost of fully Bayesian optimization grows with the number of observations n. However, ELF-BO is unaffected by this growth as long as sampling fits within the evaluation time. Figure 6 shows the decision latency as a function of the number of observations on Photo-WF3 ($d = 4$) for the deep kernel model in Section 5.2. While Standard FBO latency grows up to 24 minutes, ELF-BO remains nearly constant at under 3 seconds. This highlights the large gains that can be obtained by ELF-BO when optimization campaigns are long and use more complex models. The same figure indicates the evaluation time required to fully hide sampling at each step, providing a practical guide for when ELF-BO is applicable.

Impact of effective sample size. As noted in Section 3, ELF-BO depends on how representative the samples collected during the objective evaluation are. We analyze M_{eff} of Michalewicz ($d = 5$), a function for which ELF-BO has difficulties tracking Standard FBO performance, and Hartmann ($d = 6$), a function close in dimensionality for which ELF-BO performs well. Figure 7 presents the step-wise fraction of effective samples, defined as M_{eff}/M. The effective sample size for Michalewicz ($d = 5$) is lower for most of the optimization steps, although it remains far from collapse for both functions. The lower values of M_{eff} for Michalewicz ($d = 5$) help explain why ELF-BO performs worse on that function. Since M_{eff} is available as soon as y_t is observed, it provides a cheap diagnostic for flagging steps where the reweighted estimate may be less reliable.

7 Conclusion

We introduced ELF-BO, a fully Bayesian optimization algorithm that leverages the objective's execution latency to alleviate the time taken to make a decision on the next point to query. By drawing samples from the posterior given the data available at dispatch time and reweighting them once the observation is revealed, ELF-BO removes MCMC from the time between function evaluations. Across 15 synthetic functions and three real-world experimental datasets, ELF-BO matches the optimization performance of MCMC-based fully Bayesian optimization on most tasks while keeping decision latency close to point estimate BO times.

Limitations and Future Work ELF-BO's latency gains are observed in applications for which objective function evaluations take more time than MCMC, but are absent in applications where evaluations are near-instant. Moreover, we only consider GP surrogates in the sequential setting. Natural extensions include BNN surrogates and asynchronous batch BO with multiple pending evaluations. More broadly, ELF-BO shows that the planner's idle time, which is inherent to expensive black-box optimization, can be spent on better decisions.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods, formulating mathematical claims, assisting in the writing of proofs, and providing feedback on research methodology and experiments. We have not used generative AI tools for proposing or refining hypotheses, cleaning and reformatting datasets, supporting qualitative and thematic data analysis interpreting results, providing critical ingredients for proving mathematical claims, or helping develop theoretical models or conceptual frameworks; and assistance with translation and generating synthetic data sets are not applicable to this work. Additionally, we used generative AI tools for brainstorming, summarizing and analyzing existing literature, and editing a research paper to improve readability. We have reviewed all AI-assisted work. We used LLM-generated code to assist the development of the implementation of the proposed method, and we verified and tested it for correctness. We used LLMs with research skills during the literature survey stage of the project to generate a list of relevant works, which the authors read without using summarization tools. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The assumptions and complete proof for Proposition 1 are available in Appendix A. The full ELF-BO procedure is summarized in Algorithm 1 and Eq. (5). Further experimental details, including descriptions of real-world data, deep kernel details, and computational resources, are in Appendix B. The datasets used and code for the experiments are available anonymously at https://anonymous.4open.science/r/elf-bo-02F1.

REFERENCES

Ahsan S Alvi, Binxin Ru, Jan Calliess, Stephen J Roberts, and Michael A Osborne. Asynchronous Batch Bayesian Optimisation with Improved Local Penalisation. In *ICML*, 2019.

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for Bayesian optimization. In *NeurIPS*, 2023a.

Sebastian Ament, Andrew Witte, Nishant Garg, and Julius Kusuma. Sustainable Concrete via Bayesian Optimization. *arXiv*, 2310(18288), 2023b.

Peter Auer. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *JMLR*, 3(Nov), 2002.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *NeurIPS*, 2020.

Russell R. Barton. Minimization Algorithms for Functions with Random Noise. *American Journal of Mathematical and Management Sciences*, 4(1-2), 1984.

Romain Benassi, Julien Bect, and Emmanuel Vazquez. Robust Gaussian Process-Based Global Optimization Using a Fully Bayesian Expected Improvement Criterion. In *International Conference on Learning and Intelligent Optimization*, 2011.

Nicolas Chopin. A Sequential Particle Filter Method for Static Models. *Biometrika*, 89(3), 2002.

George De Ath, Richard M Everson, and Jonathan E Fieldsend. Asynchronous ε-Greedy Bayesian Optimisation. In *UAI*, 2021a.

George De Ath, Richard M Everson, and Jonathan E Fieldsend. How Bayesian Should Bayesian Optimisation Be? In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021b.

Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2), 1987.

David Eriksson and Martin Jankowiak. High-Dimensional Bayesian Optimization with Sparse Axis-Aligned Subspaces. In *UAI*, 2021.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. *Kriging Is Well-Suited to Parallelize Optimization*. Springer, 2010.

David Ginsbourger, Janis Janusevskis, and Rodolphe Le Riche. *Dealing with Asynchronicity in Parallel Gaussian Process Based Global Optimization*. PhD thesis, Mines Saint-Etienne, 2011.

Rebecca L Greenaway, Kim E Jelfs, Alan C Spivey, and Sophia N Yaliraki. From Alchemist to AI Chemist. *Nature Reviews Chemistry*, 7(8), 2023.

Florian Häse, Matteo Aldeghi, Riley J. Hickman, Loïc M. Roch, Melodie Christensen, Elena Liles, Jason E. Hein, and Alán Aspuru-Guzik. Olympus: a benchmarking framework for noisy optimization and experiment planning. *Machine Learning: Science and Technology*, 2(3), 2021.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *JMLR*, 15(1), 2014.

Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian Optimization Performs Great in High Dimensions. In *ICML*, 2024.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13, 1998.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised Bayesian Optimisation via Thompson Sampling. In *AISTATS*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, and Vincent Fortuin. Promises and pitfalls of the linearized Laplace in Bayesian optimization. In *Fifth Symposium on Advances in Approximate Bayesian Inference*, 2023.

Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. A sober look at LLMs for material discovery: Are they actually good for Bayesian optimization over molecules? In *ICML*, 2024.

Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in Bayesian Optimization with Applications in Aerospace Engineering. In *2018 AIAA Non-Deterministic Approaches Conference*, 2018.

Stefan Langner, Florian Häse, José Darío Perea, Tobias Stubhan, Jens Hauch, Loïc M Roch, Thomas Heumueller, Alán Aspuru-Guzik, and Christoph J Brabec. Beyond Ternary OPV: High-Throughput Experimentation and Self-Driving Laboratories Optimize Multicomponent Systems. *Advanced Materials*, 32(14), 2020.

Yucen Li, Tim GJ Rudner, and Andrew Gordon Wilson. A Study of Bayesian Neural Network Surrogates for Bayesian Optimization. In *ICLR*, 2024.

B. P. MacLeod, F. G. L. Parlane, T. D. Morrissey, F. Häse, L. M. Roch, K. E. Dettelbach, R. Moreira, L. P. E. Yunker, M. B. Rooney, J. R. Deeth, V. Lai, G. J. Ng, H. Situ, R. H. Zhang, M. S. Elliott, T. H. Haley, D. J. Dvorak, A. Aspuru-Guzik, J. E. Hein, and C. P. Berlinguette. Self-Driving Laboratory for Accelerated Discovery of Thin-Film Materials. *Science Advances*, 6(20), 2020.

J. Močkus. On Bayesian Methods for Seeking the Extremum. In *Optimization Techniques IFIP Technical Conference*, 1975.

Iain Murray and Ryan P Adams. Slice Sampling Covariance Hyperparameters of Latent Gaussian Models. In *NIPS*, 2010.

Madhav R Muthyala, Farshud Sorourifar, Tianhong Tan, You Peng, and Joel A Paulson. Generative Multiobjective Bayesian Optimization with Scalable Batch Evaluations for Sample-Efficient De Novo Molecular Design. *Industrial & Engineering Chemistry Research*, 65(1), 2026.

Radford Neal. *Handbook of Markov Chain Monte Carlo*, chapter 5: MCMC Using Hamiltonian Dynamics. CRC Press, 2011.

Radford M Neal. Slice sampling. *The Annals of Statistics*, 31(3), 2003.

Sebastian W Ober, Carl E Rasmussen, and Mark van der Wilk. The Promises and Pitfalls of Deep Kernel Learning. In *UAI*, 2021.

Michael A Osborne, Roman Garnett, and Stephen J Roberts. Gaussian Processes for Global Optimization. In *International Conference on Learning and Intelligent Optimization*, 2009.

Art B. Owen. *Monte Carlo Theory, Methods and Examples*. https://artowen.su.domains/mc/, 2013.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Ben Riegler, James Odgers, and Vincent Fortuin. Standard Acquisition Is Sufficient for Asynchronous Bayesian Optimization. In *AISTATS*, 2026.

Loïc M. Roch, Florian Häse, Christoph Kreisbeck, Teresa Tamayo-Mendoza, Lars P. E. Yunker, Jason E. Hein, and Alán Aspuru-Guzik. ChemOS: An Orchestration Software to Democratize Autonomous Discovery. *PLOS ONE*, 15(4), 2020.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *ICML*, 2010.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4), 1933.

Arun Verma, Zhongxiang Dai, and Bryan Kian Hsiang Low. Bayesian Optimization Under Stochastic Delayed Feedback. In *ICML*, 2022.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep Kernel Learning. In *AISTATS*, 2016.

A PROOFS

Proposition 1. *Let φ be a function of $\boldsymbol{\theta}$ and $p(\boldsymbol{\theta} \mid \mathcal{D}_t)$ be the posterior of hyperparameters at step $t \in \{1, \dots, T\}$. Suppose that, for each $t \geq 1$, $p(\boldsymbol{\theta} \mid \mathcal{D}_t)$ is absolutely continuous w.r.t. $p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})$. Then, for all $t = 1, \dots, T$,*

$$\mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_t)}[\varphi(\boldsymbol{\theta})] = \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})}[w(\boldsymbol{\theta})\, \varphi(\boldsymbol{\theta})], \qquad \textit{where} \quad w(\boldsymbol{\theta}) = \frac{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta})}{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t)}.$$

Proof. Let $t \in \{1, \dots, T\}$ be arbitrary. By the importance sampling argument:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_t)}[\varphi(\boldsymbol{\theta})] &= \int \varphi(\boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D}_t)\, d\boldsymbol{\theta} \\
&= \int \varphi(\boldsymbol{\theta}) \underbrace{\frac{p(\boldsymbol{\theta} \mid \mathcal{D}_t)}{p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})}}_{=: w(\boldsymbol{\theta})} p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})\, d\boldsymbol{\theta} \\
&= \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})}[w(\boldsymbol{\theta})\, \varphi(\boldsymbol{\theta})]\,.
\end{aligned}$$

In particular, the importance weight is $w(\boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta} \mid \mathcal{D}_t)}{p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})}$.

Recall that $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t, y_t)\}$. Thus $p(\boldsymbol{\theta} \mid \mathcal{D}_t) = p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, y_t)$. Therefore, by Bayes' rule we have that:

$$\begin{aligned}
p(\boldsymbol{\theta} \mid \mathcal{D}_t) &= p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, y_t) \\
&= \frac{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t)}{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t)} \\
&= \frac{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})}{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t)}\,.
\end{aligned}$$

Therefore, plugging this into $w(\boldsymbol{\theta})$ we get:

$$\begin{aligned}
w(\boldsymbol{\theta}) &= \frac{p(\boldsymbol{\theta} \mid \mathcal{D}_t)}{p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})} \\
&= \frac{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})}{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t)} \frac{1}{p(\boldsymbol{\theta} \mid \mathcal{D}_{t-1})} \\
&= \frac{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta})}{p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t)} \\
&\propto p(y_t \mid \mathcal{D}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}),
\end{aligned}$$

where the last proportionality is because the denominator is constant w.r.t. $\boldsymbol{\theta}$. □

B EXPERIMENTAL DETAILS

B.1 REAL-WORLD APPLICATIONS

HPLC. The dataset originates from the autonomous calibration of a robotic sampling sequence for direct-inject HPLC analysis (Roch et al., 2020). The authors report sessions of 100 autonomous experiments over approximately seven hours, corresponding to roughly four minutes per evaluation.

Photo-PCE10 and Photo-WF3. These datasets report the photostability of photovoltaic blends (Langner et al., 2020). The experimental time reported by the authors is 18 hours for a batch of samples. As mentioned in the main text, we consider the problem in the sequential (non-batch) optimization setting.

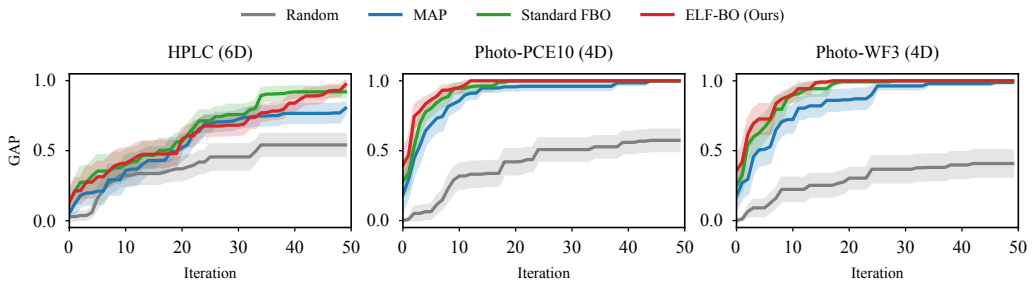

Figure 8: **Deep kernel on real-world tasks.** Optimization curves of the deep kernel methods on Olympus datasets.

B.2 Deep Kernel Details

The deep kernel is implemented as $\sigma_f \cdot k(\mathcal{E}(\boldsymbol{x}), \mathcal{E}(\boldsymbol{x}'))$, where k is a Matérn kernel with lengthscales fixed at 1, σ_f is a kernel outputscale, and $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^8$ is a neural network with a single hidden layer of dimension 16 and $\tanh$ activations. Taking into consideration the observation noise and constant mean, the deep kernel has a total of $16d + 155$ hyperparameters. To give NUTS a proper posterior to sample and prevent overfitting in MAP, we impose a zero-mean normal prior with variance 0.25 for all the neural network's biases and $4/L_{in}$ for its weights, where L_{in} is the dimension of the layer's input.

For the experiments in Section 5.2, the MAP deep kernel is optimized using 500 iterations of Adam (Kingma & Ba, 2015) per restart, with 4 restarts and a learning rate of 0.01. For Standard FBO and ELF-BO, samples are obtained using NUTS with a diagonal mass matrix, performing 512 warmup steps and 4096 samples, resulting in $M = 256$ after thinning by 16, with a max tree depth of 10. For the scaling experiments in Section 6 we use $M = 128$ and the Log-EI is evaluated on batches of 32 candidates at a time.

B.3 Computational Resources

Experiments are implemented in Python using the BoTorch (Balandat et al., 2020) library version 0.18.1. All synthetic benchmarks are executed on a single NVIDIA L40S, while real-world experiments make use of an NVIDIA H100.

B.4 Further Experimental Details

A throwaway NUTS run happens before the first BO step, so that device start-up is not considered in latency results.

Acquisition function optimization For the continuous synthetic functions, we use 10 restarts, with 512 raw samples. For the discrete real-world problems, every unused row is scored, 512 rows at a time.

C Additional Results

Deep Kernel Figure 8 presents the complete GAP curves for the deep kernel experiments described in Section 5.2.