Approximation-Aware Bayesian Optimization

Natalie Maus
University of Pennsylvania
nmaus@seas.upenn.edu

Kyurae Kim
University of Pennsylvania

Geoff Pleiss
University of British Columbia
Vector Institute

David Eriksson
Meta

John P. Cunningham
Columbia University

Jacob R. Gardner
University of Pennsylvania

Abstract

High-dimensional Bayesian optimization (BO) tasks such as molecular design often require >10,000 function evaluations before obtaining meaningful results. While methods like sparse variational Gaussian processes (SVGPs) reduce computational requirements in these settings, the underlying approximations result in suboptimal data acquisitions that slow the progress of optimization. In this paper we modify SVGPs to better align with the goals of BO: targeting informed data acquisition rather than global posterior fidelity. Using the framework of utility-calibrated variational inference, we unify GP approximation and data acquisition into a joint optimization problem, thereby ensuring optimal decisions under a limited computational budget. Our approach can be used with any decision-theoretic acquisition function and is readily compatible with trust region methods like TuRBO. We derive efficient joint objectives for the expected improvement and knowledge gradient acquisition functions for standard and batch BO. Our approach outperforms standard SVGPs on high-dimensional benchmark tasks in control and molecular design.

1 Introduction

Bayesian optimization (BO; Frazier, 2018; Garnett, 2023; Jones et al., 1998; Mockus, 1982; Shahriari et al., 2015) casts optimization as a sequential decision-making problem. Many recent successes of BO have involved complex and high-dimensional problems. In contrast to "classic" low-dimensional BO problems—where expensive black-box function evaluations far exceeded computational costs—these modern problems necessitate tens of thousands of function evaluations, and it is often the complexity and dimensionality of the search space that makes optimization challenging, rather than a limited evaluation budget (Eriksson et al., 2019; Griffiths and Hernández-Lobato, 2020; Maus et al., 2022, 2023; Stanton et al., 2022). Because of these scenarios, BO is entering a regime where computational costs are becoming a primary bottleneck (Maddox et al., 2021; Maus et al., 2023; Moss et al., 2023; Vakili et al., 2021), as the Gaussian process (GP; Rasmussen and Williams, 2005) surrogate models that underpin most of Bayesian optimization scale cubically with the number of observations.

In this new regime, we require scalable GP approximations, an area that has made tremendous progress over the last decade. In particular, sparse variational Gaussian processes (SVGP; Hensman et al., 2013; Quiñonero-Candela and Rasmussen, 2005; Titsias, 2009) have seen an increase in use (Griffiths and Hernández-Lobato, 2020; Maddox et al., 2021; Maus et al., 2022, 2023; Stanton et al., 2022; Tripp et al., 2020; Vakili et al., 2021), but many challenges remain to effectively deploy SVGPs for large-budget BO. In particular, the standard SVGP training objective is not aligned with the goals of black-box optimization. SVGPs construct an inducing point approximation that maximizes the standard variational evidence lower bound (ELBO; Jordan et al., 1999), yielding a posterior approximation $q^*(f)$ that models all observed data (Matthews et al., 2016; Moss et al., 2023). However, the optimal posterior approximation q^* is suboptimal for the decision-making tasks involved

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

in BO (Lacoste–Julien et al., 2011). In BO, we do not care about posterior fidelity at the majority of prior observations; rather, we only care about the fidelity of downstream functions involving the posterior, such as the expected utility. To illustrate this point intuitively, consider using the common expected improvement (EI; Jones et al., 1998) acquisition function for selecting new observations. Maximizing the ELBO might result in a posterior approximation that maintains fidelity for training examples in regions of virtually zero EI, thus wasting "approximation budget."

To solve this problem, we focus on the deep connections between statistical decision theory (Robert, 2001; Wasserman, 2013, §12) and Bayesian optimization (Garnett, 2023, §6-7), where acquisition maximization can be viewed as maximizing posterior-expected utility. Following this perspective, we leverage the utility-calibrated approximate inference framework (Jaiswal et al., 2020, 2023; Lacoste–Julien et al., 2011), and solve the aforementioned problem through a variational bound (Blei et al., 2017; Jordan et al., 1999)–the (log) **expected utility lower bound (`EULBO`)**—a joint function of the decision (the BO query) and the posterior approximation (the SVGP). When optimized jointly, the `EULBO` automatically yields the approximately optimal decision through the minorize-maximize principle (Lange, 2016). The `EULBO` is reminiscent of the standard variational ELBO (Jordan et al., 1999), and can indeed be viewed as a standard ELBO for a generalized Bayesian inference problem (Bissiri et al., 2016; Knoblauch et al., 2022), where we seek to approximate the *utility-weighted* posterior. This work represents the first application of utility-calibrated approximate inference towards BO despite its inherent connection with utility maximization.

The benefits of our proposed approach are visualized in Fig. 1. Furthermore, it can be applied to acquisition function that admits a decision-theoretic interpretation, which includes the popular expected improvement (EI; Jones et al., 1998) and knowledge gradient (KG; Wu et al., 2017) acquisition functions, and is trivially compatible with local optimization techniques like TuRBO (Eriksson et al., 2019) for high-dimensional problems. We demonstrate that our joint SVGP/acquisition optimization approach yields significant improvements across numerous Bayesian optimization benchmarks. As an added benefit, our approach can simplify the implementation and reduce the computational burden of complex (decision-theoretic) acquisition functions like KG. We demonstrate a novel algorithm derived from our joint optimization approach for computing and optimizing the KG that expands recent work on one-shot KG (Balandat et al., 2020) and variational GP posterior refinement (Maddox et al., 2021).

Overall, our contributions are summarized as follows:

- We propose utility-calibrated variational inference of SVGPs in the context of large-budget BO.
- We study this framework in two special cases using the utility functions of two common acquisition functions: EI and KG. For each, we derive tractable `EULBO` expressions that can be optimized.
- For KG, we demonstrate that the computation of the `EULBO` takes only negligible additional work over computing the standard ELBO by leveraging an online variational update. Thus, as a byproduct of optimizing the `EULBO`, optimizing KG becomes comparable to the cost of the EI.
- We extend this framework to be capable of running in batch mode, by introducing `q-EULBO` analogs of q-KG and q-EI as commonly used in practice (Wilson et al., 2018).
- We demonstrate the effectiveness of our proposed method against standard SVGPs trained with ELBO maximization on high-dimensional benchmark tasks in control and molecular design, where the dimensionality and evaluation budget go up to 256 and 80k, respectively.

2 Background

Noisy Black-Box Optimization. Noisy black-box optimization refers to problems of the form: $\text{maximize}_{\boldsymbol{x}\in\mathcal{X}}\ F(\boldsymbol{x})$, where $\mathcal{X} \subset \mathbb{R}^d$ is some compact domain, $F : \mathcal{X} \to \mathcal{Y}$ is some objective function, and we assume that only zeroth-order information of F is available. More formally, for some $i \in \mathbb{N}_{>0}$, we assume that observations of the objective function $(\boldsymbol{x}_i, y_i = \widehat{F}(\boldsymbol{x}_i))$ have been corrupted by independently and identically distributed (i.i.d.) Gaussian noise $\widehat{F}(\boldsymbol{x}_i) \triangleq F(\boldsymbol{x}_i) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_{\text{n}}^2)$. The noise variance σ_{n}^2 is also unknown.

Bayesian optimization. Bayesian Optimization (BO) is and iterative approach to noisy black-box optimization that iterates the following steps: ❶ At each step $t \geq 0$, we use a set of observations $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i = \widehat{F}(\boldsymbol{x}_i))\}_{i=1}^{n_t}$ of $\widehat{F}$ to fit a surrogate supervised model $f \in \mathcal{F}$. Typically, $\mathcal{F}$ is taken to be the sample space of a Gaussian process such that the function-valued posterior distribution

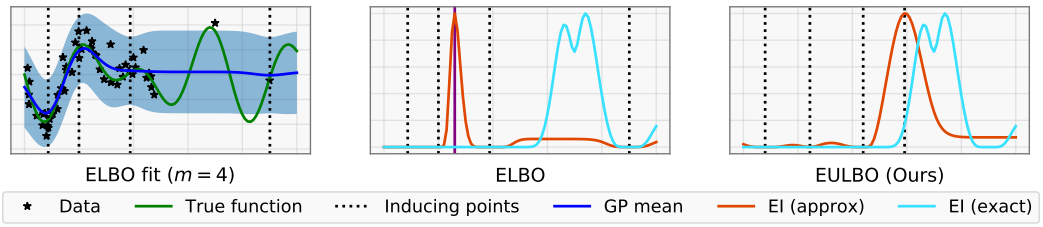

Figure 1: **(Left.)** Fitting an SVGP model with only $m = 4$ inducing points sacrifices modeling areas of high EI (few data points at right) because the ELBO focuses only on global data approximation (left data) and is ignorant of the downstream decision making task. **(Middle.)** Because of this, (normalized) EI with the SVGP model peaks in an incorrect location relative to the exact posterior. **(Right.)** Updating the GP fit and selecting a candidate jointly using the `EULBO` (our method) results in candidate selection much closer to the exact model.

$\pi(f \mid \mathcal{D})$ forms a distribution over surrogate models at step t. ❷ The posterior is then used to form a decision problem where we choose which point we should evaluate next, $\boldsymbol{x}_{t+1} = \delta_\alpha(\mathcal{D}_t)$, by maximizing an acquisition function $\alpha : \mathcal{X} \to \mathbb{R}$ as

$$\delta_\alpha(\mathcal{D}_t) \triangleq \arg\max_{\boldsymbol{x} \in \mathcal{X}} \ \alpha(\boldsymbol{x}; \mathcal{D}_t). \tag{1}$$

❸ After selecting $\boldsymbol{x}_{t+1}$, $\widehat{F}$ is evaluated to obtain the new datapoint $(\boldsymbol{x}_{t+1}, y_{t+1} = \widehat{F}(\boldsymbol{x}_{t+1}))$. This is then added to the dataset, forming $\mathcal{D}_{t+1} = \mathcal{D}_t \cup (\boldsymbol{x}_{t+1}, y_{t+1})$ to be used in the next iteration.

Utility-Based Acquisition Functions. Many commonly used acquisition functions, including EI and KG, can be expressed as posterior-expected utility functions

$$\alpha(\boldsymbol{x}; \mathcal{D}) \triangleq \int u(\boldsymbol{x}, f; \mathcal{D}) \, \pi(f \mid \mathcal{D}) \, \mathrm{d}f, \tag{2}$$

where $u(\boldsymbol{x}, f; \mathcal{D}) : \mathcal{X} \times \mathcal{F} \to \mathbb{R}$ is some utility function associated with α (Garnett, 2023, §6-7). In statistical decision theory, posterior-expected utility maximization policies such as δ_α are known as *Bayes policies*. These are important because, for a given utility function, they attain certain notions of statistical optimality such as Bayes optimality and admissibility (Robert, 2001, §2.4; Wasserman, 2013, §12). However, this only holds true if we can exactly compute Eq. (2) over the posterior. Once approximate inference is involved, making optimal Bayes decisions becomes challenging.

Sparse Variational Gaussian Processes. While the $\mathcal{O}(n^3)$ complexity of exact Gaussian process model selection and inference is not necessarily a roadblock in the traditional regression setting with 10,000-50,000 training examples, BO amplifies the scalability challenge by requiring us to sequentially train or update *many* large scale GPs as we iteratively acquire more data.

To address this, sparse variational GPs (SVGP; Hensman et al., 2013; Titsias, 2009) have become commonly used in high-throughput Bayesian optimization. SVGPs modify the original GP prior from $p(f)$ to $p(f \mid \boldsymbol{u})p(\boldsymbol{u})$, where we assume the latent function f is "induced" by a finite set of *inducing values* $\boldsymbol{u} = (u_1, \ldots, u_m) \in \mathbb{R}^m$ located at *inducing points* $\boldsymbol{z}_i \in \mathcal{X}$ for $i = 1, \ldots, m$. Inference is done through variational inference (Blei et al., 2017; Jordan et al., 1999), where the posterior of the inducing points is approximated using $q_{\boldsymbol{\lambda}}(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}; \boldsymbol{\lambda} = (\boldsymbol{m}, \boldsymbol{S}))$ and that of the latent functions with $q(f \mid \boldsymbol{u}) = p(f \mid \boldsymbol{u})$. Here, the variational parameters $\boldsymbol{m}$ and $\boldsymbol{S}$ are defined as the learned mean and covariance of the variational distribution $q_{\boldsymbol{\lambda}}(\boldsymbol{u})$. It is standard practice to define $\boldsymbol{\lambda} = (\boldsymbol{m}, \boldsymbol{S})$ so that $\boldsymbol{\lambda}$ can be used as shorthand to represent all of the trainable variational parameters. As is typical in the BO literature, we use the subscript $\boldsymbol{\lambda} \in \Lambda$ to denote that the distribution denoted as q contains trainable parameters in $\boldsymbol{\lambda}$.

For a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{>0}$, the resulting ELBO objective, which can be computed in a closed form (Hensman et al., 2013), is then

$$\mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{\lambda}; \mathcal{D}_t) \triangleq \mathbb{E}_{q_\lambda(f)}\left[\sum_{i=1}^{n_t} \log \ell(y_i \mid f(\boldsymbol{x}_i))\right] - \mathrm{D}_{\mathrm{KL}}(q_\lambda(\boldsymbol{u}), p(\boldsymbol{u})), \tag{3}$$

where $\ell(y_i \mid f(\boldsymbol{x}_i)) = \mathcal{N}(y_i \mid f(\boldsymbol{x}_i), \sigma_\epsilon)$ is a Gaussian likelihood. The marginal variational approximation can be computed as

$$q_\lambda(f) = \int q_\lambda(f, \boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} = \int p(f \mid \boldsymbol{u}) \ q_\lambda(\boldsymbol{u}) \, \mathrm{d}\boldsymbol{u}$$

such that the point-wise function evaluation on some $\boldsymbol{x} \in \mathcal{X}$ is

$$q_\lambda(f(\boldsymbol{x})) = \mathcal{N}\left(f(\boldsymbol{x}); \quad \mu_f(\boldsymbol{x}) \triangleq \boldsymbol{K}_{\boldsymbol{xZ}}\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{m}, \quad \sigma_f^2(\boldsymbol{x}) \triangleq \widetilde{k}_{\boldsymbol{xx}} + \boldsymbol{k}_{\boldsymbol{xZ}}^\top \boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{S}\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{k}_{\boldsymbol{Zx}}\right), \tag{4}$$

with $\widetilde{k}_{\boldsymbol{xx}} \triangleq k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_{\boldsymbol{xZ}}\boldsymbol{K}_{\boldsymbol{ZZ}}^{-1}\boldsymbol{k}_{\boldsymbol{Zx}}^\top$, the vector $\boldsymbol{k}_{\boldsymbol{Zx}} \in \mathbb{R}^m$ is formed as $[\boldsymbol{k}_{\boldsymbol{Zx}}]_i = k(\boldsymbol{z}_i, \boldsymbol{x})$, and the matrix $\boldsymbol{K}_{\boldsymbol{ZZ}} \in \mathbb{R}^{m \times m}$ is formed as $[\boldsymbol{K}_{\boldsymbol{ZZ}}]_{ij} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)$. Additionally, the GP likelihood and kernel contain hyperparameters, which we denote as $\boldsymbol{\theta} \in \Theta$, and we collectively denote the set of inducing point locations as $\boldsymbol{Z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m) \in \mathcal{X}^m$. We therefore denote the ELBO as $\mathcal{L}_{\text{ELBO}}(\boldsymbol{\lambda}, \boldsymbol{Z}, \boldsymbol{\theta}; \mathcal{D}_t)$.

3 Approximation-Aware Bayesian Optimization

When SVGPs are used in conjunction with BO (Maddox et al., 2021; Moss et al., 2023) at iteration $t \geq 0$, acquisition functions of the form of Eq. (2) are naïvely approximated as

$$\alpha(\boldsymbol{x}; \mathcal{D}) \approx \int u(\boldsymbol{x}, f; \mathcal{D}_t)\, q_\lambda(f)\, \mathrm{d}f,$$

where $q_\lambda(f)$ is the approximate SVGP posterior given by Eq. (4). The acquisition policy implied by this approximation contains two separate optimization problems:

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \int u(\boldsymbol{x}, f; \mathcal{D}_t)\, q_{\lambda^*_{\text{ELBO}}}(f)\, \mathrm{d}f \quad \text{and} \quad \boldsymbol{\lambda}^*_{\text{ELBO}} = \arg\max_{\boldsymbol{\lambda} \in \Lambda} \mathcal{L}_{\text{ELBO}}(\boldsymbol{\lambda}; \mathcal{D}_t). \tag{5}$$

Treating these optimization problems separately creates an artificial bottleneck that results in suboptimal data acquisition decisions. Intuitively, $\boldsymbol{\lambda}^*_{\text{ELBO}}$ is chosen to faithfully model all observed data (Matthews et al., 2016; Moss et al., 2023), without regard for how the resulting model performs at selecting the next function evaluation in the BO loop. For an illustration of this, see Figure 1. Instead, we propose a modification to SVGPs that couples the posterior approximation and data acquisition through a joint problem of the form:

$$(\boldsymbol{x}_{t+1}, \boldsymbol{\lambda}^*) = \arg\max_{\boldsymbol{\lambda} \in \Lambda, \boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{\text{EULBO}}(\boldsymbol{\lambda}, \boldsymbol{x}; \mathcal{D}_t). \tag{6}$$

This results in $\boldsymbol{x}_{t+1}$ directly approximating a solution to Eq. (2), where the **expected utility lower-bound** (`EULBO`) is an ELBO-like objective function derived below.

3.1 Expected Utility Lower-Bound

Consider an acquisition function of the form of Eq. (2), where the utility $u : \mathcal{X} \times \mathcal{F} \to \mathbb{R}_{>0}$ is strictly positive. We can derive a similar variational formulation of the acquisition function maximization problem following Lacoste–Julien et al. (2011). That is, given any distribution q_λ indexed by $\boldsymbol{\lambda} \in \Lambda$ and considering the SVGP prior augmentation $p(f) \to p(f \mid \boldsymbol{u})p(\boldsymbol{u})$, the acquisition function can be lower-bounded through Jensen's inequality as

$$\begin{aligned}
\log \alpha(\boldsymbol{x}; \mathcal{D}_t) &= \log \int u(\boldsymbol{x}, f; \mathcal{D}_t)\, \pi(f \mid \mathcal{D}_t)\, \mathrm{d}f \\
&= \log \int u(\boldsymbol{x}, f; \mathcal{D}_t)\, \pi(f, \boldsymbol{u} \mid \mathcal{D}_t)\, \frac{q_\lambda(f, \boldsymbol{u})}{q_\lambda(f, \boldsymbol{u})} \mathrm{d}f\, \mathrm{d}\boldsymbol{u} \\
&= \log \int u(\boldsymbol{x}, f; \mathcal{D}_t)\, \ell(\mathcal{D}_t \mid f)\, p(f \mid \boldsymbol{u})\, p(\boldsymbol{u})\, \frac{q_\lambda(\boldsymbol{u})\, p(f \mid \boldsymbol{u})}{q_\lambda(\boldsymbol{u})\, p(f \mid \boldsymbol{u})} \mathrm{d}f\, \mathrm{d}\boldsymbol{u} - \log Z \\
&\geq \int \log\left(\frac{u(\boldsymbol{x}, f; \mathcal{D}_t)\, \ell(\mathcal{D}_t \mid f)\, p(\boldsymbol{u})}{q_\lambda(\boldsymbol{u})}\right) p(f \mid \boldsymbol{u})\, q_\lambda(\boldsymbol{u})\, \mathrm{d}f\, \mathrm{d}\boldsymbol{u} - \log Z,
\end{aligned} \tag{7}$$

where Z is a normalizing constant. A restriction on u comes from the inequality in Eq. (7), where the utility needs to be strictly positive. This means that non-strictly positive utilities need to be modified to be incorporated into this framework. (See the examples by Kuśmierczyk et al., 2019.) Also, notice that the derivation is reminiscent of expectation-maximization (Dempster et al., 1977) and variational lower bounds (Jordan et al., 1999). That is, through the minorize-maximize principle (Lange, 2016), maximizing the lower bound with respect to $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ approximately solves the original problem of maximizing the posterior-expected utility.

Expected Utility Lower-Bound. Up to a constant and rearranging terms, maximizing Eq. (7) is equivalent to maximizing

$$\begin{aligned}\mathcal{L}_{\text{EULBO}}(\boldsymbol{\lambda}, \boldsymbol{x}; \mathcal{D}_t) &\triangleq \mathbb{E}_{p(f|\boldsymbol{u})q_\lambda(\boldsymbol{u})}\left[\log \ell(\mathcal{D}_t \mid f) + \log p(\boldsymbol{u}) - \log q_\lambda(\boldsymbol{u}) + \log u(\boldsymbol{x}, f; \mathcal{D}_t)\right] \\ &= \mathbb{E}_{q_\lambda(f)}\left[\sum_{i=1}^{n_t} \log \ell(y_i \mid f)\right] - \mathrm{D}_{\mathrm{KL}}(q_\lambda(\boldsymbol{u}), p(\boldsymbol{u})) + \mathbb{E}_{q_\lambda(f)} \log u(\boldsymbol{x}, f; \mathcal{D}_t) \\ &= \mathcal{L}_{\text{ELBO}}(\boldsymbol{\lambda}; \mathcal{D}_t) + \mathbb{E}_{q_\lambda(f)} \log u(\boldsymbol{x}, f; \mathcal{D}_t), \end{aligned} \tag{8}$$

which is the joint objective function alluded to in Eq. (6). We maximize EULBO to obtain $(\boldsymbol{x}_{t+1}, \boldsymbol{\lambda}^*) = \arg\max_{\boldsymbol{x}\in\mathcal{X}, \boldsymbol{\lambda}\in\Lambda} \mathcal{L}_{\text{EULBO}}(\boldsymbol{x}, \boldsymbol{\lambda})$, where $\boldsymbol{x}_{t+1}$ corresponds our next BO "query".

From Eq. (8), the connection between the EULBO and ELBO is obvious: the EULBO is now "nudging" the ELBO solution toward high utility regions. An alternative perspective is that we are approximating a *generalized posterior* weighted by the utility (Table. 1 by Knoblauch et al., 2022; Bissiri et al., 2016). Furthermore, Jaiswal et al. (2020, 2023) prove that the resulting actions satisfy consistency guarantees under assumptions typical in such results for variational inference (Wang and Blei, 2019).

Hyperparameters and Inducing Point Locations. For the hyperparameters $\boldsymbol{\theta}$ and inducing point locations $\boldsymbol{Z}$, we use the marginal likelihood to perform model selection, which is common practice in BO (Shahriari et al., 2015, §V.A). (Optimizing over $\boldsymbol{Z}$ was popularized by Snelson and Ghahramani, 2005.) Following suit, we also optimize the EULBO as a function of $\boldsymbol{\theta}$ and $\boldsymbol{Z}$ as

$$\underset{\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{Z}}{\text{maximize}} \left\{ \mathcal{L}_{\text{EULBO}}(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{Z}; \mathcal{D}_t) \triangleq \mathcal{L}_{\text{ELBO}}(\boldsymbol{\lambda}, \boldsymbol{Z}, \boldsymbol{\theta}; \mathcal{D}_t) + \mathbb{E}_{q_\lambda(f)} \log u(\boldsymbol{x}, f; \mathcal{D}_t) \right\}.$$

We emphasize here that the SVGP-associated parameters $\boldsymbol{\lambda}, \boldsymbol{\theta}, \boldsymbol{Z}$ have gradients that are determined by *both* terms above. Thus, the expected log-utility term $\mathbb{E}_{f\sim q_{\hat{\lambda}}(f)} \log u(\boldsymbol{x}, f; \mathcal{D}_t)$ simultaneously results in acquisition of $\boldsymbol{x}_{t+1}$ and directly influences the underlying SVGP regression model.

3.2 EULBO for Expected Improvement (EI)

The EI acquisition function can be expressed as a posterior-expected utility, where the underlying "improvement" utility function is given by the difference between the objective value of the query, $f(\boldsymbol{x})$, and the current best objective value $y_t^* = \max_{i=1,\ldots,t}\{y_i \mid y_i \in \mathcal{D}_t\}$:

$$u_{\text{EI}}(\boldsymbol{x}, f; \mathcal{D}_t) \triangleq \text{ReLU}\left(f(\boldsymbol{x}) - y_t^*\right), \qquad \text{(EI; Jones et al., 1998)} \tag{9}$$

where $\text{ReLU}(x) \triangleq \max(x, 0)$. Unfortunately, this utility is not strictly positive whenever $f(\boldsymbol{x}) \leq y^*$. Thus, we cannot immediately plug u_{EI} into the EULBO. While it is possible to add a small positive constant to u_{EI} and make it strictly positive as done by Kuśmierczyk et al. (2019), this results in a looser Jensen gap in Eq. (7), which could be detrimental. This also introduces the need for tuning the constant, which is not straightforward. Instead, we define the following "soft" EI utility:

$$u_{\text{SEI}}(\boldsymbol{x}, f; \mathcal{D}_t) \triangleq \text{softplus}\left(f(\boldsymbol{x}) - y_t^*\right),$$

where the ReLU in Eq. (9) is replaced with $\text{softplus}(x) \triangleq \log(1 + \exp(x))$. $\text{softplus}(x)$ converges to the ReLU in both extremes of $x \to \pm\infty$. Thus, u_{SEI} will behave closely to u_{EI}, while being slightly more explorative due to positivity.

Computing the EULBO and its derivatives now requires the computation of $\mathbb{E}_{f\sim q_{\hat{\lambda}}(f)} \log u_{\text{SEI}}(\boldsymbol{x}, f; \mathcal{D}_t)$, which, unlike EI, does not have a closed-form. However, since the utility function only depends on the function values of f, the expectation can be efficiently computed to high precision through one-dimensional Gauss-Hermite quadrature. Crucially, the expensive $K_{zz}^{-1}m$ and $K_{zz}^{-1}SK_{zz}^{-1}$ solves that dominate both the asymptotic and practical running time of both the ELBO and the EULBO are fixed across the log utility evaluations needed by quadrature. Because quadrature only depends on these precomputed moments, the additional work necessary due to lacking a closed form solution is negligible: Gauss-Hermite quadrature converges extremely quickly in the number of quadrature sites, and only requires on the order of 10 or so of these post-solve evaluations to achieve near machine precision.

3.3 EULBO for Knowledge Gradient (KG)

Although non-trivial, the KG acquisition is also a posterior-expected utility, where the underlying utility function is given by the maximum predictive mean value anywhere in the input domain *after*

conditioning on a new observation $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$:

$$u_{\mathrm{KG}}(\boldsymbol{x}, y; \mathcal{D}_t) \triangleq \max_{\boldsymbol{x}' \in \mathcal{X}} \mathbb{E}\left[f(\boldsymbol{x}') \mid \mathcal{D}_t \cup \{(\boldsymbol{x}, y)\}\right]. \qquad \text{(KG; Frazier, 2009; Garnett, 2023)}$$

Note that the utility function as defined above is not non-negative: the maximum predictive mean of a Gaussian process can be negative. For this reason, the utility function is commonly (and originally, *e.g.* Frazier, 2009, Eq. 4.11) written in the literature as the *difference* between the new maximum mean after conditioning on $(\boldsymbol{x}, y)$ and the maximum mean beforehand:

$$u_{\mathrm{KG}}(\boldsymbol{x}, y; \mathcal{D}_t) \triangleq \max_{\boldsymbol{x}' \in \mathcal{X}} \mathbb{E}\left[f(\boldsymbol{x}') \mid \mathcal{D}_t \cup \{(\boldsymbol{x}, y)\}\right] - \mu_t^+,$$

where $\mu_t^+ \triangleq \max_{\boldsymbol{x}'' \in \mathcal{X}} E\left[f(\boldsymbol{x}'') \mid \mathcal{D}_t\right]$. Note that μ_t^+ plays the role of a simple constant as it depends on neither $\boldsymbol{x}$ nor y. Similarly to the EI acquisition, this utility is still not strictly positive, and we thus define its "softplus-ed" variant:

$$u_{\mathrm{SKG}}(\boldsymbol{x}, y; \mathcal{D}_t) \triangleq \mathrm{softplus}\left(u_{\mathrm{KG}}(\boldsymbol{x}, y; \mathcal{D}_t) - c^+\right).$$

Here, c^+ acts as μ_t^+ by making u_{KG} positive as often as possible. This is particularly important when the GP predictive mean is negative as a consequence of the objective values being negative. One natural choice of constant is using μ_t^+; however, we find that simply choosing $c^+ = y_t^+$ works well and is more computationally efficient. Here, y_t^+ is the highest value of y_t (the highest objective value observed so far).

One-Shot KG `EULBO`. The `EULBO` using u_{SKG} results in an expensive nested optimization problem. To address this, we use an approach similar to the one-shot knowledge gradient method of Balandat et al. (2020). For clarity, we will define the reparameterization function

$$y_\lambda(\boldsymbol{x}; \epsilon_i) \triangleq \mu_{q_\lambda}(\boldsymbol{x}) + \sigma_{q_\lambda}(\boldsymbol{x})\,\epsilon_i,$$

where, for an i.i.d. sample $\epsilon_i \sim \mathcal{N}(0, 1)$, computing $y_i = y_\lambda(\boldsymbol{x}, \epsilon_i)$ is equivalent to sampling $y_i \sim \mathcal{N}\left(\mu_{q_\lambda}(\boldsymbol{x}), \sigma_{q_\lambda}(\boldsymbol{x})\right)$. This enables the use of the reparameterization gradient estimator (Kingma and Welling, 2014; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014). Now, notice that the KG acquisition function can be approximated through Monte Carlo as

$$\alpha_{\mathrm{KG}}(\boldsymbol{x}; \mathcal{D}) \approx \frac{1}{S}\sum_{i=1}^{S} u_{\mathrm{KG}}(\boldsymbol{x}, y_\lambda(\boldsymbol{x}; \epsilon_i); \mathcal{D}_t) = \frac{1}{S}\sum_{i=1}^{S} \max_{\boldsymbol{x}'} \mathbb{E}\left[f(\boldsymbol{x}') \mid \mathcal{D}_t \cup \{\boldsymbol{x}, y_\lambda(\boldsymbol{x}; \epsilon_i)\}\right],$$

where, for $i = 1, \ldots, S$, $\epsilon_i \sim \mathcal{N}(0, 1)$ are i.i.d. The one-shot KG approach absorbs the nested optimization over $\boldsymbol{x}'$ into a simultaneous joint optimization over $\boldsymbol{x}$ and a mean maximizer for each of the S samples, $\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_S$ such that $\max_{\boldsymbol{x}} \alpha_{\mathrm{KG}}(\boldsymbol{x}; \mathcal{D}_t) \approx \max_{\boldsymbol{x}, \boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_S} \alpha_{\text{1-KG}}(\boldsymbol{x}; \mathcal{D})$, where

$$\alpha_{\text{1-KG}}(\boldsymbol{x}; \mathcal{D}_t) \triangleq \frac{1}{S}\sum_{i=1}^{S} u_{\text{1-KG}}(\boldsymbol{x}, \boldsymbol{x}'_i, y_\lambda(\boldsymbol{x}; \epsilon_i); \mathcal{D}_t) = \frac{1}{S}\sum_{i=1}^{S} \mathbb{E}\left[f(\boldsymbol{x}'_i) \mid \mathcal{D}_t \cup \{\boldsymbol{x}, y_\lambda(\boldsymbol{x}; \epsilon_i)\}\right],$$

Evidently, there is no longer an inner optimization problem over $\boldsymbol{x}'$. To estimate the ith term of this sum, we draw a sample of the objective value of $\boldsymbol{x}$, $y_\lambda(\boldsymbol{x}; \epsilon_i)$, and condition the model on this sample. We then compute the new posterior predictive mean at $\boldsymbol{x}'_i$. After summing, we compute gradients with respect to both the candidate $\boldsymbol{x}$ and the mean maximizers $\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_S$. Again, we use the "soft" version of one-shot KG in our `EULBO` optimization problem:

$$u_{\text{1-SKG}}(\boldsymbol{x}, \boldsymbol{x}', y; \mathcal{D}_t) = \mathrm{softplus}\left(\mathbb{E}\left[f(\boldsymbol{x}') \mid \mathcal{D}_t \cup \{(\boldsymbol{x}, y)\}\right] - c^+\right),$$

where this utility function is crucially a function of both $\boldsymbol{x}$ and a free parameter $\boldsymbol{x}'$. As with $\alpha_{\text{1-KG}}$, maximizing the `EULBO` can be set up as a joint optimization problem:

$$\underset{\boldsymbol{x}, \boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_S, \lambda, \boldsymbol{Z}, \theta}{\text{maximize}} \quad \mathcal{L}_{\mathrm{ELBO}}(\lambda, \boldsymbol{Z}, \theta) + \frac{1}{S}\sum_{i=1}^{S} \log u_{\text{1-SKG}}\left(\boldsymbol{x}, \boldsymbol{x}'_i, y_\lambda(\boldsymbol{x}; \epsilon_i); \mathcal{D}_t\right) \tag{10}$$

Efficient KG-`EULBO` Computation. The computation time of the non-ELBO term in Eq. (10) is dominated by having to compute $\mathbb{E}\left[f(\boldsymbol{x}'_i) \mid \mathcal{D}_t \cup \{(\boldsymbol{x}, y_\lambda(\boldsymbol{x}; \epsilon_i))\}\right]$ S-times. Notice that we only need to compute an updated posterior predictive mean, and can ignore predictive variances. For this, we can leverage the online updating strategy of Maddox et al. (2021). In particular, the predictive mean can be updated in $\mathcal{O}(m^2)$ time using a simple Cholesky update. The additional $\mathcal{O}(Sm^2)$ cost of computing the `EULBO` is therefore amortized by the original $\mathcal{O}(m^3)$ cost of computing the ELBO.

3.4 Extension to q-EULBO for Batch Bayesian Optimization

The EULBO can be extended to support batch Bayesian optimization by using the Monte Carlo batch mode analogs of utility functions as discussed *e.g.* by Balandat et al. (2020); Wilson et al. (2018). Given a set of candidates $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_q) \in \mathcal{X}^q$, the q-EI utility function is given by:

$$u_{q\text{-EI}}(\boldsymbol{X}, \boldsymbol{f}; \mathcal{D}_t) \triangleq \max_{j=1...q} \text{ReLU}\left(f(\boldsymbol{x}_j) - y_t^*\right) \quad \text{(q-EI; Balandat et al., 2020; Wilson et al., 2018)}$$

This utility can again be softened as:

$$u_{q\text{-SEI}}(\boldsymbol{X}, \boldsymbol{f}; \mathcal{D}_t) \triangleq \max_{j=1...q} \text{softplus}\left(f(\boldsymbol{x}_j) - y_t^*\right)$$

Because this is now a q-dimensional integral, Gauss-Hermite quadrature is no longer applicable. However, we can apply Monte Carlo as

$$\mathbb{E}_{q_\lambda(f)} \log u_{q\text{-SEI}}(\boldsymbol{X}, \boldsymbol{f}; \mathcal{D}_t) \approx \frac{1}{S} \sum_{i=1}^{S} \max_{j=1...q} \text{softplus}\left(y_\lambda(\boldsymbol{x}; \epsilon_i) - y_t^*\right).$$

As done in the BoTorch software package (Balandat et al., 2020), we observe that fixing the set of base samples $\epsilon_1, ..., \epsilon_S$ during each BO iteration results in better optimization performance at the cost of negligible q-EULBO bias. Now, optimizing the q-EULBO is done over the full set of q candidates $(\boldsymbol{x}_1, ..., \boldsymbol{x}_q)$ jointly, as well as the GP hyperparameters, inducing points, and variational parameters.

Knowledge Gradient. The KG version of the EULBO can be similarly extended. The expected log utility term in the maximization problem Eq. (10) becomes:

$$\underset{\boldsymbol{x}_1,...,\boldsymbol{x}_q,\boldsymbol{x}'_1,...,\boldsymbol{x}'_S,\lambda,\boldsymbol{Z},\theta}{\text{maximize}} \mathcal{L}_{\text{ELBO}}(\lambda, \boldsymbol{Z}, \theta) + \frac{1}{S} \sum_{i=1}^{S} \max_{j=1..q} \log u_{1\text{-SKG}}(\boldsymbol{x}_j, \boldsymbol{x}'_i, y_\lambda(\boldsymbol{x}; \epsilon_i); \mathcal{D}_t),$$

resulting in a similar analog to q-KG as described by Balandat et al. (2020).

3.5 Optimizing the EULBO

Optimizing the ELBO for SVGPs is known to be challenging (Galy-Fajou and Opper, 2021; Terenin et al., 2024) as the optimization landscape for the inducing points is non-convex, multi-modal, and non-smooth. Naturally, these are also challenges for EULBO; we found that care must be taken when implementing and initializing the EULBO maximization problem. In this subsection, we outline some key ideas, while a detailed description with pseudocode is presented in Appendix A.

Initialization and Warm-Starting. We warm-start the EULBO maximization procedure by solving the conventional two-step scheme in Eq. (5): At each BO iteration, we obtain the "warm" initial values for $(\lambda, \boldsymbol{Z}, \theta)$ by optimizing the standard ELBO. Then, we use this to maximize the conventional acquisition function corresponding to the chosen utility function u (the expectation of u over $q_\lambda(f)$), which provides the warm-start initialization for $\boldsymbol{x}$.

Alternating Maximization Scheme. To optimize $\mathcal{L}_{\text{EULBO}}(\boldsymbol{x}, \lambda, \boldsymbol{Z}, \theta)$, we alternate between optimizing over the query $\boldsymbol{x}$ and the SVGP parameters $\lambda, \boldsymbol{Z}, \theta$. We find this block-coordinate descent scheme to be more stable and robust than jointly updating all parameters, though the reason why this is more stable than jointly optimizing all parameters requires further investigation.

4 Experiments

We evaluate EULBO-based SVGPs on a number of benchmark BO tasks, described in detail in Section 4.1. These tasks include standard low-dimensional BO problems, e.g., the 6D Hartmann function, as well as 7 high-dimensional and high-throughput optimization tasks.

Baselines. We compare EULBO to several baselines with the main goal of achieving a high reward using as few function evaluations as possible. Our primary point of comparison is ELBO-based SVGPs. We consider two approaches for inducing point locations: 1. optimizing inducing point locations via the ELBO (denoted as **ELBO**), 2. placing the inducing points using the strategy proposed by Moss et al. (2023) at each stage of ELBO optimization (denoted as **Moss et al.**). The latter offers improved BO performance over standard ELBO-SVGP in BO settings, yet—unlike our method—it exclusively

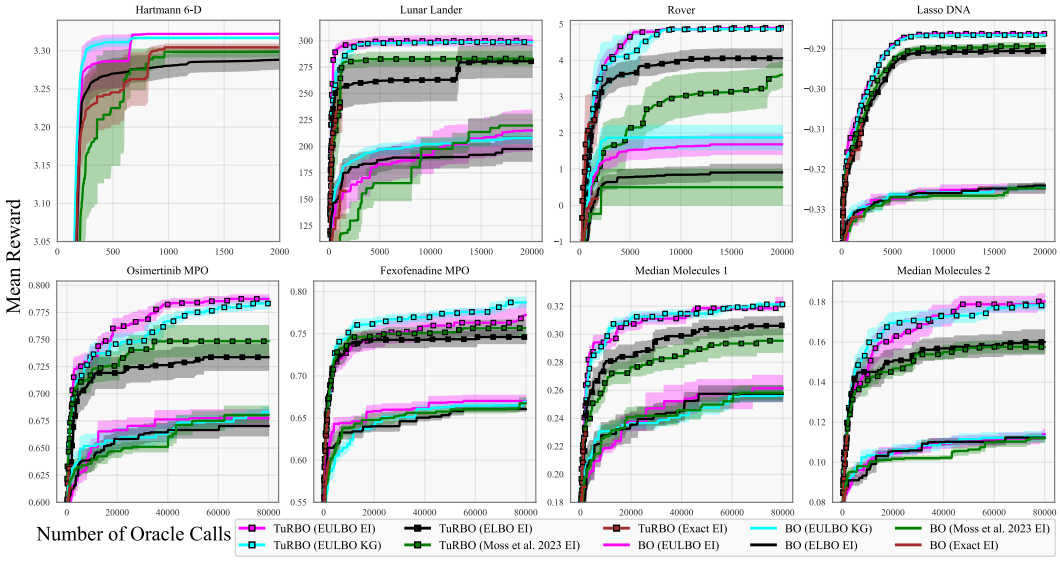

Figure 2: **Optimization results on the 8 considered tasks.** We compare all methods for both standard BO and TuRBO-based BO (on all tasks except Hartmann). Each line/shaded region represents the mean/standard error over 20 runs See subsection B.1 for additional molecule results.

targets inducing point placement and does not affect variational parameters or hyperparameters of the model. In addition, we compare to BO using exact GPs using 2,000 function evaluations as the use of exact GP is intractable beyond this point due to the need to *repeatedly* fit models.

Acquisition Functions and BO algorithms. For EULBO, we test the versions based on both the Expected Improvement (EI) and Knowledge Gradient (KG) acquisition functions as well as the batch variant. We test the baseline methods using EI only. On high-dimensional tasks (tasks with dimensionality above 10), we run EULBO and baseline methods with standard BO and with trust region Bayesian optimization (TuRBO) (Eriksson et al., 2019). For the largest tasks (Lasso, Molecules) we use acquisition batch size of 20 ($q = 20$), and batch size 1 ($q = 1$) for all others.

Implementation Details and Hyperparameters. Code to reproduce all results in the paper is available at https://github.com/nataliemaus/aabo. We implement EULBO and baseline methods using the GPyTorch (Gardner et al., 2018) and BoTorch (Balandat et al., 2020) packages. For all methods, we initialize using a set of 100 data points sampled uniformly at random in the search space. We use the same trust region hyperparameters as in (Eriksson et al., 2019). In Appendix B.1, we also evaluate an additional initialization strategy for the molecular design tasks. This alternative initialization matches prior work in using 10,000 molecules from the GuacaMol dataset Brown et al. (2019) rather than the details we used above for consistency across tasks, but does achieve higher overall performance.

4.1 Tasks

Hartmann 6D. The widely used Hartmann benchmark function (Surjanovic and Bingham, 2013).

Lunar Lander. The goal of this task is to find an optimal 12-dimensional control policy that allows an autonomous lunar lander to consistently land without crashing. The final objective value we optimize is the reward obtained by the policy averaged over a set of 50 random landing terrains. For this task, we use the same controller setup used by Eriksson et al. (2019).

Rover. The rover trajectory optimization task introduced by Wang et al. (2018) consists of finding a 60-dimensional policy that allows a rover to move along some trajectory while avoiding a set of obstacles. We use the same obstacle set up as in Maus et al. (2023).

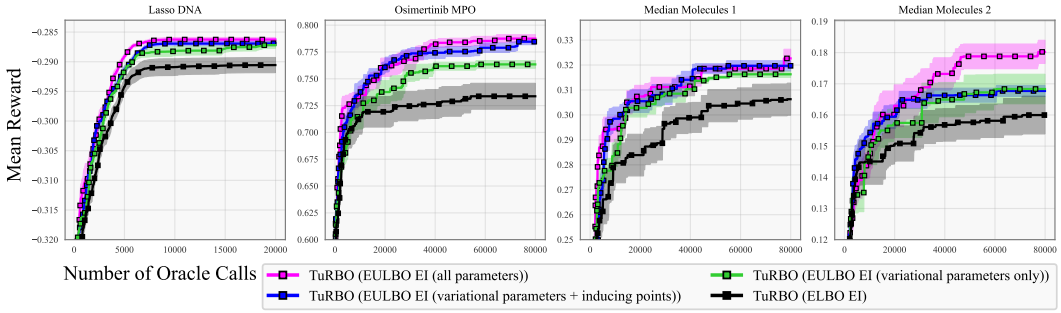

Figure 3: **Ablation study measuring the impact of EULBO optimization on various SVGP parameters.** At each BO iteration, we use the standard ELBO objective to optimize the SVGP hyperparameters, variational parameters, and inducing point locations. We then refine some subset of these parameters by further optimizing them with respect to the EULBO objective.

Lasso DNA. We optimize the 180−dimensional DNA task from the LassoBench library (Šehić et al., 2022) of benchmarks based on weighted LASSO regression (Gasso et al., 2009).

Molecular design tasks (x4). We select four challenging tasks from the Guacamol benchmark suite of molecular design tasks (Brown et al., 2019): Osimertinib MPO, Fexofenadine MPO, Median Molecules 1, and Median Molecules 2. We use the SELFIES-VAE introduced by Maus et al. (2022) to enable continuous 256 dimensional optimization.

4.2 Optimization Results

In Figure 2, we plot the reward of the best point found by the optimizer after a given number of function evaluations. Error bars show the standard error of the mean over 20 replicate runs. EULBO with TuRBO outperforms the other baselines with TuRBO. Similarly, EULBO with standard BO outperforms the other standard BO baselines. One noteworthy observation is that neither acquisition function appears to consistently outperform the other. However, EULBO-SVGP almost always dominates ELBO-SVGP and often requires a small fraction of the number of oracle calls to achieve comparable performance. These results suggest that coupling data acquisition with approximate inference/model selection results in significantly more sample-efficient optimization.

4.3 Ablation Study

While the results in Fig. 2 demonstrate that EULBO-SVGP improves the BO performance it is not immediately clear to what extent joint optimization modifies the posterior approximation beyond what is obtained by standard ELBO optimization. To that end, in Fig. 3 we refine an ELBO-SVGP model with varying degrees of additional EULBO optimization. At every BO iteration we begin by obtaining a SVGP model (where the variational parameters, inducing point locations, and GP hyperparameters are all obtained by optimizing the standard ELBO objective). We then refine some subset of parameters (either the inducing points, the variational parameters, the GP hyperparameters, or all of the above) through additional optimization with respect to the EULBO objective. Interestingly, we find that tasks respond differently to the varying levels of EULBO refinement. In the case of Lasso DNA, there is not much of a difference between EULBO refinement on all parameters versus refinement on the variational parameters alone. On the other hand, the performance on Median Molecules 2 is clearly dominated by refinement on all parameters. Nevertheless, we see that EULBO is always beneficial, whether applied to all parameters or some subset.

5 Related Work

Scaling Bayesian Optimization to the Large-Budget Regime. BO has traditionally been confined to the small-budget optimization regime with a few hundred objective evaluations at most. However, recent interest in high-dimensional optimization problems has demonstrated the need to scale BO to large data acquisition budgets. For problems with $\sim 10^3$ data acquisitions, Hernández-Lobato et al. (2017); Snoek et al. (2015); Springenberg et al. (2016) consider Bayesian neural networks (BNN; Neal, 1996), McIntire et al. (2016) use SVGP, and Wang et al. (2018) turn to ensembles of

subsampled GPs. For problems with $\gg 10^3$ acquisitions, SVGP has become the *de facto* approach to alleviate computational complexity (Griffiths and Hernández-Lobato, 2020; Maus et al., 2022, 2023; Stanton et al., 2022; Tripp et al., 2020; Vakili et al., 2021). As in this paper, many works have proposed modifications to SVGP to improve its performance in BO applications. Moss et al. (2023) proposed an inducing point placement based on a heuristic modification of determinantal point processes (Kulesza and Taskar, 2012), which we used for initialization, while Maddox et al. (2021) proposed a method for a fast online update strategy for SVGPs, which we utilize for the KG acquisition strategy.

Utility-Calibrated Approximate Inference. The utility-calibrated VI objective was first proposed by Lacoste–Julien et al. (2011), where they used a coordinate ascent algorithm to maximize it. Since then, various extensions have been proposed: Kuśmierczyk et al. (2019) leverage black-box variational inference (Ranganath et al., 2014; Titsias and Lázaro-Gredilla, 2014); Morais and Pillow (2022) use expectation-propagation (EP; Minka, 2001); Abbasnejad et al. (2015) employ importance sampling; Cobb et al. (2018) and Li and Zhang (2023) derive a specific variant for BNNs; and (Wei et al., 2021) derive a specific variant for GP classification. Closest to our work is the GP-based recommendation model learning algorithm by Abbasnejad et al. (2013), which sparsifies an EP-based GP approximation by maximizing a utility similar to those used in BO.

6 Limitations and Discussion

The main limitation of our proposed approach is increased computational cost. While `EULBO`-SVGP still retains the $O(m^3)$ computational complexity of standard SVGP, our practical implementation requires a warm-start: first fitting SVGP with the ELBO loss and then maximizing the acquisition function before jointly optimizing with the `EULBO` loss. Furthermore, `EULBO` optimization currently requires multiple tricks such as clipping and block-coordinate updates. In future work, we aim to develop a better understanding of the `EULBO` geometry in order to develop developing more stable, efficient, and easy-to-use `EULBO` optimization schemes. Nevertheless, our results in Section 4 demonstrate that the additional computation of `EULBO` yields substantial improvements in BO data-efficiency, a desirable trade-off in many applications. Moreover, `EULBO`-SVGP is modular, and our experiments capture a fraction of its potential use. It can be applied to any decision-theoretic acquisition function, and it is likely compatible with non-standard Bayesian optimization problems such as cost-constrained BO (Snoek et al., 2012), causal BO (Aglietti et al., 2020), and many more.

More importantly, our paper highlights a new avenue for research in BO, where surrogate modeling, approximate inference, and data selection are jointly determined from a unified objective. Extending this idea to GP approximations beyond SVGP and acquisition functions beyond EI/KG may yield further improvements, especially in the increasingly popular high-throughput BO setting.

Acknowledgments and Disclosure of Funding

The authors thank the anonymous reviewers for suggestions that improved the quality of the work.

N. Maus was supported by the National Science Foundation Graduate Research Fellowship; K. Kim was supported by a gift from AWS AI to Penn Engineering's ASSET Center for Trustworthy AI; G. Pleiss was supported by NSERC and the Canada CIFAR AI Chair program; J. P. Cunningham was supported by the Gatsby Charitable Foundation (GAT3708), the Simons Foundation (542963), the NSF AI Institute for Artificial and Natural Intelligence (ARNI: NSF DBI 2229929), and the Kavli Foundation; J. R. Gardner was supported by NSF awards IIS-2145644 and DBI-2400135.

References

Ehsan Abbasnejad, Justin Domke, and Scott Sanner. Loss-calibrated Monte Carlo action selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29 of *AAAI*. AAAI Press, March 2015. (page 10)

M. Ehsan Abbasnejad, Edwin V. Bonilla, and Scott Sanner. Decision-theoretic sparsification for Gaussian process preference learning. In *Machine Learning and Knowledge Discovery in Databases*, volume 13717 of *LNCS*, pages 515–530, Berlin, Heidelberg, 2013. Springer. (page 10)

Virginia Aglietti, Xiaoyu Lu, Andrei Paleyes, and Javier González. Causal Bayesian optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 108 of *PMLR*, pages 3155–3164. JMLR, June 2020. (page 10)

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 21524–21538. Curran Associates, Inc., 2020. (pages 2, 6, 7, 8, 16)

P. G. Bissiri, C. C. Holmes, and S. G. Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 78(5):1103–1130, 2016. (pages 2, 5)

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017. (pages 2, 3)

Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3): 1096–1108, Mar 2019. (pages 8, 9)

Adam D. Cobb, Stephen J. Roberts, and Yarin Gal. Loss-Calibrated Approximate Inference in Bayesian Neural Networks. arXiv Preprint arXiv:1805.03901, arXiv, May 2018. (page 10)

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, September 1977. (page 4)

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32, pages 5496–5507. Curran Associates, Inc., 2019. (pages 1, 2, 8)

Peter I Frazier. *Knowledge-gradient methods for statistical learning*. PhD thesis, Princeton University Princeton, 2009. (page 6)

Peter I Frazier. A tutorial on Bayesian optimization. arXiv Preprint arXiv:1807.02811, ArXiv, 2018. (page 1)

Théo Galy-Fajou and Manfred Opper. Adaptive inducing points selection for Gaussian processes. arXiv Preprint arXiv:2107.10066, arXiv, 2021. (page 7)

Jacob Gardner, Geoff Pleiss, Kilian Q. Weinberger, David Bindel, and Andrew G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, volume 31, pages 7576–7586. Curran Associates, Inc., 2018. (pages 8, 16)

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, Cambridge, United Kingdom ; New York, NY, 2023. (pages 1, 2, 3, 6)

Gilles Gasso, Alain Rakotomamonjy, and Stéphane Canu. Recovering sparse signals with a certain family of nonconvex penalties and DC programming. *IEEE Transactions on Signal Processing*, 57 (12):4686–4698, 2009. (page 9)

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. (pages 1, 10)

James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 282–290. AUAI Press, 2013. (pages 1, 3)

José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *Proceedings of the International Conference on Machine Learning*, volume 70 of *PMLR*, pages 1470–1479. JMLR, July 2017. (page 9)

Prateek Jaiswal, Harsha Honnappa, and Vinayak A. Rao. Asymptotic consistency of loss-calibrated variational Bayes. *Stat*, 9(1):e258, 2020. (pages 2, 5)

Prateek Jaiswal, Harsha Honnappa, and Vinayak Rao. On the statistical consistency of risk-sensitive bayesian decision-making. In *Advances in Neural Information Processing Systems*, volume 36, pages 53158–53200. Curran Associates, Inc., December 2023. (pages 2, 5)

Martin Jankowiak, Geoff Pleiss, and Jacob R. Gardner. Parametric gaussian process regressors. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020. (page 19)

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. (pages 1, 2, 5)

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999. (pages 1, 2, 3, 4)

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*, San Diego, California, USA, 2015. (pages 15, 16)

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*, Banff, AB, Canada, April 2014. (page 6)

Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. An optimization-centric view on Bayes' rule: Reviewing and generalizing variational inference. *Journal of Machine Learning Research*, 23 (132):1–109, 2022. (pages 2, 5)

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012. (page 10)

Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. Variational Bayesian decision-making for continuous utilities. In *Advances in Neural Information Processing Systems*, volume 32, pages 6395–6405. Curran Associates, Inc., 2019. (pages 4, 5, 10)

Simon Lacoste–Julien, Ferenc Huszár, and Zoubin Ghahramani. Approximate inference for the loss-calibrated Bayesian. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 15 of *PMLR*, pages 416–424. JMLR, June 2011. (pages 2, 4, 10)

Kenneth Lange. *MM Optimization Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, 2016. (pages 2, 4)

Bolian Li and Ruqi Zhang. Long-tailed Classification from a Bayesian-decision-theory Perspective. arXiv Preprint arXiv:2303.06075, arXiv, 2023. (page 10)

Wesley J Maddox, Samuel Stanton, and Andrew G Wilson. Conditioning sparse variational Gaussian processes for online decision-making. In *Advances in Neural Information Processing Systems*, volume 34, pages 6365–6379. Curran Associates, Inc., 2021. (pages 1, 2, 4, 6, 10)

Alexander G. de G. Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 51 of *PMLR*, pages 231–239. JMLR, May 2016. (pages 1, 4)

Natalie Maus, Haydn Jones, Juston Moore, Matt J. Kusner, John Bradshaw, and Jacob Gardner. Local latent space Bayesian optimization over structured inputs. In *Advances in Neural Information Processing Systems*, volume 35, pages 34505–34518, December 2022. (pages 1, 9, 10)

Natalie Maus, Kaiwen Wu, David Eriksson, and Jacob Gardner. Discovering many diverse solutions with Bayesian optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 206, pages 1779–1798. PMLR, April 2023. (pages 1, 8, 10)

Mitchell McIntire, Daniel Ratner, and Stefano Ermon. Sparse Gaussian Processes for Bayesian Optimization. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, Jersey City, New Jersey, USA, 2016. AUAI Press. (page 9)

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. (page 10)

Jonas Mockus. The Bayesian approach to global optimization. In *System Modeling and Optimization*, pages 473–481. Springer, 1982. (page 1)

Michael J. Morais and Jonathan W. Pillow. Loss-calibrated expectation propagation for approximate Bayesian decision-making. Technical Report arXiv:2201.03128, arXiv, January 2022. (page 10)

Henry B. Moss, Sebastian W. Ober, and Victor Picheny. Inducing point allocation for sparse Gaussian processes in high-throughput Bayesian optimisation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 206 of *PMLR*, pages 5213–5230. JMLR, April 2023. (pages 1, 4, 7, 10, 16, 17, 18)

Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY, 1996. (page 9)

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005. (page 1)

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 33 of *PMLR*, pages 814–822. JMLR, April 2014. (page 10)

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, November 2005. (page 1)

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, volume 32 of *PMLR*, pages 1278–1286. JMLR, June 2014. (page 6)

Christian P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer Texts in Statistics. Springer, New York Berlin Heidelberg, 2. ed edition, 2001. (pages 2, 3)

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. (pages 1, 5)

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18, pages 1257–1264. MIT Press, 2005. (page 5)

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25:2951–2959, 2012. (page 10)

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *Proceedings of the International Conference on Machine Learning*, volume 37 of *PMLR*, pages 2171–2180. JMLR, June 2015. (page 9)

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian Optimization with Robust Bayesian Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29, pages 4134–4142. Curran Associates, Inc., 2016. (page 9)

Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, volume 162 of *PMLR*, pages 20459–20478. JMLR, June 2022. (pages 1, 10)

Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets, 2013. (page 8)

Alexander Terenin, David R. Burt, Artem Artemev, Seth Flaxman, Mark van der Wilk, Carl Edward Rasmussen, and Hong Ge. Numerically stable sparse Gaussian processes via minimum separation using cover trees. *Journal of Machine Learning Research*, 25(26):1–36, 2024. (page 7)

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5 of *PMLR*, pages 567–574. JMLR, April 2009. (pages 1, 3)

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the International Conference on Machine Learning*, volume 32 of *PMLR*, pages 1971–1979. JMLR, June 2014. (pages 6, 10)

Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *Advances in Neural Information Processing Systems*, volume 33, pages 11259–11272. Curran Associates, Inc., 2020. (pages 1, 10)

Sattar Vakili, Henry Moss, Artem Artemev, Vincent Dutordoir, and Victor Picheny. Scalable Thompson sampling using sparse Gaussian process models. In *Advances in Neural Information Processing Systems*, volume 34, pages 5631–5643, 2021. (pages 1, 10)

Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. Lassobench: A high-dimensional hyperparameter optimization benchmark suite for LASSO. In *Proceedings of the International Conference on Automated Machine Learning*, volume 188 of *PMLR*, pages 2/1–24. JMLR, 25–27 Jul 2022. (page 9)

Yixin Wang and David M. Blei. Frequentist consistency of variational Bayes. *Journal of the American Statistical Association*, 114(527):1147–1161, July 2019. (page 5)

Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 84 of *PMLR*, pages 745–754. JMLR, March 2018. (pages 8, 9)

Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013. (pages 2, 3)

Yadi Wei, Rishit Sheth, and Roni Khardon. Direct loss minimization for sparse Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 130 of *PMLR*, pages 2566–2574. JMLR, March 2021. (page 10)

James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 9884–9895. Curran Associates, Inc., 2018. (pages 2, 7)

Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. In *Advances in Neural Information Processing Systems*, volume 30, pages 5267–5278. Curran Associates, Inc., 2017. (page 2)

A Implementation Details

We will now provide additional details on the implementation. For the implementation, we treat the SVGP parameters, such as the variational parameters $\boldsymbol{\lambda}$, inducing point locations $\boldsymbol{Z}$, and hyperparameters $\boldsymbol{\theta}$, equally. Therefore, for clarity, we will collectively denote them as $\boldsymbol{w} = (\boldsymbol{\lambda}, \boldsymbol{Z}, \boldsymbol{\theta})$ such that $\boldsymbol{w} \in \mathcal{W} \triangleq \Lambda \times \mathcal{X}^m \times \Theta$, and the resulting SVGP variational approximation as $q_{\boldsymbol{w}}$. Then, the ELBO and EULBO are equivalently denoted as follows:

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{w}; \mathcal{D}) \triangleq \mathcal{L}_{\text{ELBO}}(\boldsymbol{\lambda}, \boldsymbol{Z}, \boldsymbol{\theta}; \mathcal{D})$$
$$\mathcal{L}_{\text{EULBO}}(\boldsymbol{x}, \boldsymbol{w}; \mathcal{D}_{\boldsymbol{x}}, \mathcal{D}_{\boldsymbol{w}}) \triangleq \mathbb{E}_{f \sim q_{\boldsymbol{w}}(f)} \log u(\boldsymbol{x}, f; \mathcal{D}_{\boldsymbol{x}}) + \mathcal{L}_{\text{ELBO}}(\boldsymbol{w}; \mathcal{D}_{\boldsymbol{w}}).$$

Also, notice that the $\mathcal{L}_{\text{EULBO}}$ separately denote the dataset to be passed to the utility and the ELBO. (Setting $\mathcal{D}_t = \mathcal{D}_{\boldsymbol{w}} = \mathcal{D}_{\boldsymbol{x}}$ retrieves the original formulation in Eq. (8).)

Alternating Updates We perform block-coordinate ascent on the EULBO by alternating between maximizing over $\boldsymbol{x}$ as $\boldsymbol{w}$. Using vanilla gradient descent, the $\boldsymbol{x}$-update is equivalent to

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \gamma_{\boldsymbol{x}} \nabla_{\boldsymbol{x}} \mathcal{L}_{\text{EULBO}}(\boldsymbol{x}, \boldsymbol{w}; \mathcal{D}) = \boldsymbol{x} + \gamma_{\boldsymbol{x}} \nabla_{\boldsymbol{x}} \mathbb{E}_{f \sim q_{\boldsymbol{w}}(f)} \log u(\boldsymbol{x}, f; \mathcal{D}),$$

where $\gamma_{\boldsymbol{x}}$ is the stepsize. On the other hand, for the $\boldsymbol{w}$-update, we subsample the data such that we optimize the ELBO over a minibatch $S \subset \mathcal{D}$ of size $B = |S|$ as

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \gamma_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} \mathcal{L}_{\text{EULBO}}(\boldsymbol{x}, \boldsymbol{w}; S, \mathcal{D}) = \boldsymbol{w} + \gamma_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} \left(\mathbb{E}_{f \sim q_{\boldsymbol{w}}(f)} \log u(\boldsymbol{x}, f; \mathcal{D}) + \mathcal{L}_{\text{ELBO}}(\boldsymbol{w}; S) \right),$$

where $\gamma_{\boldsymbol{w}}$ is the stepsize. Naturally, the $\boldsymbol{w}$-update is stochastic due to minibatching, while the $\boldsymbol{x}$-update is deterministic. In practice, we leverage the Adam update rule (Kingma and Ba, 2015) instead of simple gradient descent. Together with gradient clipping, this alternating update scheme is much more robust than jointly updating $(\boldsymbol{x}, \boldsymbol{w})$.

Algorithm 1: EULBO Maximization Policy

Input: SVGP parameters $\boldsymbol{w}_0 = (\boldsymbol{\lambda}_0, \boldsymbol{Z}_0, \boldsymbol{\theta}_0)$, Dataset $\mathcal{D}_t$, BO utility function u,
Output: BO query $\boldsymbol{x}_{t+1}$

▷ `Compute Warm-Start Initializations`
$\boldsymbol{w} \leftarrow \arg\max_{\boldsymbol{w} \in \mathcal{W}} \mathcal{L}_{\text{ELBO}}(\boldsymbol{w}; \mathcal{D}_t)$ with $\boldsymbol{w}_0$ as initialization.
$\boldsymbol{x} \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{X}} \int u(\boldsymbol{x}, f; \mathcal{D}_t)\, q_{\boldsymbol{w}}(f)\, \mathrm{d}f$

▷ `Maximize EULBO`
repeat
  ▷ `Update posterior approximation` $q_{\boldsymbol{w}}$
  Fetch minibatch S from $\mathcal{D}_t$
  Compute $\boldsymbol{g}_{\boldsymbol{w}} \leftarrow \nabla_{\boldsymbol{w}} \mathcal{L}_{\text{EULBO}}(\boldsymbol{x}, \boldsymbol{w}; S, \mathcal{D}_t)$
  Clip $\boldsymbol{g}_{\boldsymbol{w}}$ with threshold G_{clip}
  $\boldsymbol{w} \leftarrow \text{AdamStep}_{\gamma_{\boldsymbol{w}}}(\boldsymbol{w}, \boldsymbol{g}_{\boldsymbol{w}})$

  ▷ `Update BO query` $\boldsymbol{x}$
  Compute $\boldsymbol{g}_{\boldsymbol{x}} \leftarrow \nabla_{\boldsymbol{x}} \mathcal{L}_{\text{EULBO}}(\boldsymbol{x}, \boldsymbol{w}; S, \mathcal{D}_t)$
  Clip $\boldsymbol{g}_{\boldsymbol{x}}$ with threshold G_{clip}
  $\boldsymbol{x} \leftarrow \text{AdamStep}_{\gamma_{\boldsymbol{x}}}(\boldsymbol{x}, \boldsymbol{g}_{\boldsymbol{x}})$
  $\boldsymbol{x} \leftarrow \text{proj}_{\mathcal{X}}(\boldsymbol{x})$
until *until converged*
$\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}$

Overview of Pseudocode. The complete high-level view of the algorithm is presented in Algorithm 1, except for the acquisition-specific details. $\text{AdamStep}_{\gamma}(\boldsymbol{x}, \boldsymbol{g})$ applies the Adam stepsize rule (Kingma and Ba, 2015) to the current location $\boldsymbol{x}$ with the gradient estimate $\boldsymbol{g}$ and the stepsize γ. In practice, Adam is a "stateful" optimizer, which maintains two scalar-valued states for each scalar parameter. For this, we re-initialize the Adam states at the beginning of each BO step.

Initialization. In the initial BO step $t = 0$, we initialize $\boldsymbol{Z}_0$ with the DPP-based inducing point selection strategy of Moss et al. (2023). For the remaining SVGP parameters $\boldsymbol{\lambda}_0$ and θ_0, we used the default initialization of GPyTorch (Gardner et al., 2018). For the remaining BO steps $t > 0$, we use $\boldsymbol{w}$ from the previous BO step as the initialization $\boldsymbol{w}_0$ of the current BO step.

Warm-Starting. Due to the non-convexity and multi-modality of both the ELBO and the acquisition function, it is critical to appropriately initialize the EULBO maximization procedure. As mentioned in Section 3.5, to warm-start the EULBO maximization procedure, we use the conventional 2-step scheme Eq. (5), where we maximize the ELBO and then maximize the acquisition function. For ELBO maximization, we apply Adam (Kingma and Ba, 2015) with the stepsize set as γ_w until the convergence criteria (described below) are met. For acquisition function maximization, we invoke the highly optimized `BoTorch.optimize.optimize_acqf` function (Balandat et al., 2020).

Minibatch Subsampling Strategy. As commonly done, we use the reshuffling subsampling strategy where the dataset $\mathcal{D}_t$ is shuffled and partitioned into minibatches of size B. The number of minibatches constitutes an "epoch." The dataset is reshuffled/repartitioned after going through a full epoch.

Convergence Determination. For both maximizing the ELBO during warm-starting and maximizing the EULBO, we continue optimization until we stop making progress or exceed k_{epochs} number of epochs. That is if the ELBO/EULBO function value fails to make progress for n_{fail} number of steps.

Table 1: Configurations of Hyperparameters used for the Experiments

Hyperparameter	Value	Description
$\gamma_{\boldsymbol{x}}$	0.001	ADAM stepsize for the query $\boldsymbol{x}$
$\gamma_{\boldsymbol{w}}$	0.01	ADAM stepsize for the SVGP parameters $\boldsymbol{w}$
B	32	Minibatch size
G_{clip}	2.0	Gradient clipping threshold
k_{epochs}	30	Maximum number of epochs
n_{fail}	3	Maximum number of failure to improve
m	100	Number of inducing points
$n_0 = \lvert\mathcal{D}_0\rvert$	100	Number of observations for initializing BO
# quad.	20	Number of Gauss-Hermite quadrature points
`optimize_acqf: restarts`	10	
`optimize_acqf: raw_samples`	256	
`optimize_acqf: batch_size`	1/20	Depends on task; see details in Section 4

Hyperparameters. The hyperparameters used in our experiments are organized in Table 1. For the full-extent of the implementation details and experimental configuration, please refer to the supplementary code.

B Additional Plots

We provide additional results and plots that were omitted from the main text.

B.1 Additional Results on Molecule Tasks

In Fig. 4, we provide plots on additional results that are similar to those in Fig. 2. On three of the molecule tasks, we use 10,000 random molecules from the GuacaMol dataset as initialization. This is more consistent with what has been done in previous works and achieves better overall optimization performance.

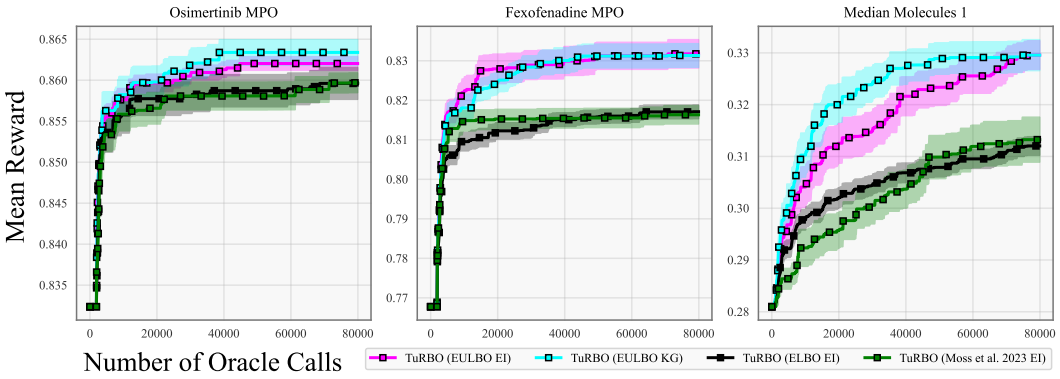

Figure 4: **Additional optimization results on three molecule tasks using 10,000 random molecules from the GuacaMol dataset as initialization**. Each line/shaded region represents the mean/standard error over 20 runs. We count oracle calls starting *after* these initialization evaluations for all methods.

B.2 Separate Plots for BO and TuRBO Results

In this section, we provide additional plots separating out BO and TuRBO results to make visualization easier.

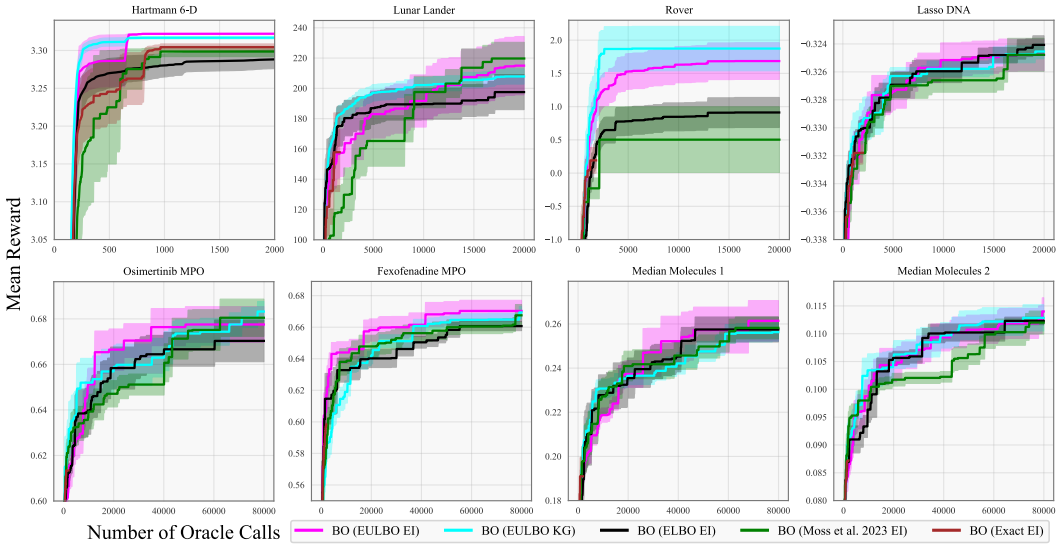

Figure 5: **BO-only optimization results of Fig. 2**. We compare EULBO-SVGP, ELBO-SVGP, ELBO-SVGP with DPP inducing point placement (Moss et al., 2023), and exact GPs. These are a subset of the same results shown in Fig. 2. Each line/shaded region represents the mean/standard error over 20 runs.

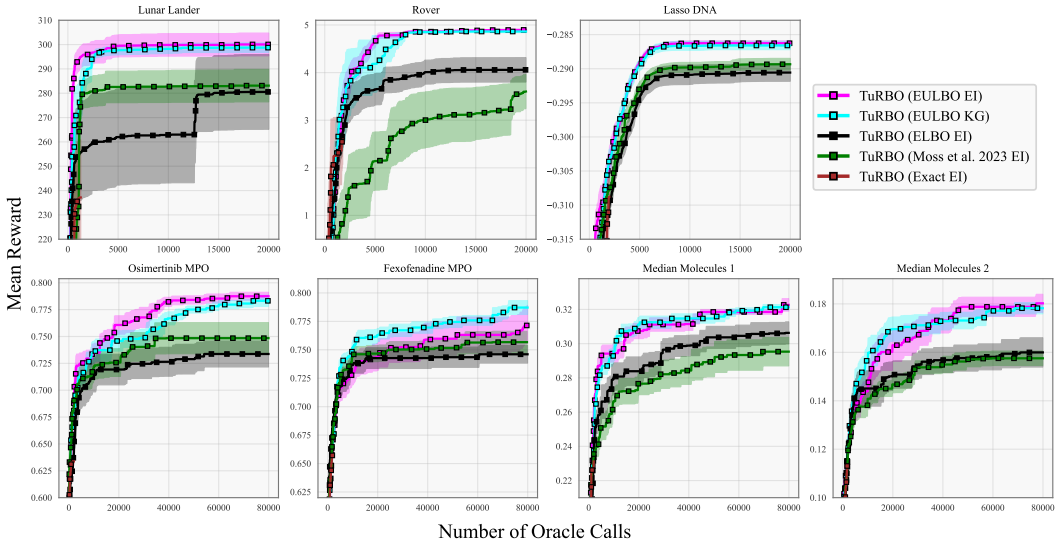

Figure 6: **TuRBO-only optimization results of Fig. 2.** We compare EULBO-SVGP, ELBO-SVGP, ELBO-SVGP with DPP inducing point placement (Moss et al., 2023), and exact GPs. These are a subset of the same results shown in Fig. 2. Each line/shaded region represents the mean/standard error over 20 runs.

B.3 Effect of Number of Inducing Points

For the results with approximate-GPs in Section 4, we used $m = 100$ inducing points. In Fig. 7, we evaluate the effect of using a larger number of inducing points ($m = 1024$) for EULBO-SVGP and ELBO-SVGP.

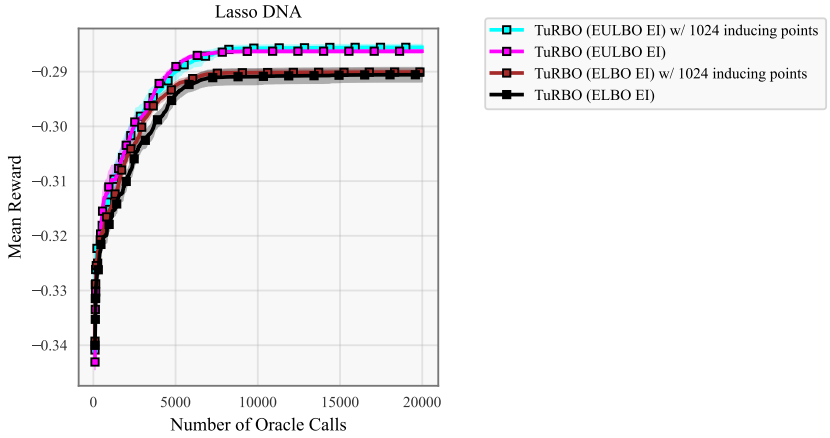

Figure 7: **Ablating the number of inducing points used by EULBO-SVGP and ELBO-SVGP.** As in Fig. 2, we compare running TuRBO with EULBO-SVGP and with ELBO-SVGP using $m = 100$ inducing points used for both methods. We add two additional curves for TuRBO with EULBO-SVGP and TuRBO with ELBO-SVGP using $m = 1024$ inducing points. Each line/shaded region represents the mean/standard error over 20 runs.

Fig. 7 shows that the number of inducing points has limited impact on the overall performance of TuRBO, and EULBO-SVGP outperforms ELBO-SVGP regardless of the number of inducing points used.

B.4 Effect of GP Objective

The results in Section 4 used a standard SVGP objective. In this section, we evaluate the effect of using an alternative objective: the parametric Gaussian process regressor (PPGPR; Jankowiak et al., 2020) objective. PPGPR differs from the standard SVGP objective in that the variational approximation is optimized to maximize the predictive accuracy instead of matching the posterior.

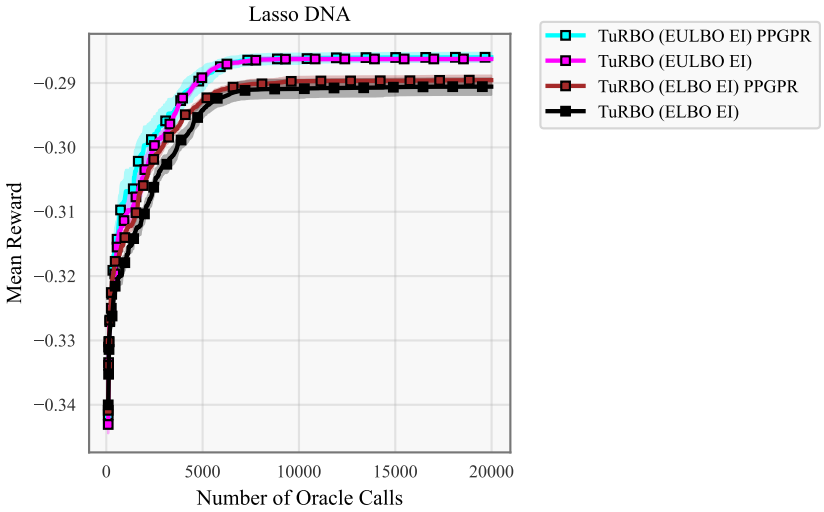

Figure 8: **Effect of using the PPGPR objective instead of the SVGP objective for `EULBO-EI` and ELBO-EI**. As in Fig. 2, we compare running `TuRBO` with `EULBO`-EI and with ELBO-EI using an SVGP model for both methods. We add two additional curves for `TuRBO` with `EULBO`-EI with a PPGPR model, and `TuRBO` with ELBO-EI using a PPGPR model. Each line/shaded region represents the mean/standard error over 20 runs.

We compare the choice of objective (PPGPR vs SVGP) in Fig. 8 and observe that the objective has limited impact on the overall performance of `TuRBO`. In particular, `EULBO`-EI outperforms ELBO-EI regardless of the GP objective.

C Compute Resources

Table 2: Internal Cluster Setup

Type	Model and Specifications
System Topology	20 nodes with 2 sockets each with 24 logical threads (total 48 threads)
Processor	1 Intel Xeon Silver 4310, 2.1 GHz (maximum 3.3 GHz) per socket
Cache	1.1 MiB L1, 30 MiB L2, and 36 MiB L3
Memory	250 GiB RAM
Accelerator	1 NVIDIA RTX A5000 per node, 2 GHZ, 24GB RAM

Type of Compute and Memory. All results in the paper required the use of GPU workers (one GPU per run of each method on each task). The majority of runs were executed on an internal cluster, where details are shown in Table 2, where each node was equipped with an NVIDIA RTX A5000 GPU. In addition, we used cloud compute resources for a short period leading up to the subsmission of the paper. We used 40 RTX 4090 GPU workers from `runpod.io`, where each GPU had approximately 24 GB of GPU memory. While we used 24 GB GPUs for our experiments, each run of our experiments only requires approximately 15 GB of GPU memory.

Execution Time. Each optimization run for non-molecule tasks takes approximately one day to finish. Since we run the molecule tasks out to a much larger number of function evaluations than other tasks (80000 total function evaluations for each molecule optimization task), each molecule optimization task run takes approximately 2 days of execution time. With all eight tasks, ten methods run, and 20 runs completed per method, results in Fig. 2 include 1600 total optimization runs (800 for molecule tasks and 800 for non-molecule tasks). Additionally, the two added curves in each plot in Fig. 3 required 160 additional runs (120 for molecule tasks and 40 for non-molecule task). Completing all of the runs needed to produce all of the results in this paper therefore required roughly 2680 total GPU hours.

Compute Resources Used During Preliminary Investigations. In addition to the computational resources required to produce experimental results in the paper discussed above, we spent approximately 500 hours of GPU time on preliminary investigations. This was done on the aforementioned internal cluster shown in Table 2.

D Wall-clock Run Times

In Table 3, we provide average wall-clock run times of different methods on the Lasso DNA optimization task.

Table 3: Average wall-clock run times for one full run of TuRBO on the Lasso DNA task. We compare the average wall-clock run time of TuRBO on all TuRBO methods from Figure 2. Note that we do not include the wall clock run time for TuRBO with Exact EI here because we only ran this method out to 2k oracle calls (rather than the full budget of 20k oracle calls).

Method	Wall-clock Run Time in Minutes
EULBO EI	267.30 ± 2.53
EULBO KG	296.95 ± 1.31
ELBO EI	184.40 ± 0.59
Moss et al. 20203 EI	194.32 ± 0.77

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All stated claims are backed-up with results in Section 4 and the stated focus/scope of the paper accurately reflects what is discussed throughout the rest of the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA] .

Justification: This work does not contain a formal theoretical analysis.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes] .

Justification: We provide detailed explanation of how our method works in Section 3 and all additional required details to reproduce results in Section 4 and Appendix A. Additionally, we have included a link to a public GitHub repository containing all of the source code used in the work in Section 4. This source code allows any reader to run our code to reproduce all results in the paper. Additionally, the README in the repository provides detailed instructions to make setting up the proper environment and running the code easy for users.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes] Replace by [Yes] , [No] , or [NA] .

   Justification: We have included a link to a public GitHub repository containing all of the source code used in the work in Section 4. This source code allows any reader to run our code to reproduce all results in the paper. Additionally, the README in the repository provides detailed instructions to make setting up the proper environment and running the code easy for users.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All chosen hyper-parameters and implementation details are stated in section 4 and Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: On all plots, we plot the mean taken over multiple random runs and include error bars to show the standard error over the runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and made sure to adhere to them in all aspects.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No] .

    Justification: The paper is methodological, where the considered algorithm does not immediately pose societal risks.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA] .

    Justification: The paper does not use data with potential societal concerns.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All creators of assets used to produce our results are cited in Section 4. All assets used are open source software or models.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA] .

    Justification: The paper does not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA] .

    Justification: The paper does not involve human participants.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA] .

    Justification: The paper does not involve live participants.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.