Robust Latent Space Bayesian Optimization with Marginalized Kernel

Ziyi Yang **Yifan Zhu**
South China University of Technology
{csziyi.yang, cszyf}@mail.scut.edu.cn

Jinghui Zhong*
South China University of Technology
jinghuizhong@scut.edu.cn

Abstract

Latent space Bayesian optimization circumvents the curse of dimensionality over structured inputs by performing Gaussian process surrogate-guided search in a continuous latent space learned by deep generative models. However, existing latent space Bayesian optimization methods overlook both the inherent stochasticity of latent space representations and the geometric properties of the manifold induced by VAE, leading to suboptimal optimization performance. In this work, we propose a marginalized kernel that integrates out the latent variables through the VAE inference posterior, yielding a closed-form kernel on the observation space that naturally accounts for encoding uncertainty. We adopt a quadratic polynomial kernel as the base kernel in the latent space, which captures discrepancies in the first two moments of the latent posteriors while avoiding the bandwidth sensitivity inherent to the RBF kernel. Furthermore, inspired by the Riemannian interpretation of VAEs, we incorporate a geometry-aware sampling scheme that leverages the metric structure revealed by the learned posterior covariances to guide candidate acquisition toward high-information regions of the latent space. Empirical evaluations on molecular design and robot design tasks demonstrate that our method outperforms existing state-of-the-art baselines across the majority of tasks.

1 Introduction

Bayesian optimization (BO) is a sample-efficient global optimization method designed for expensive black-box functions [1–3]. BO relies on a surrogate model and an acquisition function. The surrogate is typically a Gaussian process (GP) [4] that models the posterior distribution over the objective function to capture the predictive mean and uncertainty. The acquisition function then leverages these estimates to select the next evaluation point by balancing exploitation and exploration. Through this iterative loop, the method progressively approaches the optimum under a limited evaluation budget.

However, Bayesian optimization cannot be directly applied to structured input spaces, including molecular sequences [5, 6], graphs [7, 8], and strings [9, 10]. Beyond the discreteness of the search space, another key obstacle is that kernel functions defined in Euclidean spaces cannot be directly applied to such non-Euclidean structured objects. To overcome this limitation, Latent Space Bayesian Optimization (LSBO) [5, 11] incorporates deep generative models to learn compact representations of structured inputs. Specifically, the deep generative model used in LSBO typically consists of an encoder and a decoder. An encoder maps complex discrete inputs into a continuous latent space, where a Gaussian process is constructed to propose latent candidate points. The selected latent vectors are subsequently decoded back through a decoder into the original input space for evaluations of the true objective function. By reformulating discrete structured optimization as surrogate optimization in a continuous latent space, LSBO enables effective representation of complex data distributions and

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

improves the efficiency of proposing novel candidates. This paradigm has been widely adopted in de novo molecular design [12], automated robot design [13, 14], and circuit optimization [15, 16].

Despite the recent emergence of various improvements [17–21] for latent space Bayesian optimization that have demonstrated progress on several tasks [22, 23], existing methods still have not exploited the latent space structure learned by deep generative models, particularly the distributional information encoded in latent variables. Taking the Variational Autoencoder (VAE) [24] as an example, after encoding structured inputs into a posterior distribution over latent variables, most algorithms either draw a single reparameterized sample from the posterior or use the posterior mean as the latent representation for surrogate model training and acquisition optimization. This treatment improperly compresses distribution information into a single point estimate, causing the posterior statistics learned by the VAE to be discarded. As a result, the surrogate model cannot explicitly incorporate the uncertainty provided by the generative model, and the latent space search may be guided toward regions unsupported by labeled or even unlabeled training data, degrading the validity of decoded outputs and reducing the robustness of the optimization process.

To address this problem, we incorporate VAE representation uncertainty as distributional information in both surrogate modeling and candidate sampling. Unlike prior approaches that reduce each sample to a single latent vector z, we use the encoder's approximate posterior distribution as the representation. Its mean and variance are retained throughout the optimization loop to capture uncertainty and guide acquisition. More specifically, our contributions are as follows:

- We propose a marginalized kernel for VAEs that integrates out the encoder's approximate posterior to obtain a kernel over the structured input space. We adopt a quadratic polynomial base kernel, which captures discrepancies in the mean and covariance of the variational posteriors without requiring higher-order moment matching. The resulting kernel has a closed-form expression that incorporates the encoder's uncertainty into kernel evaluation.
- We introduce a latent geometry-aware sampling strategy for VAEs by interpreting the VAE encoder's learned posterior covariances as a Riemannian structure in the latent space. We construct a smooth global metric from local precision tensors and draw samples from the intrinsic Riemannian uniform distribution using a gradient-based Hamiltonian Monte Carlo (HMC) sampler (NUTS [25]) to efficiently avoid low-density informed regions.
- We integrate the proposed **mar**ginalized **k**ernel and latent geometry-aware sampling strategy to LS**BO** framework called **MarKBO** for optimizing expensive black-box functions over structured inputs via VAE latent spaces. We evaluate MarKBO on molecular optimization and automated robot design tasks. Empirical results show that MarKBO achieves competitive or superior performance against strong baselines. Ablation studies further validate the individual effectiveness of the proposed kernel and sampling strategy.

This paper is organized as follows. **Section 2** reviews related work on robust Bayesian optimization and latent space Bayesian optimization. **Section 3** introduces the background and notation for LSBO. **Section 4** presents our proposed method, including marginalized kernels for GP and latent geometry-aware sampling for acquisition. **Section 5** provides empirical evaluations on molecular design and robot design benchmark. **Section 6** concludes the paper and discusses limitations and societal impacts briefly. Additional experimental results and setup details, pseudocode, hyperparameters, extended discussions on limitations, and derivations of marginalized kernels are provided in the **Appendix**.

2 Related Work

Robust Bayesian Optimization Robust Bayesian Optimization (RBO) considers expensive black-box optimization under limited evaluation budgets when input-side uncertainty is present (e.g., execution noise [26, 27], contextual variability [28, 29], or distribution shift [30–32]). Methodologically, prior work can be broadly categorized into: (i) **surrogate-level robustness**, where Gaussian processes are defined directly over input distributions via distributional kernels [33, 34] or MMD-based kernels with Nyström approximations [35, 36]; (ii) **acquisition-level uncertainty propagation**, where input uncertainty is propagated into acquisition functions [26, 37] and (iii) **adversarial/distributionally robust formulations**, where uncertainty is modeled through perturbation or ambiguity sets and optimized via min–max criteria [38, 31, 30]. In this work, we adopt a surrogate-level approach

and propose a closed-form marginalized kernel that integrates out the VAE encoder's approximate posterior. This avoids approximations and naturally incorporates uncertainty into the GP surrogate.

Latent Space Bayesian Optimization Prior work on LSBO primarily improves upon the standard LSBO (Alg. 1) from the following four perspectives: (i) **Developing novel generative model architectures [39, 19, 40]**, NF-BO [19] adopts an autoregressive normalizing flow as the underlying deep generative model for LSBO, thereby addressing the reconstruction-error issue of VAEs. GenBO [40] presents a general strategy for turning generative models into candidate solution samplers for batch BO; (ii) **Designing fine-tuning strategies for generative models [41, 17, 42, 43].** The deep generative models trained on unlabeled data may yield a non-discriminative latent space that is ill-suited for optimization. Fine-tuning strategies address this by reshaping the latent space during the BO loop to better reflect the objective landscape [41, 17]. However, such fine-tuning strategies require careful balance of regularization strength; (iii) **Enhancing the surrogate model expressiveness [44–47].** A standard GP surrogate fitted in a pre-trained latent space may inadequately capture the mapping from latent space to objective values. Deep Kernel Learning (DKL [48]) addresses this by passing latent vectors through an MLP before feeding them into the GP kernel. Jointly training the generative model and the GP under a unified objective further aligns the latent geometry with the surrogate's smoothness prior [45, 47], and semi-supervised techniques such as pseudo-labeling additionally enrich the training signal by exploiting unlabeled data [42, 46]; (iv) **Improving acquisition strategies [49–51].** LSBO is closely connected to high-dimensional Bayesian optimization (HDBO). Despite the dimensionality reduction offered by generative models, the latent space typically remains high-dimensional [45]. Local Bayesian optimization [49, 51] thus serves as a natural strategy that avoids over-exploration without imposing simplifying assumptions on the objective.

3 Background

In this section, we provide a brief overview of the core components of LSBO. Additional background on the methods introduced in Section 4.1 and 4.2 can be found in the Appendix B and C.

In latent space Bayesian optimization (LSBO), let $f : \mathcal{X} \to \mathbb{R}$ denote an expensive black-box objective function defined over a structured domain $\mathcal{X}$. LSBO employs a deep generative model to learn a continuous latent space $\mathcal{Z} \subset \mathbb{R}^d$ and performs Bayesian optimization therein. At iteration t, the algorithm maintains an observation dataset $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^t$ with $y_i = f(\boldsymbol{x}_i) + \varepsilon_i$ and a corresponding latent dataset $\mathcal{D}_t^{\boldsymbol{z}} = \{(\boldsymbol{z}_i, y_i)\}_{i=1}^t$, where $\boldsymbol{z}_i \in \mathcal{Z}$ is the latent representation of $\boldsymbol{x}_i$. A surrogate model is first fitted on $\mathcal{D}_t^{\boldsymbol{z}}$, after which an acquisition function is maximized to select a new latent point $\boldsymbol{z}_{t+1}$. After selecting $\boldsymbol{z}_{t+1}$, a new candidate is generated via $\boldsymbol{x}_{t+1} \sim p_\theta(\boldsymbol{x} \mid \boldsymbol{z}_{t+1})$ and evaluated to obtain y_{t+1}. Both $\mathcal{D}_{t+1}$ and $\mathcal{D}_{t+1}^{\boldsymbol{z}}$ are then updated accordingly.

Deep Generative Model. Most LSBO methods adopt VAEs to establish a probabilistic mapping between the structured space $\mathcal{X}$ and the continuous latent space $\mathcal{Z} \subset \mathbb{R}^d$. The encoder parameterizes an approximate posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}\big(\mu_\phi(\boldsymbol{x}), \boldsymbol{\Sigma}_\phi(\boldsymbol{x})\big)$, the decoder parameterizes a generative distribution $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$, and the prior is set to a standard Gaussian $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The model is trained by maximizing the evidence lower bound (ELBO) [24]. Latent samples are obtained via the reparameterization trick $\boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) \odot \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Surrogate Model. A surrogate model is trained on $\mathcal{D}_t^{\boldsymbol{z}} = \{(\boldsymbol{z}_i, y_i)\}_{i=1}^t$ with $\boldsymbol{z}_i \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_i)$ to capture the statistical relationship between latent representations and observations. Gaussian process posterior provides a predictive mean $\mu_t(\mathbf{z})$ and a predictive variance $\sigma_t^2(\boldsymbol{z})$ at each $\boldsymbol{z}$. In larger-scale settings, sparse or variational GP approximations [52, 45] are often adopted to improve scalability.

Acquisition Strategy. The acquisition function $\alpha_t(\boldsymbol{z})$ is derived from the surrogate posterior and serves to select the next latent query point $\boldsymbol{z}_{t+1} \in \arg\max_{\boldsymbol{z} \in \mathcal{Z}} \alpha_t(\boldsymbol{z})$. It leverages $\mu_t(\boldsymbol{z})$ and $\sigma_t^2(\boldsymbol{z})$ to balance exploration and exploitation. A common choice is expected improvement (EI) [53, 54], which scores each candidate by its expected gain over the current best observation. An alternative is Thompson sampling (TS) [55], which draws a sample function $f_t \sim \mathcal{GP}(\mu_t, \Sigma_t)$ from the surrogate posterior and selects $\boldsymbol{z}_{t+1} = \arg\max_{\boldsymbol{z} \in \mathcal{Z}} f_t(\boldsymbol{z})$. In practice, the maximization is approximated by evaluating f_t over a finite set of candidate points. In high-dimensional settings, Papenmeier et al. [51] demonstrate that random axis-aligned subspace perturbations (RAASP) of the incumbent can improve the quality of these candidates by promoting local search behavior.

4 Method

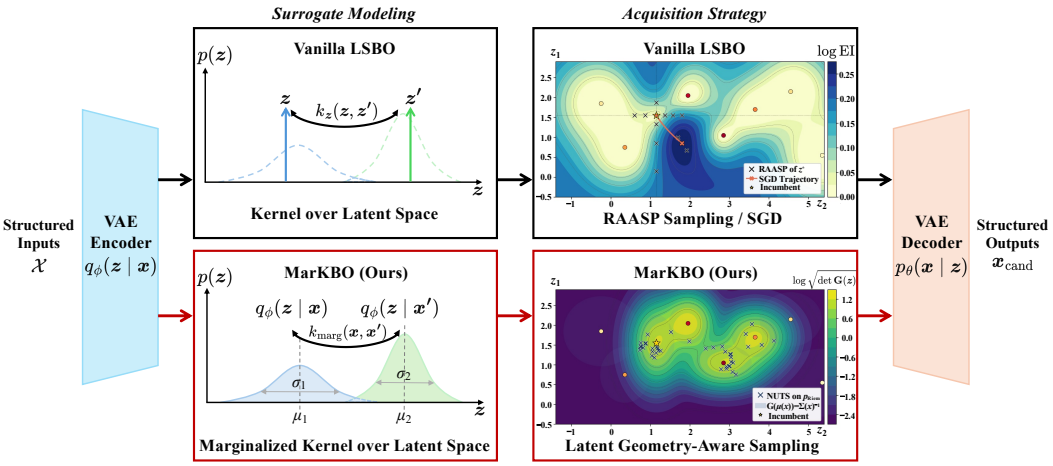


Figure 1: Overview of MarKBO (Alg. 3). Structured inputs $\mathcal{X}$ are mapped by the VAE encoder $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ to latent posterior distributions, and selected candidates are decoded back through $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ into structured outputs. **Left:** Vanilla LSBO constructs a GP surrogate using a kernel evaluated at reparameterized latent samples, whereas MarKBO integrates the kernel over the encoder posteriors to incorporate their means and covariances into surrogate modeling. **Right:** The upper panel illustrates RAASP candidates and an SGD optimization trajectory on the logEI acquisition function landscape. In contrast, MarKBO constructs a Riemannian metric $\mathbf{G}(\boldsymbol{z})$ from the encoder's posterior precisions and uses NUTS to draw latent candidates from the corresponding volume density $p_{\mathrm{Riem}}(\boldsymbol{z})$ (Alg. 2).

In this section, we introduce our approach that exploits the uncertainty captured by the VAE encoder to improve both GP surrogate modeling (Section 4.1) and latent candidate acquisition (Section 4.2). As illustrated in Fig. 1, the VAE encoder induces a deep kernel GP over structured inputs through its variational posteriors and provides the covariance information used for geometry-aware sampling, while the VAE decoder generates structured candidates for black-box objective evaluation.

4.1 Marginalized Kernel

The marginalized kernel $k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}')$ was introduced by Tsuda et al. [56] as a principled approach to designing kernel functions for data generated by latent variable models. The core idea is to define a joint kernel on the complete data comprising both observed and latent variables and then obtain a kernel on the observed data alone by marginalizing out the latent variables. This framework has since been extended to graphs [57–59]. Formally, let $\boldsymbol{x}$ denote the observed variable, $\boldsymbol{z}$ the latent variable, and $(\boldsymbol{x}, \boldsymbol{z})$ the complete data. Let $k_{\mathrm{joint}}\big((\boldsymbol{x}, \boldsymbol{z}),\ (\boldsymbol{x}', \boldsymbol{z}')\big)$ be a positive definite kernel on the complete data space. When the latent variable is discrete, the marginalized kernel is defined as

$$k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{z}} \sum_{\boldsymbol{z}'} p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{z}' \mid \boldsymbol{x}')\, k_{\mathrm{joint}}\big((\boldsymbol{x}, \boldsymbol{z}),\ (\boldsymbol{x}', \boldsymbol{z}')\big). \tag{1}$$

When the latent variable is continuous, the summations are replaced by integrals accordingly:

$$k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}') = \iint p(\boldsymbol{z} \mid \boldsymbol{x})\, p(\boldsymbol{z}' \mid \boldsymbol{x}')\, k_{\mathrm{joint}}\big((\boldsymbol{x}, \boldsymbol{z}),\ (\boldsymbol{x}', \boldsymbol{z}')\big)\, \mathrm{d}\boldsymbol{z}\, \mathrm{d}\boldsymbol{z}'. \tag{2}$$

Since this operation amounts to averaging a positive definite kernel with respect to a non-negative measure, the resulting kernel $k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}')$ is guaranteed to be positive definite.

The key design challenge in this framework is the specification of the joint kernel k_{joint}. When the input $\boldsymbol{x}$ is a structured object such as a graph or string, the joint kernel should consider both the combinatorial structure of $\boldsymbol{x}$ and the semantics of $\boldsymbol{z}$. Existing approaches typically rely on domain-specific decomposition strategies [56, 57] and incur additional computational cost. In VAE-LSBO (Alg. 1) setting, the VAE's inference network $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ has already compressed the structural and

semantic information of $\boldsymbol{x}$ into the latent code $\boldsymbol{z}$. The joint kernel therefore need not explicitly model the coupling between input and latent variables. We define it to depend solely on the latent variables:

$$k_{\text{joint}}\big((\boldsymbol{x}, \boldsymbol{z}),\ (\boldsymbol{x}', \boldsymbol{z}')\big) := k_{\boldsymbol{z}}(\boldsymbol{z}, \boldsymbol{z}'), \tag{3}$$

which induces the following marginalized kernel over the observation space:

$$k_{\text{marg}}(\boldsymbol{x}, \boldsymbol{x}') := \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x}),\ \boldsymbol{z}'\sim q_\phi(\boldsymbol{z}'|\boldsymbol{x}')}\Big[\, k_{\boldsymbol{z}}(\boldsymbol{z}, \boldsymbol{z}')\Big]. \tag{4}$$

Under the standard Gaussian variational family $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}\big(\mu_\phi(\boldsymbol{x}),\ \Sigma_\phi(\boldsymbol{x})\big)$, the expectation in Eq. (4) admits closed-form expressions for several common base kernels $k_{\boldsymbol{z}}$ including the linear, the quadratic polynomial, and the Gaussian RBF kernels [60]. The marginalized kernel $k_{\text{marg}}(\boldsymbol{x}, \boldsymbol{x}')$ can therefore be evaluated exactly without Monte Carlo approximation.

In Bayesian optimization the GP surrogate induces a posterior mean and variance on which most acquisition functions directly depend. The resulting acquisition landscape is shaped by the kernel hyperparameters. Consider the RBF base kernel $k_{\boldsymbol{z}}(\boldsymbol{z}, \boldsymbol{z}') = \exp\big(-\|\boldsymbol{z} - \boldsymbol{z}'\|^2/(2\ell^2)\big)$ widely used in LSBO [45, 18, 19]. As $\ell \to \infty$ the kernel matrix approaches a constant and the acquisition landscape flattens to produce vanishing gradients [61, 62, 51]. As $\ell \to 0$ the posterior collapses to the prior away from observed points. The RBF kernel also possesses an infinite-dimensional feature map that implicitly matches moments of all orders. In our setting the objects being compared are Gaussian variational posteriors that are fully determined by their first two moments. A finite-dimensional kernel that captures interactions up to second order therefore already provides a rich characterization of inter-distribution similarity. We adopt a quadratic polynomial base kernel

$$k_{\boldsymbol{z}}(\boldsymbol{z}, \boldsymbol{z}') = (\boldsymbol{z}^\top \boldsymbol{z}' + c)^2, \tag{5}$$

where $c \geq 0$ controls the relative weight of first- and second-order terms. Its feature map contains only terms up to degree two and involves no bandwidth parameter ℓ to tune. Substituting this base kernel into Eq. (4) and applying standard Gaussian moment identities yields

$$k_{\text{marg}}(\boldsymbol{x}, \boldsymbol{x}') \;=\; \operatorname{tr}\big(M_\phi(\boldsymbol{x})\, M_\phi(\boldsymbol{x}')\big) \;+\; 2c\, \mu_\phi(\boldsymbol{x})^\top \mu_\phi(\boldsymbol{x}') \;+\; c^2, \tag{6}$$

where $M_\phi(\boldsymbol{x}) := \Sigma_\phi(\boldsymbol{x}) + \mu_\phi(\boldsymbol{x})\, \mu_\phi(\boldsymbol{x})^\top \in \mathbb{R}^{d\times d}$ denotes the second-moment matrix of $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and d is the latent dimensionality. A detailed derivation of Eq. (6) is provided in Appendix B.3. In practice VAE encoders commonly parameterize diagonal covariances $\Sigma_\phi(\boldsymbol{x}) = \operatorname{diag}\big(\sigma_\phi^2(\boldsymbol{x})\big)$. The trace then reduces to a standard inner product of d-dimensional vectors and each kernel evaluation costs $\mathcal{O}(d)$ operations. Constructing the full $n \times n$ Gram matrix therefore requires $\mathcal{O}(n^2 d)$ time, matching the cost of a standard RBF kernel in $\mathbb{R}^d$.

Eq. (6) shows that the marginalized kernel compares the first two moments of two variational posteriors. The trace term $\operatorname{tr}\big(M_\phi(\boldsymbol{x})\, M_\phi(\boldsymbol{x}')\big)$ measures agreement in both the location and spread of the latent distributions. The constant c controls the relative contribution of the linear term $\mu_\phi(\boldsymbol{x})^\top \mu_\phi(\boldsymbol{x}')$ that captures mean similarity. Because the encoder covariances $\Sigma_\phi(\boldsymbol{x})$ and $\Sigma_\phi(\boldsymbol{x}')$ enter through the second-moment matrices, the uncertainty of the latent representation is incorporated into kernel evaluation. In regions where the encoder is less confident the larger covariance inflates the second-moment matrix and produces a smoothing effect that propagates into the GP posterior.

4.2 Latent Geometry-Aware Sampling

Effective exploration of the latent space is another key challenge in latent space Bayesian optimization. Although the ELBO loss encourages each per-sample posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to match the prior $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ via the KL term, after training the aggregate posterior $q_\phi(\boldsymbol{z}) = \int q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\, p_{\text{data}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$ still differs from $p(\boldsymbol{z})$. This mismatch creates *prior holes* where regions that carry non-negligible probability under the prior yet are rarely visited by any encoded data point [63, 64]. Decoding samples drawn from these holes produces low-quality or invalid outputs [65, 50]. Several works mitigate over-exploration by constraining the search space [45] or refining the acquisition function [65, 50]. Moss et al. [21] use a modified PCN Metropolis MCMC to sample latent vectors and deterministically decode each $\boldsymbol{z}$ to its most likely structure $\hat{\boldsymbol{x}}(\boldsymbol{z}) = \arg\max_{\boldsymbol{x}\in\mathcal{X}} p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$, and accept proposals using the VAE Gaussian prior together with a probability of improvement (PI)-like likelihood computed from a structured-space GP while initializing from the best-so-far latent and adapting the step size. Nevertheless, their method uses the standard Gaussian prior as the base distribution of the transition kernel and is therefore still susceptible to prior holes.

A subtler yet overlooked issue concerns the geometry of the latent space. Specifically, we often take it for granted that the geometry of the latent space is Euclidean. However, since the decoder of a VAE is a highly nonlinear mapping from a low-dimensional latent space to a high-dimensional observation space, it induces spatially varying distortions. Equal-length steps in different latent regions can correspond to different distances in observation space. This phenomenon has a differential-geometric interpretation [66–68]. Tosi et al. [69] showed that the generative mapping of a GP latent variable model (GPLVM) endows the latent space with a pull-back Riemannian metric through the Jacobian of the mapping from the latent space to the observation space. Arvanitidis et al. [67] extended this idea to VAEs and demonstrated that the resulting stochastic Riemannian metric captures local distortions induced by the generator. Subsequent studies confirmed that treating the latent space as a Riemannian manifold yields more accurate distances and improves downstream tasks such as interpolation and clustering [67, 68]. A practical limitation is that computing the decoder Jacobian is expensive for high-dimensional observations. Notably, Chadebec and Allassonnière [70] observed that the covariance structure learned by the variational posterior can itself define a latent space metric closely related to the pullback metric. This metric can be obtained directly from the pretrained encoder without explicit Jacobian computation.

Motivated by both observations, we construct a latent space metric that reflects model uncertainty by leveraging the encoder outputs $\boldsymbol{\mu}(\boldsymbol{x}_i)$ and $\boldsymbol{\Sigma}(\boldsymbol{x}_i)$ available on the labeled dataset, following the geometric framework of [70]. We replace the fixed Gaussian prior with a sampling distribution induced by the corresponding Riemannian volume element so that candidate generation concentrates on regions well supported by the aggregate posterior and avoids invalid latent holes.

Specifically, we assume a Gaussian encoder posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}\big(\mu_\phi(\boldsymbol{x}), \boldsymbol{\Sigma}_\phi(\boldsymbol{x})\big)$. For each training sample $\boldsymbol{x}_i$, we interpret posterior precision $\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)$ as the value of a local Riemannian metric tensor at the latent point $\mu(\boldsymbol{x}_i)$: $\mathbf{G}(\mu(\boldsymbol{x}_i)) = \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)$. Intuitively, sampling $\boldsymbol{z} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_i)$ during VAE training can be viewed as sampling under a locally constant metric around $\mu(\boldsymbol{x}_i)$. Consequently, the scale of the covariance $\boldsymbol{\Sigma}(\boldsymbol{x}_i)$ determines which perturbation directions in latent space are more likely to be regarded as semantically close to $\boldsymbol{x}_i$.

Since we only observe metric information at finitely many points $\{\boldsymbol{\mu}(\boldsymbol{x}_i)\}$, we construct a smooth metric $\mathbf{G}(\boldsymbol{z})$ over the entire latent space by a weighted interpolation of $\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)$ and add an isotropic term that decays with $\|\boldsymbol{z}\|_2$ to keep the overall volume finite:

$$\mathbf{G}(\boldsymbol{z}) = \sum_{i=1}^{K} \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)\, \omega_i(\boldsymbol{z}) \;+\; \lambda\, e^{-\tau \|\boldsymbol{z}\|_2^2}\, \mathbf{I}_d\,, \tag{7}$$

where K is the number of selected centroids, $\lambda > 0$ is a small regularization constant, and $\tau > 0$ is chosen close to zero (10^{-6}) so that the isotropic regularization term in Eq. (7) takes effect only far from the data-supported region. The interpolation weights $\omega_i(\boldsymbol{z})$ are defined by a kernel based on the local Mahalanobis distance from $\boldsymbol{z}$ to the i-th centroid:

$$\omega_i(\boldsymbol{z}) = \exp\!\left(-\frac{(\boldsymbol{z} - \boldsymbol{\mu}(\boldsymbol{x}_i))^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)\,(\boldsymbol{z} - \boldsymbol{\mu}(\boldsymbol{x}_i))}{\rho^2}\right). \tag{8}$$

where the bandwidth ρ controls the spatial extent of each centroid's influence on the global metric. Following [70], we set ρ to the maximum nearest-neighbor Euclidean distance among the selected centroids, i.e. $\rho = \max_i \min_{j \neq i} \|\boldsymbol{\mu}(\boldsymbol{x}_i) - \boldsymbol{\mu}(\boldsymbol{x}_j)\|_2$, which ensures sufficient overlap between neighboring kernels and yields a smooth interpolation of the metric field. Recent advances in HDBO have shown that concentrating search around the best-performing observations is essential for scalability [49, 51]. Drawing on this insight, we rank the N labeled samples by their observed objective values y and retain the top $p\%$ as centroids $\{\boldsymbol{x}_i\}_{i=1}^{K}$ with $K = \lceil pN/100 \rceil$. K is determined by the initial labeled dataset size and remains fixed. This focuses the metric construction on the most promising region of the latent space and naturally biases the sampler toward high-quality solutions.

Given $\mathbf{G}(\boldsymbol{z})$, we sample from the distribution induced by the associated Riemannian volume element. The lower-right panel of Fig. 1 illustrates the interpolated metric in a 2-D latent space with top-K points, where the shading reflects the local volume element $\sqrt{\det \mathbf{G}(\boldsymbol{z})}$. The Riemannian measure has volume form $\sqrt{\det \mathbf{G}(\boldsymbol{z})}\, \mathrm{d}\boldsymbol{z}$, and so the target density with respect to the Lebesgue measure is

$$p_{\text{Riem}}(\boldsymbol{z}) = \frac{\sqrt{\det \mathbf{G}(\boldsymbol{z})}}{\int_{\mathbb{R}^d} \sqrt{\det \mathbf{G}(\boldsymbol{z})}\, \mathrm{d}\boldsymbol{z}}\,. \tag{9}$$

This density assigns higher probability mass to regions where $\sqrt{\det \mathbf{G}(\boldsymbol{z})}$ is large. Because $\mathbf{G}(\boldsymbol{z})$ is built by interpolating local metrics from the top-performing samples, those high-density regions naturally coincide with neighborhoods of high-quality solutions that possess well-supported geometry. Sampling is performed using NUTS sampler [25]. The potential energy is

$$U(\boldsymbol{z}) = -\log p_{\text{Riem}}(\boldsymbol{z}) = -\tfrac{1}{2}\log\det \mathbf{G}(\boldsymbol{z}) + \text{const}\,, \tag{10}$$

and the required gradient can be written as

$$\frac{\partial \log p_{\text{Riem}}(\boldsymbol{z})}{\partial z_j} = \tfrac{1}{2}\,\text{tr}\left(\mathbf{G}(\boldsymbol{z})^{-1}\frac{\partial \mathbf{G}(\boldsymbol{z})}{\partial z_j}\right). \tag{11}$$

Here $\partial \mathbf{G}(\boldsymbol{z})/\partial z_j$ can be obtained via automatic differentiation [71] of the closed-form expression in Eq. (7). The terms $\log\det \mathbf{G}(\boldsymbol{z})$ and $\mathbf{G}(\boldsymbol{z})^{-1}\boldsymbol{v}$ can be computed using a Cholesky factorization and linear solves. The normalization constant in Eq. (9) cancels out in both the gradient Eq. (11) and the Metropolis acceptance ratio and therefore need not be computed explicitly.

4.3 MarKBO

Combining the marginalized kernel of Sec. 4.1 with the latent geometry-aware sampling of Sec. 4.2, we propose **MarKBO** (Latent Space **B**ayesian **O**ptimization with **Mar**ginalized **K**ernel). At each iteration MarKBO fits a GP surrogate using the marginalized kernel (Eq. 4) over the labeled dataset and generates candidate points by drawing samples from the Riemannian volume element (Eq. 9) via NUTS [25]. The next query is then selected by maximizing the acquisition function over these candidates. The marginalized kernel enables the GP to account for encoder uncertainty without additional approximation. The latent geometry-aware sampler focuses exploration on well-supported and high-performing latent regions while avoiding prior holes. Both components are modular and require no additional fine-tuning of the pretrained VAE. We provide the full pseudocode along with implementation details and hyperparameter settings in Appendix F.

5 Experiment

5.1 Tasks

PMO Benchmark [72] We evaluate our method on 16 challenging tasks from the Practical Molecular Optimization (PMO) benchmark [72], which builds upon the GuacaMol benchmark [22] and the Therapeutics Data Commons (TDC) [23]. These tasks span several categories: similarity and rediscovery tasks (e.g., Albuterol Similarity, Celecoxib Rediscovery), multi-property optimization targets (e.g., Amlodipine MPO), and biological activity predictors (DRD2, GSK3β). Most oracles return scores in $[0.0, 1.0]$ with higher values indicating more desirable properties. All oracles are accessed through the `poli` library [73], which provides a unified interface for querying black-box objectives using SMILES or SELFIES [74] molecular representations.

Similar to NF-BO [19], we consider a small-budget setting with 100 initial points and 500 additional calls at a batch size of 5, and a large-budget setting with 10,000 initial points and 10,000 additional calls at a batch size of 100. We use the first N molecules from the Guacamol test set [22] as initial data. We report four small-budget tasks (Amlodipine MPO, Perindopril MPO, Fexofenadine MPO, Zaleplon MPO) and two large-budget tasks (Osimertinib MPO, Valsartan SMARTS) in the main text. Results on an additional 12 small-budget and 4 large-budget PMO tasks are provided in Appendix D.

Automated Robot Design [14, 75] We consider the morphology design tasks from [14], where the goal is to automatically discover robot body structures that maximize task reward under a limited budget of expensive evaluations. Candidate robots are represented as acyclic design graphs assembled from a discrete component library, with graph-grammar constraints [75] restricting the combinatorial search space to valid designs. For any proposed morphology, performance is measured by the reward achieved in physics simulation after solving a nested control problem via model predictive path integral control (MPPI). The experiment comprises four locomotion tasks defined by their terrains: Flat Terrain, Frozen Lake Terrain, Ridged Terrain, and Wall Terrain. We initialize each run with the first 100 designs from the GLSO-VAE training dataset [14] and evaluate a batch of 5 candidates per iteration under a total budget of 500 evaluations. Methods are compared by tracking the cumulative best reward over the number of oracle evaluations. Further details on the GLSO-VAE architecture, training procedure, initial data generation, and hyperparameters are provided in Appendix E.1.2.

5.2 Experimental Results

For the PMO benchmark, we conduct a comparison between our proposed algorithm and the following categories of methods: non-Bayesian optimization method (Graph-GA [76]), recently proposed VAE-based latent space Bayesian optimization methods (CoBO [17], InvBO [18], PG-LBO [42], COWBOYS [21]), normalizing flow-based latent space Bayesian optimization method (NF-BO [19]), recently proposed robust Bayesian optimization method (AIRBO [36]) and recently proposed high-dimensional Bayesian optimization methods (MSR [51], LinearBO [77]). All VAE-LSBO, HDBO and RBO methods share the same SELFIES-VAE [45] as the generative model and employ a sparse variational GP [45] as the surrogate model. For the large-budget PMO setting, we exclude AIRBO and COWBOYS from the comparison due to the scalability limitations of their kernel functions when handling large datasets. To ensure fair comparison, we modify the acquisition strategies of certain baselines. Further details on baselines are provided in the Appendix E.4.

For the Automated Robot Design task, in addition to the baselines compared in the GLSO paper [14] (LSBO [14], Random Search, MCTS [75], GHS [75], GA [78]), we further include the state-of-the-art VAE-LSBO and HDBO method (InvBO, AIRBO, MSR, LinearBO) as an additional comparison. Since PG-LBO and CoBO require fine-grained hyperparameter tuning for the regularization term, and COWBOYS relies on specific structured kernel, we did not include these methods in the baselines.

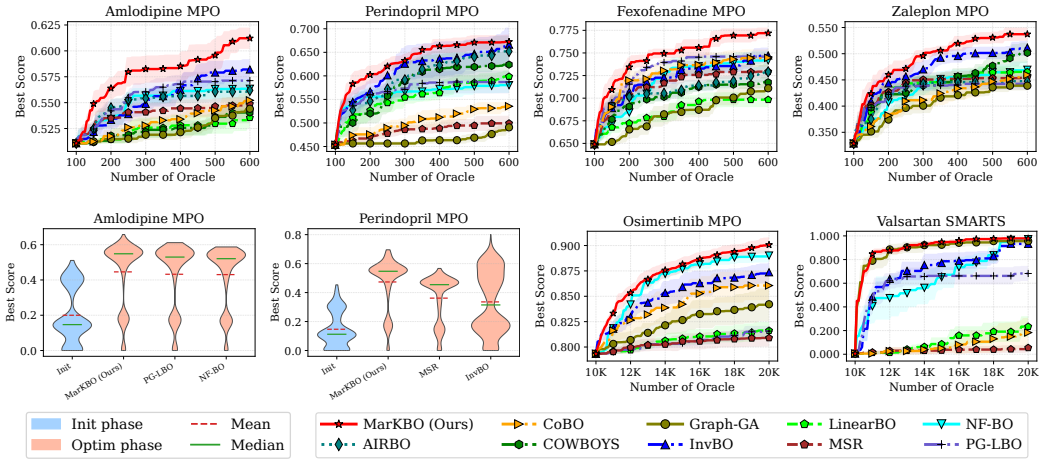


Figure 2: Results on PMO benchmark over 10 seeds. **Top row:** optimization curves on four small-budget tasks. **Bottom left:** violin plots of final scores on two small-budget tasks; green lines indicate medians, red lines indicate means. **Bottom right:** optimization curves on two large-budget tasks.

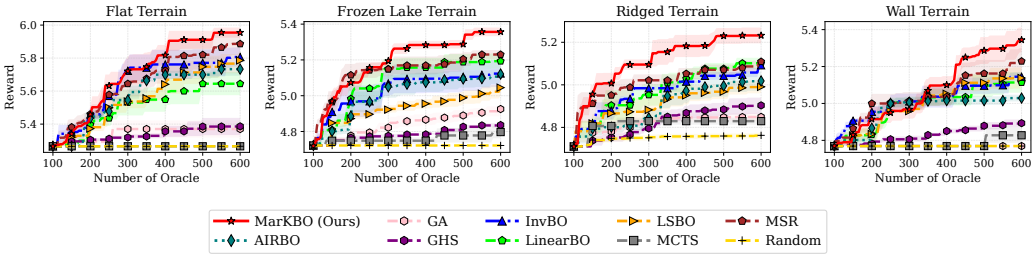


Figure 3: Performance of MarKBO against nine baseline methods on four locomotion tasks. The solid line shows the mean reward over 10 random seeds, and the shaded region indicates the standard deviation. The x-axis denotes the number of objective function evaluations.

Figure 2 summarizes the results on the PMO benchmark. In the top row, MarKBO achieves the highest best score across all four small-budget tasks with faster convergence. The bottom-left violin plots present the oracle score distributions over 10 seeds for deduplicated molecules generated by the top-3 baselines ranked by median score. Researchers prefer to obtain a set of high-scoring and structurally diverse candidates to improve the success rate of downstream experimental validation. MarKBO produces molecules concentrated in a higher score range, indicating robust generation of

diverse candidates rather than reliance on a few high-scoring outliers, which is important for practical de novo molecular design. Violin plots including all baselines are provided in Appendix D. In the bottom-right large-budget setting, MarKBO maintains its advantage over baselines. This scalability benefits from two design choices. The marginalized kernel admits a closed-form expression that avoids costly approximation at each kernel matrix computation. The latent geometry-aware HMC sampling strategy operates directly on the pretrained VAE latent space without requiring retraining. Full results on additional 12 small-budget and 4 large-budget tasks are provided in Appendix D, where MarKBO similarly achieves competitive or superior performance.

Figure 3 shows the cumulative best reward over oracle evaluations on four locomotion terrains tasks. MarKBO achieves the highest final reward on all four tasks and converges faster than every baseline. Classical combinatorial search methods such as GA and MCTS generally underperform compared to latent space Bayesian optimization approaches across all tasks.

5.3 Ablation and Component Analysis

To validate the effect of the marginalized kernel on LSBO performance, we conduct an ablation study on four small-budget PMO tasks from Sec. 5.1. We compare three groups of methods: (1) the proposed marginalized kernels, including Quadratic, Linear, RBF (isotropic) and RBF (ARD) variants; (2) the corresponding standard kernels without marginalization that operate on latent variables obtained via reparameterized sampling; (3) additional kernels including the Maximum Mean Discrepancy (MMD) kernel adopted by AIRBO [36], the Tanimoto kernel adopted by COWBOYS [21] and the Deep Kernel Learning (DKL) [48] + RBF kernel adopted by LOL-BO [45]. For ARD kernels, following MSR [51], we initialize length scales with a dimension-scaled scheme ($1/10\sqrt{d}$) and learn them via maximum likelihood estimation. All methods adopt the sampling strategy introduced in Section 4.2.

Table 1: Effect of kernel marginalization on four small-budget PMO tasks under the same Latent Geometry-Aware Sampling strategy. Results are reported as mean $\pm$ std over 10 random seeds.

Kernel	Amlodipine MPO	Perindopril MPO	Fexofenadine MPO	Zaleplon MPO
Marginalized Quadratic Kernel	$\mathbf{0.615 \pm 0.010}$	$\mathbf{0.673 \pm 0.015}$	$\mathbf{0.772 \pm 0.006}$	$\mathbf{0.521 \pm 0.025}$
Marginalized Linear Kernel	0.591 ± 0.009	0.635 ± 0.016	0.751 ± 0.004	0.498 ± 0.013
Marginalized RBF Kernel	0.572 ± 0.015	0.615 ± 0.013	0.742 ± 0.008	0.485 ± 0.010
Marginalized RBF Kernel (ARD)	0.547 ± 0.016	0.568 ± 0.025	0.726 ± 0.017	0.454 ± 0.007
Quadratic Kernel	0.583 ± 0.014	0.561 ± 0.015	0.693 ± 0.010	0.482 ± 0.023
Linear Kernel	0.579 ± 0.011	0.584 ± 0.018	0.714 ± 0.006	0.471 ± 0.020
RBF Kernel	0.564 ± 0.013	0.572 ± 0.006	0.691 ± 0.021	0.466 ± 0.012
RBF Kernel (ARD)	0.556 ± 0.008	0.540 ± 0.007	0.672 ± 0.005	0.423 ± 0.014
MMD Kernel [36]	0.547 ± 0.005	0.585 ± 0.014	0.742 ± 0.018	0.469 ± 0.005
Tanimoto Kernel [21]	0.575 ± 0.007	0.601 ± 0.012	0.735 ± 0.013	0.491 ± 0.016
DKL + RBF Kernel [45]	0.562 ± 0.012	0.607 ± 0.017	0.738 ± 0.022	0.483 ± 0.015

The final top-1 scores are summarized in Table 1. Overall, marginalization yields consistent improvements across all kernel types, confirming the effectiveness of incorporating encoder uncertainty into kernel evaluation. Among all variants, the Marginalized Quadratic Kernel achieves the best overall performance. As analyzed in Section 4.1, the variational posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\Sigma}_\phi(\boldsymbol{x}))$ is fully characterized by its first and second moments, and the quadratic polynomial kernel naturally captures discrepancies at the level of means and covariances without implicitly matching all-order moments as the RBF kernel does. Moreover, the quadratic kernel avoids the bandwidth ℓ tuning sensitivity of the RBF kernel, which is prone to vanishing likelihood gradients in high-dimensional settings. Similarly, the ARD variants do not outperform their isotropic counterparts, as the abundance of per-dimension length-scale parameters increases model selection difficulty under limited data.

We also ablate the candidate sampling strategy on the same four small-budget PMO tasks in Sec. 5.1. We compare four approaches: (1) the proposed Latent Geometry-Aware Sampling from Section 4.2; (2) Gaussian Prior Sampling, which draws candidates directly from the VAE prior; (3) PCN-MCMC Sampling adopted by COWBOYS [21]; (4) RAASP Sampling adopted by MSR [51], which perturbs the incumbent along random axis-aligned subspaces. All methods use the Marginalized Quadratic Kernel for GP fitting. The final top-1 score results in Table 2 show that Latent Geometry-Aware Sampling outperforms the other strategies, suggesting that leveraging the local geometric structure of the learned latent space provides more informative candidates for acquisition.

Table 2: Effect of candidate sampling strategies on four small-budget PMO tasks using the Marginalized Quadratic Kernel. Results are reported as mean ± std over 10 random seeds.

z Sampling Strategy	Amlodipine MPO	Perindopril MPO	Fexofenadine MPO	Zaleplon MPO
Latent Geometry-Aware Sampling	$\mathbf{0.615 \pm 0.010}$	$\mathbf{0.673 \pm 0.015}$	$\mathbf{0.772 \pm 0.006}$	$\mathbf{0.521 \pm 0.025}$
Gaussian Prior Sampling	0.503 ± 0.013	0.472 ± 0.011	0.682 ± 0.003	0.385 ± 0.012
PCN-MCMC Sampling [21]	0.576 ± 0.027	0.638 ± 0.016	0.725 ± 0.014	0.493 ± 0.009
RAASP Sampling [51]	0.559 ± 0.015	0.624 ± 0.013	0.731 ± 0.019	0.467 ± 0.008

Table 3: Joint ablation of kernel covariance and candidate sampling on four small-budget PMO tasks. Full and Zero indicate whether the encoder covariance is retained or set to zero in the kernel. LGAS uses encoder covariance in both settings. Results are reported as mean ± std over 10 random seeds.

Covariance	Sampler	Amlodipine MPO	Perindopril MPO	Fexofenadine MPO	Zaleplon MPO
Full	LGAS	$\mathbf{0.615 \pm 0.010}$	$\mathbf{0.673 \pm 0.015}$	$\mathbf{0.772 \pm 0.006}$	$\mathbf{0.521 \pm 0.025}$
	RAASP	0.559 ± 0.015	0.624 ± 0.013	0.731 ± 0.019	0.467 ± 0.008
Zero	LGAS	0.571 ± 0.014	0.638 ± 0.007	0.749 ± 0.009	0.493 ± 0.012
	RAASP	0.564 ± 0.021	0.606 ± 0.012	0.746 ± 0.011	0.469 ± 0.017

To examine the interaction between the two components described in Section 4.1 and Section 4.2, we compare kernels with and without encoder covariance under latent geometry-aware sampling (LGAS) and RAASP in Table 3. Setting encoder covariance to zero in the kernel yields the mean-only kernel $k_0(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \boldsymbol{\mu}_\phi(\boldsymbol{x}') + c)^2$ while LGAS continues to use the VAE encoder covariance. Combining the marginalized kernel with LGAS achieves the highest score on all four tasks, while using covariance in the kernel gives mixed results under RAASP. We emphasize that the two uses of covariance in MarKBO serve complementary roles. The marginalized kernel incorporates encoder covariance into the GP prior and thereby influences the posterior over functions. The increase in pointwise prior variance relative to the mean-only kernel is $k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}) - k_0(\boldsymbol{x}, \boldsymbol{x}) = \mathrm{tr}\big(\boldsymbol{\Sigma}_\phi(\boldsymbol{x})^2\big) + 2\boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) \boldsymbol{\mu}_\phi(\boldsymbol{x}) \geq 0$. The latent geometry-aware sampler uses encoder covariance to define a latent proposal distribution and decoded proposals form the candidate set $\mathcal{X}_{\mathrm{gen}}$. Thompson sampling selects each query by maximizing a posterior function sample over this set. The kernel thus influences the distribution of the function being maximized while the sampler defines the candidate domain.

6 Conclusion

We presented MarKBO, a LSBO framework that incorporates VAE encoder uncertainty into both GP modeling and candidate sampling. The marginalized kernel integrates out the approximate posterior to provide a principled kernel function over structured inputs with a closed-form expression. The latent geometry-aware sampler leverages posterior variance as a Riemannian metric to focus exploration on well-supported regions. Experiments on diverse molecular design and robot design tasks demonstrate that MarKBO achieves competitive or superior performance compared to strong baselines.

Limitations Our work depends on the quality of the latent representation learned by a deep generative model. For VAEs, an inadequately designed training strategy can lead to posterior collapse or poor reconstruction quality. This can reduce the reliability of latent space surrogate modeling and local search. We further discuss the limitations of our algorithm with experimental results in Appendix A.

Broader Impact Statement Our work aims to advance LSBO methods in structured domains. By improving the sample efficiency, our approach can reduce reliance on expensive experimental evaluations. However, the generality of our framework poses misuse risk if applied to objectives that conflict with safety considerations. We therefore advocate responsible use of these tools.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for helpful discussions and feedback on an earlier draft of this paper. This work was supported in part by the National Natural Science Foundation of China under Grant 62476096, in part by the Key Core and Pivotal Technology Research Project of the Hengqin Guangdong-Macao In-Depth Cooperation Zone under Grant 2430004045040.

References

[1] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[2] 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.

[3] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[4] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[5] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[6] 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 *International conference on machine learning*, pages 20459–20478. PMLR, 2022.

[7] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[8] Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-vae: a variational autoencoder for directed acyclic graphs. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 1588–1600, 2019.

[9] Henry Moss, David Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. Boss: Bayesian optimization over string spaces. *Advances in neural information processing systems*, 33:15476–15486, 2020.

[10] Jeffrey Tao, Natalie Maus, Haydn Jones, Yimeng Zeng, Jacob R Gardner, and Ryan Marcus. Learned offline query planning via bayesian optimization. *Proceedings of the ACM on Management of Data*, 3(3): 1–29, 2025.

[11] 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.

[12] Onur Boyar. De novo molecular and crystal design with latent space bayesian optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 29247–29248, 2025.

[13] Rika Antonova, Akshara Rai, Tianyu Li, and Danica Kragic. Bayesian optimization in variational latent spaces with dynamic compression. In *Conference on Robot Learning*, pages 456–465. PMLR, 2020.

[14] Jiaheng Hu, Julian Whitman, and Howie Choset. Glso: Grammar-guided latent space optimization for sample-efficient robot design automation. In *Conference on Robot Learning*, pages 1321–1331. PMLR, 2023.

[15] Wangzhen Li, Zhaori Bi, and Xuan Zeng. High-dimensional analog circuit sizing via bayesian optimization in the variational autoencoder enhanced latent space. In *2024 2nd International Symposium of Electronics Design Automation (ISEDA)*, pages 193–197. IEEE, 2024.

[16] Jialin Song, Aidan Swope, Robert Kirby, Rajarshi Roy, Saad Godil, Jonathan Raiman, and Bryan Catanzaro. Circuitvae: Efficient and scalable latent circuit optimization. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024.

[17] Seunghun Lee, Jaewon Chu, Sihyeon Kim, Juyeon Ko, and Hyunwoo J Kim. Advancing bayesian optimization via learning correlated latent space. *Advances in Neural Information Processing Systems*, 36: 48906–48917, 2023.

[18] Jaewon Chu, Jinyoung Park, Seunghun Lee, and Hyunwoo J Kim. Inversion-based latent bayesian optimization. *Advances in Neural Information Processing Systems*, 37:68258–68286, 2024.

[19] Seunghun Lee, Jinyoung Park, Jaewon Chu, Minseo Yoon, and Hyunwoo J Kim. Latent bayesian optimization via autoregressive normalizing flows. In *13th International Conference on Learning Representations, ICLR 2025*, pages 100792–100814. International Conference on Learning Representations, ICLR, 2025.

[20] Natalie Maus, Zhiyuan Jerry Lin, Maximilian Balandat, and Eytan Bakshy. Joint composite latent space bayesian optimization. In *International Conference on Machine Learning*, pages 35163–35180. PMLR, 2024.

[21] Henry Moss, Sebastian W Ober, and Tom Diethe. Return of the latent space cowboys: Re-thinking the use of vaes for bayesian optimisation of structured spaces. In *International Conference on Machine Learning*, pages 44956–44970. PMLR, 2025.

[22] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[23] Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf H Roohani, Jure Leskovec, Connor W. Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=8nvgnORnoWr.

[24] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[25] Matthew D Hoffman, Andrew Gelman, et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[26] José Nogueira, Ruben Martinez-Cantin, Alexandre Bernardino, and Lorenzo Jamone. Unscented bayesian optimization for safe robot grasping. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1967–1972. IEEE, 2016.

[27] Samuel Daulton, Sait Cakmak, Maximilian Balandat, Michael A Osborne, Enlu Zhou, and Eytan Bakshy. Robust multi-objective bayesian optimization under input noise. In *International Conference on Machine Learning*, pages 4831–4866. PMLR, 2022.

[28] Sait Cakmak, Raul Astudillo Marban, Peter Frazier, and Enlu Zhou. Bayesian optimization of risk measures. *Advances in Neural Information Processing Systems*, 33:20130–20141, 2020.

[29] Saul Toscano-Palmerin and Peter I Frazier. Bayesian optimization with expensive integrands. *SIAM Journal on Optimization*, 32(2):417–444, 2022.

[30] Hisham Husain, Vu Nguyen, and Anton van den Hengel. Distributionally robust bayesian optimization with $varphi$-divergences. *Advances in Neural Information Processing Systems*, 36:20133–20145, 2023.

[31] Johannes Kirschner, Ilija Bogunovic, Stefanie Jegelka, and Andreas Krause. Distributionally robust bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2174–2184. PMLR, 2020.

[32] Nikitas Rontsis, Michael A Osborne, and Paul J Goulart. Distributionally ambiguous optimization for batch bayesian optimization. *Journal of Machine Learning Research*, 21(149):1–26, 2020.

[33] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions:: A review and beyond. *Foundations and Trends® in Machine Learning*, 10 (1-2):1–141, 2017.

[34] Rafael Oliveira, Lionel Ott, and Fabio Ramos. Bayesian optimisation under uncertain inputs. In *The 22nd international conference on artificial intelligence and statistics*, pages 1177–1184. PMLR, 2019.

[35] Antoine Chatalic, Nicolas Schreuder, Lorenzo Rosasco, and Alessandro Rudi. Nyström kernel mean embeddings. In *International Conference on Machine Learning*, pages 3006–3024. PMLR, 2022.

[36] Lin Yang, Junlong Lyu, Wenlong Lyu, and Zhitang Chen. Efficient robust bayesian optimization for arbitrary uncertain inputs. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 43156–43176, 2023.

[37] Ryan B Christianson and Robert B Gramacy. Robust expected improvement for bayesian optimization. *IISE Transactions*, 56(12):1294–1306, 2024.

[38] Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher. Adversarially robust optimization with gaussian processes. *Advances in neural information processing systems*, 31, 2018.

[39] Taeyoung Yun, Kiyoung Om, Jaewoo Lee, Sujin Yun, and Jinkyoo Park. Posterior inference with diffusion models for high-dimensional black-box optimization. In *International Conference on Machine Learning*, pages 73897–73917. PMLR, 2025.

[40] Rafael Oliveira, Daniel M. Steinberg, and Edwin V. Bonilla. Generative bayesian optimization: Generative models as acquisition functions. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=GBWkRRJrdu.

[41] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33:11259–11272, 2020.

[42] Taicai Chen, Yue Duan, Dong Li, Lei Qi, Yinghuan Shi, and Yang Gao. Pg-lbo: Enhancing high-dimensional bayesian optimization with pseudo-label and gaussian process guidance. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 11381–11389, 2024.

[43] Onur Boyar and Ichiro Takeuchi. Latent space bayesian optimization with latent data augmentation for enhanced exploration. *Neural Computation*, 36(11):2446–2478, 2024.

[44] Eero Siivola, Andrei Paleyes, Javier González, and Aki Vehtari. Good practices for bayesian optimization of high dimensional structured spaces. *Applied AI Letters*, 2(2):e24, 2021.

[45] Natalie Maus, Haydn Jones, Juston Moore, Matt J Kusner, John Bradshaw, and Jacob Gardner. Local latent space bayesian optimization over structured inputs. *Advances in neural information processing systems*, 35:34505–34518, 2022.

[46] Yuxuan Yin, Yu Wang, and Peng Li. High-dimensional bayesian optimization via semi-supervised learning with optimized unlabeled data sampling. In *Proceedings of the 41st International Conference on Machine Learning*, pages 57085–57100, 2024.

[47] Siddharth Ramchandran, Manuel Haussmann, and Harri Lähdesmäki. High-dimensional bayesian optimisation with gaussian process prior variational autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025.

[48] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[49] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[50] Omer Ronen, Ahmed Imtiaz Humayun, Richard Baraniuk, Randall Balestriero, and Bin Yu. Mitigating over-exploration in latent space optimization using les. In *International Conference on Machine Learning*, pages 51996–52024. PMLR, 2025.

[51] Leonard Papenmeier, Matthias Poloczek, and Luigi Nardi. Understanding high-dimensional bayesian optimization. In *International Conference on Machine Learning*, pages 47902–47923. PMLR, 2025.

[52] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.

[53] 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.

[54] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. *Advances in neural information processing systems*, 36:20577–20612, 2023.

[55] Bahador Rashidi, Kerrick Johnstonbaugh, and Chao Gao. Cylindrical thompson sampling for high-dimensional bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 3502–3510. PMLR, 2024.

[56] Koji Tsuda, Taishin Kin, Kiyoshi Asai, et al. Marginalized kernels for biological sequences. *BIOINFORMATICS-OXFORD-*, 18:S268–S275, 2002.

[57] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 321–328, 2003.

[58] Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Extensions of marginalized graph kernels. In *Proceedings of the twenty-first international conference on Machine learning*, page 70, 2004.

[59] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010.

[60] Krikamol Muandet, Kenji Fukumizu, Francesco Dinuzzo, and Bernhard Schölkopf. Learning from distributions via support measure machines. *Advances in neural information processing systems*, 25, 2012.

[61] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions. In *International Conference on Machine Learning*, pages 20793–20817. PMLR, 2024.

[62] Zhitong Xu, Haitao Wang, Jeff M. Phillips, and Shandian Zhe. Standard gaussian process is all you need for high-dimensional bayesian optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kX8h23UG6v.

[63] Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in advances in approximate Bayesian inference, NIPS*, volume 1, 2016.

[64] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International conference on artificial intelligence and statistics*, pages 1214–1223. PMLR, 2018.

[65] Pascal Notin, José Miguel Hernández-Lobato, and Yarin Gal. Improving black-box optimization in vae latent space using decoder uncertainty. *Advances in Neural Information Processing Systems*, 34:802–814, 2021.

[66] Hang Shao, Abhishek Kumar, and P Thomas Fletcher. The riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 315–323, 2018.

[67] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJzRZ-WCZ.

[68] Georgios Arvanitidis, Soren Hauberg, and Bernhard Schölkopf. Geometrically enriched latent spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 631–639. PMLR, 2021.

[69] Alessandra Tosi, Sören Hauberg, Alfredo Vellido, and Neil D Lawrence. Metrics for probabilistic geometries. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 800–808, 2014.

[70] Clément Chadebec and Stéphanie Allassonnière. A geometric perspective on variational autoencoders. *Advances in neural information processing systems*, 35:19618–19630, 2022.

[71] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *Journal of machine learning research*, 20(28):1–6, 2019.

[72] Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35:21342–21357, 2022.

[73] Miguel González-Duque, Richard Michael, Simon Bartels, Yevgen Zainchkovskyy, Søren Hauberg, and Wouter Boomsma. A survey and benchmark of high-dimensional bayesian optimization of discrete sequences. *Advances in Neural Information Processing Systems*, 37:140478–140508, 2024.

[74] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

[75] Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

[76] Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019.

[77] Colin Doumont, Donney Fan, Natalie Maus, Jacob R. Gardner, Henry Moss, and Geoff Pleiss. We still don't understand high-dimensional bayesian optimization. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026. URL https://openreview.net/forum?id=EvjTyCbBZX.

[78] Tingwu Wang, Yuhao Zhou, Sanja Fidler, and Jimmy Ba. Neural graph evolution: Towards efficient automatic robot design. In *International Conference on Learning Representations*, 2019.

[79] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 240–250, 2019.

[80] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 2002.

[81] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International conference on algorithmic learning theory*, pages 13–31. Springer, 2007.

[82] Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1550. PMLR, 2018.

[83] Xavier Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006.

[84] Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 1. 2015.

[85] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

[86] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[87] James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from gaussian process posteriors. In *International conference on machine learning*, pages 10292–10302. PMLR, 2020.

Appendix

A Detailed Discussion of Algorithmic Limitations **16**

B Derivation of the Marginalized Kernel **18**

B.1 Positive Semi-Definiteness of the Marginalized Kernel 18

B.2 Marginalized Linear Kernel 19

B.3 Marginalized Quadratic Polynomial Kernel 19

B.4 Marginalized RBF Kernel 20

B.5 Connection to Kernel Mean Embeddings 22

C Preliminaries on Riemannian Geometry **22**

D Additional Experiments **23**

D.1 Small Budget Setting on PMO Benchmark 23

D.2 Large Budget Setting on PMO Benchmark 24

E Experimental Details **26**

E.1 Deep Generative Models 26

E.2 Surrogate Model 28

E.3 Acquisition Strategy 29

E.4 Baselines 29

E.5 Compute Resources 30

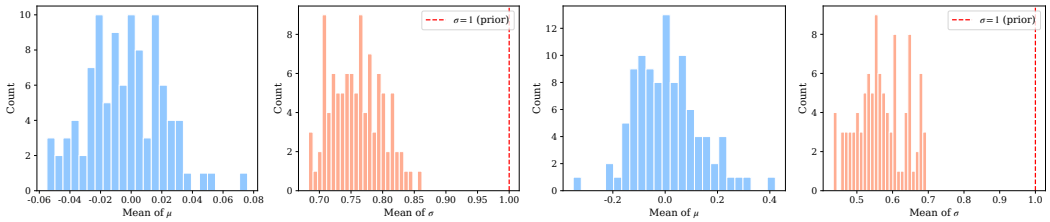
E.6 Licenses for Existing Assets 30

F Pseudocode and Hyperparameters of MarKBO **31**

A Detailed Discussion of Algorithmic Limitations

In this section, we discuss three main limitations of our approach. First, we analyze the dependence on the quality of the learned latent representation and show that posterior collapse is unlikely to be a practical concern for well-pretrained VAEs. Second, we acknowledge that our method does not uniformly outperform all baselines on every task. Third, we examine the computational overhead introduced by the NUTS sampling and argue that it remains acceptable in expensive black-box optimization settings.

As noted in Sec. 6, our framework relies on the quality of the learned latent representation and posterior collapse could reduce the reliability of latent space surrogate modeling. However, this issue is unlikely to arise in practice for well-pretrained VAEs due to established techniques such as KL annealing [79]. To verify this, we examine the approximate posterior statistics of the 100 initial points used in the small-budget PMO experiments and robot morphology tasks of Section 5.2. As shown in Fig. 4, for both SELFIES-VAE ($256d$) and GLSO-VAE ($28d$), the per-sample mean of $\boldsymbol{\mu}$ is concentrated near zero, consistent with the standard normal prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The per-sample mean of $\boldsymbol{\sigma}$ is neither collapsed to zero nor uniformly equal to one. Instead it exhibits a diverse spread across data points (roughly 0.65–0.85 for SELFIES-VAE and 0.4–0.7 for GLSO-VAE), indicating that the encoder has learned meaningful and data-dependent uncertainty. It also motivates our proposed method in Sec. 4.1 and Sec. 4.2 that leverages the VAE distributional parameters $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ throughout the optimization loop.



(a) SELFIES-VAE (256d) [45] (b) GLSO-VAE (28d) [14]

Figure 4: Empirical posterior statistics of the first 100 initial points. The left histograms show the per-sample mean of the encoder mean $\boldsymbol{\mu}$ and the right histograms show the per-sample mean of the encoder standard deviation $\boldsymbol{\sigma}$. The red dashed line marks the prior σ=1. For both VAEs, $\boldsymbol{\sigma}$ is neither collapsed to zero nor uniformly equal to one.

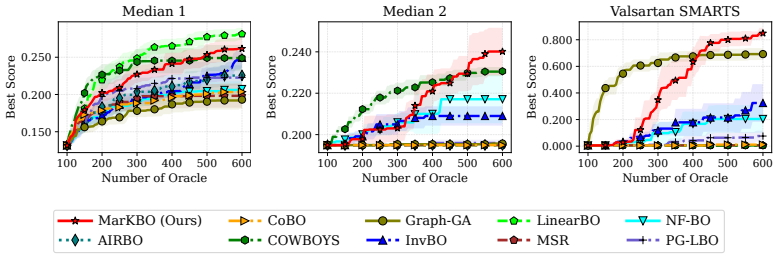


Figure 5: Three small-budget PMO tasks where MarKBO shows limitations or slower convergence.

Our algorithm is an improvement upon VAE-based LSBO. The main contribution of this paper lies in how to effectively leverage the mean and covariance of the posterior distribution during the surrogate modeling and candidate sampling stages. Our method does not involve any modifications to the architecture of the generative model itself nor any supervised fine-tuning. As shown in Figure 5, we highlight three tasks where MarKBO exhibits suboptimal or slower convergence behavior. On Median 1, LinearBO maintains a higher best score than MarKBO throughout the entire optimization budget. On Median 2 and Valsartan SMARTS, MarKBO exhibits slower progress during the early optimization stage. As the No Free Lunch theorem [80] states, no single optimization algorithm can achieve optimal performance across all problems. Our experimental results also demonstrate no algorithm consistently dominates on all test problems. We therefore acknowledge that our method does not achieve state-of-the-art results on every problem.

Table 4: Runtime analysis of MarKBO on the robot morphology benchmark (500 evaluations, batch size 5, 100 iterations). **Left:** Wall-clock time of MarKBO across the four terrain tasks. **Right:** Per-component time breakdown on the least expensive task (Frozen Lake Terrain).

Task	**Iters**	**Total (s)**	**Avg/iter (s)**
Flat	100	20 468	204.7
Frozen Lake	100	17 475	174.8
Ridged	100	24 772	247.7
Wall	100	27 058	270.6
Average	100	**22 443**	**224.4**

Component	**Time (s)**	**Share (%)**
Black-box simulation	14 190	81.20
Surrogate fitting	245	1.40
Candidate generation	2 848	16.30
Acquisition evaluation	192	1.10
Total	**17 475**	**100.00**

Another limitation of our method lies in its computational overhead. The primary cost stems from the NUTS sampling procedure. As shown in Table 5, MarKBO is faster than COWBOYS [21], which employs the gradient-free PCN-MCMC sampler for candidate generation. This speedup can be attributed to the gradient-based nature of NUTS, which enables more efficient traversal than a random walk. Although NUTS is more efficient than traditional MCMC sampling [25], it still introduces a non-negligible time cost. However, this overhead is acceptable in practice for two reasons. First, our algorithm achieves competitive or better optimization performance than the baseline methods. This means that under the same evaluation budget, it finds superior solutions. The additional computational cost is justified by the improved sample efficiency. Second, in expensive black-box optimization

scenarios, a single evaluation of the objective function is often time-consuming. Tasks such as engineering simulations typically impose a limited evaluation budget. In such settings, the runtime of the optimization algorithm itself is negligible compared to the cost of objective function evaluations. Therefore, this computational overhead does not constitute a practical bottleneck.

Table 5: Average wall-clock time on the 4 small-budget PMO tasks from Sec. 5.1 (500 evaluations, 100 iterations, batch size 5). Methods are sorted by total time in ascending order.

Method	Total (s)	Avg/iter (s)	Std/iter (s)
MSR	36.4	0.36	0.05
Graph-GA	56.7	0.57	0.11
CoBO	59.7	0.60	0.38
PG-LBO	70.8	0.71	0.15
InvBO	291.6	2.92	1.85
NF-BO	303.4	3.03	0.04
LinearBO	370.2	3.70	0.58
AIRBO	1021.8	10.22	0.52
MarKBO (Ours)	1754.6	17.55	2.18
COWBOYS	2733.3	27.33	2.14

In the PMO benchmark the oracle functions are relatively cheap to evaluate. In the robot morphology benchmark, each evaluation requires a full physics simulation coupled with nested control optimization, making individual evaluations computationally expensive. Table 4 provides a per-component runtime breakdown on the Frozen Lake Terrain task. As shown, black-box simulation dominates the total runtime at approximately 81%. Surrogate fitting and candidate generation together account for approximately 18%. These results confirm that the overhead introduced by our algorithm is relatively small compared to the simulation cost.

B Derivation of the Marginalized Kernel

In this section, we provide the complete derivations of three marginalized kernels that admit closed-form expressions under a Gaussian variational posterior proposed in Sec. 4.1. We begin by establishing that the marginalized kernel inherits positive semi-definiteness from the base latent space kernel $k_{\boldsymbol{z}}$ (Section B.1). We then derive the analytic forms for the linear kernel (Section B.2), the quadratic polynomial kernel (Section B.3), and the RBF kernel (Section B.4). Finally, we discuss the connection between the marginalized kernel and the kernel mean embedding (Section B.5).

B.1 Positive Semi-Definiteness of the Marginalized Kernel

We first prove that the marginalized kernel k_{marg} inherits positive semi-definiteness (p.s.d.) from the base latent space kernel $k_{\boldsymbol{z}}$, regardless of the choice of variational family.

Assumption 1. *The base kernel $k_{\boldsymbol{z}} : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is a symmetric positive semi-definite kernel on $\mathcal{Z}$.*

Assumption 2. *For every $\boldsymbol{x} \in \mathcal{X}$, the variational posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ is a probability distribution on $\mathcal{Z}$ satisfying the integrability condition*

$$\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\sqrt{k_{\boldsymbol{z}}(\boldsymbol{z},\boldsymbol{z})}\big] < \infty. \tag{12}$$

Proposition 1. *Under Assumptions 1 and 2, the marginalized kernel*

$$k_{\text{marg}}(\boldsymbol{x},\boldsymbol{x}') \;:=\; \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x}),\, \boldsymbol{z}'\sim q_\phi(\boldsymbol{z}'|\boldsymbol{x}')}\big[k_{\boldsymbol{z}}(\boldsymbol{z},\boldsymbol{z}')\big] \tag{13}$$

is a symmetric positive semi-definite kernel on $\mathcal{X}$.

Proof. **Symmetry.** The symmetry of k follows directly from the symmetry of $k_{\boldsymbol{z}}$ and the commutativity of the product measure $q_\phi(\boldsymbol{z}|\boldsymbol{x})\, q_\phi(\boldsymbol{z}'|\boldsymbol{x}')$.

Positive semi-definiteness. By Assumption 1, $k_{\boldsymbol{z}}$ is a p.s.d. kernel, so by the Moore–Aronszajn theorem (Theorem 6.1 in [4]) there exists a reproducing kernel Hilbert space $\mathcal{H}_{k_{\boldsymbol{z}}}$ with canonical feature map $\varphi : \mathcal{Z} \to \mathcal{H}_{k_{\boldsymbol{z}}}$, $\varphi(\boldsymbol{z}) := k_{\boldsymbol{z}}(\cdot, \boldsymbol{z})$, satisfying

$$k_{\boldsymbol{z}}(\boldsymbol{z},\boldsymbol{z}') = \langle \varphi(\boldsymbol{z}),\, \varphi(\boldsymbol{z}')\rangle_{\mathcal{H}_{k_{\boldsymbol{z}}}} \quad \forall\, \boldsymbol{z},\boldsymbol{z}' \in \mathcal{Z}. \tag{14}$$

Assumption 2 guarantees that $\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\|\varphi(\boldsymbol{z})\|_{\mathcal{H}_{k_{\boldsymbol{z}}}}] = \mathbb{E}[\sqrt{k_{\boldsymbol{z}}(\boldsymbol{z},\boldsymbol{z})}] < \infty$, so the *Bochner integral*

$$\bar{\varphi}(\boldsymbol{x}) \;:=\; \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\varphi(\boldsymbol{z})\big] \;=\; \int_{\mathcal{Z}} \varphi(\boldsymbol{z})\, q_\phi(\boldsymbol{z}\mid\boldsymbol{x})\,\mathrm{d}\boldsymbol{z} \;\in\; \mathcal{H}_{k_{\boldsymbol{z}}} \tag{15}$$

is well-defined as an element of $\mathcal{H}_{k_{\boldsymbol{z}}}$ for every $\boldsymbol{x}\in\mathcal{X}$. Because the inner product in a Hilbert space is continuous and bilinear, it commutes with Bochner integrals:

$$\begin{aligned}
k_{\mathrm{marg}}(\boldsymbol{x},\boldsymbol{x}') &= \iint k_{\boldsymbol{z}}(\boldsymbol{z},\boldsymbol{z}')\, q_\phi(\boldsymbol{z}|\boldsymbol{x})\, q_\phi(\boldsymbol{z}'|\boldsymbol{x}')\,\mathrm{d}\boldsymbol{z}\,\mathrm{d}\boldsymbol{z}' \\
&= \iint \langle \varphi(\boldsymbol{z}),\, \varphi(\boldsymbol{z}')\rangle_{\mathcal{H}_{k_{\boldsymbol{z}}}}\, q_\phi(\boldsymbol{z}|\boldsymbol{x})\, q_\phi(\boldsymbol{z}'|\boldsymbol{x}')\,\mathrm{d}\boldsymbol{z}\,\mathrm{d}\boldsymbol{z}' \\
&= \left\langle \int \varphi(\boldsymbol{z})\, q_\phi(\boldsymbol{z}|\boldsymbol{x})\,\mathrm{d}\boldsymbol{z},\; \int \varphi(\boldsymbol{z}')\, q_\phi(\boldsymbol{z}'|\boldsymbol{x}')\,\mathrm{d}\boldsymbol{z}' \right\rangle_{\mathcal{H}_{k_{\boldsymbol{z}}}} \\
&= \big\langle \bar{\varphi}(\boldsymbol{x}),\; \bar{\varphi}(\boldsymbol{x}')\big\rangle_{\mathcal{H}_{k_{\boldsymbol{z}}}}.
\end{aligned} \tag{16}$$

Hence $k_{\mathrm{marg}}(\boldsymbol{x},\boldsymbol{x}')$ equals the inner product of images under the feature map $\bar{\varphi}:\mathcal{X}\to\mathcal{H}_{k_{\boldsymbol{z}}}$. For any $n\in\mathbb{N}$, any $\boldsymbol{x}_1,\dots,\boldsymbol{x}_n\in\mathcal{X}$, and any $c_1,\dots,c_n\in\mathbb{R}$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i\, c_j\, k(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_{i,j} c_i\, c_j\, \big\langle \bar{\varphi}(\boldsymbol{x}_i),\, \bar{\varphi}(\boldsymbol{x}_j)\big\rangle_{\mathcal{H}_{k_{\boldsymbol{z}}}} = \left\| \sum_{i=1}^{n} c_i\, \bar{\varphi}(\boldsymbol{x}_i) \right\|^2_{\mathcal{H}_{k_{\boldsymbol{z}}}} \;\geq\; 0, \tag{17}$$

which is the defining property of a positive semi-definite kernel. □

Remark. Proposition 1 guarantees that the closed-form kernels derived in the subsequent sections are valid (p.s.d.) kernels. The proof relies only on the RKHS structure of $k_{\boldsymbol{z}}$ and the integrability of the feature map under q_ϕ. It places no restriction on the parametric form of the variational posterior.

B.2 Marginalized Linear Kernel

Let the base kernel in the latent space be the linear kernel:

$$k_{\boldsymbol{z}}(\boldsymbol{z},\boldsymbol{z}') = \boldsymbol{z}^\top \boldsymbol{z}'. \tag{18}$$

Under the Gaussian variational posterior $q_\phi(\boldsymbol{z}\mid\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}),\, \boldsymbol{\Sigma}_\phi(\boldsymbol{x}))$, the marginalized kernel (Eq. 4) becomes

$$k_{\mathrm{marg}}(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x}),\, \boldsymbol{z}'\sim q_\phi(\boldsymbol{z}'|\boldsymbol{x}')}\left[\boldsymbol{z}^\top \boldsymbol{z}'\right]. \tag{19}$$

Since $\boldsymbol{z}$ and $\boldsymbol{z}'$ are drawn independently from their respective posteriors, the expectation factorizes:

$$k_{\mathrm{marg}}(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\boldsymbol{z}]^\top\, \mathbb{E}_{\boldsymbol{z}'\sim q_\phi(\boldsymbol{z}'|\boldsymbol{x}')}[\boldsymbol{z}']. \tag{20}$$

Substituting the mean of each Gaussian variational posterior, we obtain

$$k_{\mathrm{marg}}(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \boldsymbol{\mu}_\phi(\boldsymbol{x}'). \tag{21}$$

This shows that the marginalized linear kernel reduces to the inner product between the variational mean embeddings, and can be computed in $\mathcal{O}(d)$ time where d is the latent dimensionality. Notably, the covariance $\Sigma_\phi(\boldsymbol{x})$ does not appear in the final expression, since the linear kernel depends only on first moments.

B.3 Marginalized Quadratic Polynomial Kernel

Let the base kernel in the latent space be the quadratic polynomial kernel:

$$k_{\boldsymbol{z}}(\boldsymbol{z},\boldsymbol{z}') = (\boldsymbol{z}^\top \boldsymbol{z}' + c)^2, \tag{22}$$

where $c\geq 0$ is a constant. Under the Gaussian variational posterior $q_\phi(\boldsymbol{z}\mid\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\Sigma}_\phi(\boldsymbol{x}))$, the marginalized kernel (Eq. 4) becomes

$$k_{\mathrm{marg}}(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x}),\, \boldsymbol{z}'\sim q_\phi(\boldsymbol{z}'|\boldsymbol{x}')}\left[(\boldsymbol{z}^\top \boldsymbol{z}' + c)^2\right]. \tag{23}$$

Expanding the square yields three terms:

$$(\boldsymbol{z}^\top \boldsymbol{z}' + c)^2 = (\boldsymbol{z}^\top \boldsymbol{z}')^2 + 2c\,(\boldsymbol{z}^\top \boldsymbol{z}') + c^2. \tag{24}$$

We evaluate the expectation of each term in turn.

Term 1: $\mathbb{E}[(\boldsymbol{z}^\top \boldsymbol{z}')^2]$**.** Since $\boldsymbol{z}$ and $\boldsymbol{z}'$ are independent, we condition on $\boldsymbol{z}$ and write

$$\mathbb{E}_{\boldsymbol{z},\boldsymbol{z}'}\left[(\boldsymbol{z}^\top \boldsymbol{z}')^2\right] = \mathbb{E}_{\boldsymbol{z}}\left[\mathbb{E}_{\boldsymbol{z}'}\left[(\boldsymbol{z}^\top \boldsymbol{z}')^2\right]\right] = \mathbb{E}_{\boldsymbol{z}}\left[\mathbb{E}_{\boldsymbol{z}'}\left[\boldsymbol{z}^\top \boldsymbol{z}'\, \boldsymbol{z}'^\top \boldsymbol{z}\right]\right] = \mathbb{E}_{\boldsymbol{z}}\left[\boldsymbol{z}^\top\, \mathbb{E}_{\boldsymbol{z}'}\left[\boldsymbol{z}'\boldsymbol{z}'^\top\right] \boldsymbol{z}\right]. \quad (25)$$

Define the *second moment matrix* $M_\phi(\boldsymbol{x}') := \mathbb{E}_{\boldsymbol{z}'}[\boldsymbol{z}'\boldsymbol{z}'^\top] = \boldsymbol{\Sigma}_\phi(\boldsymbol{x}') + \boldsymbol{\mu}_\phi(\boldsymbol{x}')\boldsymbol{\mu}_\phi(\boldsymbol{x}')^\top$. Then the inner expectation gives a quadratic form in $\boldsymbol{z}$:

$$\mathbb{E}_{\boldsymbol{z}}\left[\boldsymbol{z}^\top M_\phi(\boldsymbol{x}')\, \boldsymbol{z}\right]. \quad (26)$$

Applying the standard identity $\mathbb{E}[\boldsymbol{z}^\top \boldsymbol{A}\, \boldsymbol{z}] = \boldsymbol{\mu}^\top \boldsymbol{A}\, \boldsymbol{\mu} + \mathrm{tr}(\boldsymbol{A}\, \boldsymbol{\Sigma})$ for $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\Sigma}_\phi(\boldsymbol{x}))$, we obtain

$$\begin{aligned}
\mathbb{E}[(\boldsymbol{z}^\top \boldsymbol{z}')^2] &= \boldsymbol{\mu}_\phi(\boldsymbol{x})^\top M_\phi(\boldsymbol{x}')\, \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \mathrm{tr}(M_\phi(\boldsymbol{x}')\, \boldsymbol{\Sigma}_\phi(\boldsymbol{x})) \\
&= \boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \left[\boldsymbol{\Sigma}_\phi(\boldsymbol{x}') + \boldsymbol{\mu}_\phi(\boldsymbol{x}')\boldsymbol{\mu}_\phi(\boldsymbol{x}')^\top\right] \boldsymbol{\mu}_\phi(\boldsymbol{x}) \\
&\quad + \mathrm{tr}\big(\left[\boldsymbol{\Sigma}_\phi(\boldsymbol{x}') + \boldsymbol{\mu}_\phi(\boldsymbol{x}')\boldsymbol{\mu}_\phi(\boldsymbol{x}')^\top\right] \boldsymbol{\Sigma}_\phi(\boldsymbol{x})\big) \\
&= \big(\boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \boldsymbol{\mu}_\phi(\boldsymbol{x}')\big)^2 + \boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \boldsymbol{\Sigma}_\phi(\boldsymbol{x}')\, \boldsymbol{\mu}_\phi(\boldsymbol{x}) \\
&\quad + \boldsymbol{\mu}_\phi(\boldsymbol{x}')^\top \boldsymbol{\Sigma}_\phi(\boldsymbol{x})\, \boldsymbol{\mu}_\phi(\boldsymbol{x}') + \mathrm{tr}\big(\boldsymbol{\Sigma}_\phi(\boldsymbol{x})\, \boldsymbol{\Sigma}_\phi(\boldsymbol{x}')\big). \quad (27)
\end{aligned}$$

Term 2: $2c\,\mathbb{E}[\boldsymbol{z}^\top \boldsymbol{z}']$**.** By independence and the result from Section B.2,

$$2c\,\mathbb{E}[\boldsymbol{z}^\top \boldsymbol{z}'] = 2c\,\boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \boldsymbol{\mu}_\phi(\boldsymbol{x}'). \quad (28)$$

Term 3. The constant term contributes c^2.

Full expression. Combining all three terms, the marginalized quadratic polynomial kernel can be written as

$$k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{tr}\big(M_\phi(\boldsymbol{x})\, M_\phi(\boldsymbol{x}')\big) + 2c\,\boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \boldsymbol{\mu}_\phi(\boldsymbol{x}') + c^2, \quad (29)$$

where $M_\phi(\boldsymbol{x}) = \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) + \boldsymbol{\mu}_\phi(\boldsymbol{x})\boldsymbol{\mu}_\phi(\boldsymbol{x})^\top$.

Remark. In contrast to the marginalized linear kernel (Section B.2), the covariance $\boldsymbol{\Sigma}_\phi(\boldsymbol{x})$ now appears explicitly in the result. This means the quadratic polynomial kernel in the marginalized space is sensitive not only to the posterior means but also to the posterior uncertainties. Two inputs $\boldsymbol{x}, \boldsymbol{x}'$ whose encoders share similar covariance structure will receive additional kernel similarity through the $\mathrm{tr}(\boldsymbol{\Sigma}_\phi(\boldsymbol{x})\boldsymbol{\Sigma}_\phi(\boldsymbol{x}'))$ and cross terms. When $\boldsymbol{\Sigma}_\phi(\boldsymbol{x}) = \boldsymbol{0}$ for all $\boldsymbol{x}$ (i.e., a deterministic encoder), Eq. (29) reduces to the standard polynomial kernel evaluated at the means: $k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{\mu}_\phi(\boldsymbol{x})^\top \boldsymbol{\mu}_\phi(\boldsymbol{x}') + c)^2$.

B.4 Marginalized RBF Kernel

Let the base kernel in the latent space be the Gaussian RBF kernel with lengthscale $\ell > 0$:

$$k_z(\boldsymbol{z}, \boldsymbol{z}') = \exp\left(-\frac{\|\boldsymbol{z} - \boldsymbol{z}'\|^2}{2\ell^2}\right). \quad (30)$$

Under the Gaussian variational posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}),\ \boldsymbol{\Sigma}_\phi(\boldsymbol{x}))$, the marginalized kernel (Eq. 4) becomes

$$k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}') \;=\; \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x}),\, \boldsymbol{z}'\sim q_\phi(\boldsymbol{z}'|\boldsymbol{x}')}\left[\exp\left(-\frac{\|\boldsymbol{z} - \boldsymbol{z}'\|^2}{2\ell^2}\right)\right]. \quad (31)$$

Since $\boldsymbol{z}$ and $\boldsymbol{z}'$ are drawn independently, their difference $\boldsymbol{\delta} := \boldsymbol{z} - \boldsymbol{z}'$ is also Gaussian:

$$\boldsymbol{\delta} \;\sim\; \mathcal{N}\big(\boldsymbol{\mu}_\delta,\ \boldsymbol{\Sigma}_\delta\big), \qquad \boldsymbol{\mu}_\delta := \boldsymbol{\mu}_\phi(\boldsymbol{x}) - \boldsymbol{\mu}_\phi(\boldsymbol{x}'), \qquad \boldsymbol{\Sigma}_\delta := \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}'). \quad (32)$$

Equation (31) therefore reduces to

$$k_{\mathrm{marg}}(\boldsymbol{x}, \boldsymbol{x}') \;=\; \mathbb{E}_{\boldsymbol{\delta}\sim\mathcal{N}(\boldsymbol{\mu}_\delta,\, \boldsymbol{\Sigma}_\delta)}\left[\exp\left(-\frac{\boldsymbol{\delta}^\top \boldsymbol{\delta}}{2\ell^2}\right)\right]. \quad (33)$$

Writing out the expectation explicitly,

$$k_{\text{marg}}(\boldsymbol{x}, \boldsymbol{x}') = \int \exp\!\left(-\frac{\boldsymbol{\delta}^\top \boldsymbol{\delta}}{2\ell^2}\right) \cdot \frac{1}{(2\pi)^{d/2}\,|\boldsymbol{\Sigma}_\delta|^{1/2}} \exp\!\big(-\tfrac{1}{2}(\boldsymbol{\delta}-\boldsymbol{\mu}_\delta)^\top \boldsymbol{\Sigma}_\delta^{-1}(\boldsymbol{\delta}-\boldsymbol{\mu}_\delta)\big)\, \mathrm{d}\boldsymbol{\delta}. \quad (34)$$

We combine the two quadratic forms in the exponent. Define the precision-like matrix

$$\boldsymbol{P} \;:=\; \ell^{-2}\mathbf{I} + \boldsymbol{\Sigma}_\delta^{-1}. \quad (35)$$

Then the combined exponent can be written as

$$-\frac{\boldsymbol{\delta}^\top \boldsymbol{\delta}}{2\ell^2} - \frac{1}{2}(\boldsymbol{\delta}-\boldsymbol{\mu}_\delta)^\top \boldsymbol{\Sigma}_\delta^{-1}(\boldsymbol{\delta}-\boldsymbol{\mu}_\delta) \;=\; -\frac{1}{2}\,\boldsymbol{\delta}^\top \boldsymbol{P}\,\boldsymbol{\delta} + \boldsymbol{\delta}^\top \boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{\mu}_\delta - \frac{1}{2}\,\boldsymbol{\mu}_\delta^\top \boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{\mu}_\delta. \quad (36)$$

Completing the square in $\boldsymbol{\delta}$,

$$-\frac{1}{2}\,\boldsymbol{\delta}^\top \boldsymbol{P}\,\boldsymbol{\delta} + \boldsymbol{\delta}^\top \boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{\mu}_\delta = -\frac{1}{2}\big(\boldsymbol{\delta} - \boldsymbol{P}^{-1}\boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{\mu}_\delta\big)^\top \boldsymbol{P}\,\big(\boldsymbol{\delta} - \boldsymbol{P}^{-1}\boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{\mu}_\delta\big) + \frac{1}{2}\,\boldsymbol{\mu}_\delta^\top \boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{P}^{-1}\boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{\mu}_\delta. \quad (37)$$

The Gaussian integral over $\boldsymbol{\delta}$ evaluates to $(2\pi)^{d/2}\,|\boldsymbol{P}^{-1}|^{1/2}$, and collecting the remaining scalar terms we obtain

$$k_{\text{marg}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{|\boldsymbol{P}^{-1}|^{1/2}}{|\boldsymbol{\Sigma}_\delta|^{1/2}} \exp\!\left(-\frac{1}{2}\,\boldsymbol{\mu}_\delta^\top \big(\boldsymbol{\Sigma}_\delta^{-1} - \boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{P}^{-1}\boldsymbol{\Sigma}_\delta^{-1}\big)\boldsymbol{\mu}_\delta\right). \quad (38)$$

Since $\boldsymbol{P} = \boldsymbol{\Sigma}_\delta^{-1}(\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I})$, we have $|\boldsymbol{P}| = |\boldsymbol{\Sigma}_\delta|^{-1}\,|\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I}|$, and thus

$$\frac{|\boldsymbol{P}^{-1}|^{1/2}}{|\boldsymbol{\Sigma}_\delta|^{1/2}} \;=\; \big|\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I}\big|^{-1/2}. \quad (39)$$

From $\boldsymbol{P}^{-1} = (\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I})^{-1}\boldsymbol{\Sigma}_\delta$ it follows that $\boldsymbol{P}^{-1}\boldsymbol{\Sigma}_\delta^{-1} = (\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I})^{-1}$. Therefore,

$$\begin{aligned}
\boldsymbol{\Sigma}_\delta^{-1} - \boldsymbol{\Sigma}_\delta^{-1}\boldsymbol{P}^{-1}\boldsymbol{\Sigma}_\delta^{-1} &= \boldsymbol{\Sigma}_\delta^{-1}\big[\mathbf{I} - (\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I})^{-1}\big] \\
&= \boldsymbol{\Sigma}_\delta^{-1}\,(\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I})^{-1}\,\ell^{-2}\boldsymbol{\Sigma}_\delta \\
&= \ell^{-2}\,(\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I})^{-1} \;=\; (\boldsymbol{\Sigma}_\delta + \ell^2\mathbf{I})^{-1}, \qquad (40)
\end{aligned}$$

where the penultimate equality uses the fact that $\boldsymbol{\Sigma}_\delta^{-1}$ and $(\ell^{-2}\boldsymbol{\Sigma}_\delta + \mathbf{I})^{-1}$ commute (both are functions of the symmetric matrix $\boldsymbol{\Sigma}_\delta$).

Substituting back into Eq. (38) and restoring $\boldsymbol{\mu}_\delta = \boldsymbol{\mu}_\phi(\boldsymbol{x}) - \boldsymbol{\mu}_\phi(\boldsymbol{x}')$ and $\boldsymbol{\Sigma}_\delta = \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}')$, we obtain

$$k(\boldsymbol{x}, \boldsymbol{x}') = \big|\mathbf{I} + \ell^{-2}\big(\boldsymbol{\Sigma}_\phi(\boldsymbol{x}) + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}')\big)\big|^{-1/2} \exp\!\left(-\frac{1}{2}\,\Delta\boldsymbol{\mu}^\top \big(\ell^2\mathbf{I} + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}')\big)^{-1}\Delta\boldsymbol{\mu}\right), \quad (41)$$

where $\Delta\boldsymbol{\mu} := \boldsymbol{\mu}_\phi(\boldsymbol{x}) - \boldsymbol{\mu}_\phi(\boldsymbol{x}')$.

If the variational posterior uses a diagonal covariance, $\boldsymbol{\Sigma}_\phi(\boldsymbol{x}) = \mathrm{diag}(\sigma_{\phi,1}^2(\boldsymbol{x}), \ldots, \sigma_{\phi,d}^2(\boldsymbol{x}))$. In this case Eq. (41) factorizes over dimensions:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \prod_{j=1}^{d} \left(1 + \frac{\sigma_{\phi,j}^2(\boldsymbol{x}) + \sigma_{\phi,j}^2(\boldsymbol{x}')}{\ell^2}\right)^{-1/2} \exp\!\left(-\frac{(\mu_{\phi,j}(\boldsymbol{x}) - \mu_{\phi,j}(\boldsymbol{x}'))^2}{2\big(\ell^2 + \sigma_{\phi,j}^2(\boldsymbol{x}) + \sigma_{\phi,j}^2(\boldsymbol{x}')\big)}\right), \quad (42)$$

which can be computed in $\mathcal{O}(d)$ time and requires no matrix inversion.

Remark. Unlike the marginalized linear kernel (Section B.2), but similar to the quadratic polynomial kernel (Section B.3), the posterior covariance enters the result explicitly in two distinct ways: (i) an **amplitude modulation** through the determinantal prefactor $|\mathbf{I} + \ell^{-2}(\boldsymbol{\Sigma}_\phi(\boldsymbol{x}) + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}'))|^{-1/2}$, which down-weights kernel evaluations when either input has high posterior uncertainty; and (ii) a **lengthscale inflation** in the Mahalanobis exponent, where the effective squared distance is measured by $(\ell^2\mathbf{I} + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) + \boldsymbol{\Sigma}_\phi(\boldsymbol{x}'))^{-1}$ rather than $\ell^{-2}\mathbf{I}$, making the kernel more tolerant of mean differences when uncertainty is large. When $\boldsymbol{\Sigma}_\phi(\boldsymbol{x}) = \mathbf{0}$ for all $\boldsymbol{x}$ (deterministic encoder), Eq. (41) reduces to the standard RBF kernel evaluated at the posterior means: $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{\mu}_\phi(\boldsymbol{x}) - \boldsymbol{\mu}_\phi(\boldsymbol{x}')\|^2/(2\ell^2))$.

B.5 Connection to Kernel Mean Embeddings

The marginalized kernel introduced in Section 4.1 is closely related to the kernel mean embedding (KME) of distributions [33]. Both constructions rest on the same operation that taking expectations of a kernel function under probability distributions.

Given a positive definite kernel k and its associated reproducing kernel Hilbert space $\mathcal{H}$, the KME maps a distribution P to an element $\mu_P = \mathbb{E}_{\boldsymbol{x}\sim P}[\, k(\cdot, \boldsymbol{x})\,]$ in $\mathcal{H}$ [81, 33]. The marginalized kernel on the other hand defines a kernel between structured objects by taking expectations over conditional distributions of latent variables:

$$k_{\text{marg}}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}|\boldsymbol{x}),\ \boldsymbol{z}'\sim p(\boldsymbol{z}'|\boldsymbol{x}')}\big[\, k_{\text{joint}}\big((\boldsymbol{x}, \boldsymbol{z}),\ (\boldsymbol{x}', \boldsymbol{z}')\big)\big]. \tag{43}$$

This formulation was introduced by Tsuda et al. [56] and Kashima et al. [57] in the context of biological sequences and labeled graphs respectively. When the joint kernel factorizes over the latent variables alone, i.e. $k_{\text{joint}}\big((\boldsymbol{x}, \boldsymbol{z}),\ (\boldsymbol{x}', \boldsymbol{z}')\big) := k_{\boldsymbol{z}}(\boldsymbol{z}, \boldsymbol{z}')$, the marginalized kernel reduces to the inner product of two conditional KMEs in the RKHS:

$$k_{\text{marg}}(\boldsymbol{x}, \boldsymbol{x}') = \big\langle \mu_{p(\boldsymbol{z}|\boldsymbol{x})},\ \mu_{p(\boldsymbol{z}'|\boldsymbol{x}')} \big\rangle_{\mathcal{H}}. \tag{44}$$

In other words, the marginalized kernel measures the similarity between two latent variable distributions after they have been embedded into $\mathcal{H}$, and the KME provides the concrete form of this embedding. In this sense, the marginalized kernel can be viewed as a special case of performing marginalization within the KME framework.

To the best of our knowledge, this work presents the first algorithm that incorporates marginalized kernels into latent space Bayesian optimization.

C Preliminaries on Riemannian Geometry

In this section we briefly review the elements of Riemannian geometry that underpin the geometric perspective on VAEs developed in Sec. 4.2, for readers who are not familiar with the subject.

Riemannian manifolds and the metric tensor. A d-dimensional *Riemannian manifold* $(\mathcal{M}, \mathbf{G})$ is a smooth manifold that is locally homeomorphic to $\mathbb{R}^d$ and is equipped with a smoothly varying inner product on each tangent space. At every point $\boldsymbol{z} \in \mathcal{M}$ the tangent space $T_{\boldsymbol{z}}\mathcal{M}$ consists of all velocity vectors of smooth curves passing through $\boldsymbol{z}$. In a local coordinate system the inner product is represented by a symmetric positive-definite matrix $\mathbf{G}(\boldsymbol{z}) \in \mathbb{R}^{d\times d}$ called the *metric tensor*. For two tangent vectors $\boldsymbol{u}, \boldsymbol{w} \in T_{\boldsymbol{z}}\mathcal{M}$,

$$\langle \boldsymbol{u}, \boldsymbol{w} \rangle_{\boldsymbol{z}} \;=\; \boldsymbol{u}^{\top}\mathbf{G}(\boldsymbol{z})\,\boldsymbol{w}. \tag{45}$$

When $\mathbf{G}(\boldsymbol{z}) = \mathbf{I}_d$ everywhere the manifold reduces to ordinary Euclidean space. A constant non-identity metric $\mathbf{G}(\boldsymbol{z}) = \boldsymbol{\Sigma}^{-1}$ produces the Mahalanobis geometry. This locally varying inner product structure is the foundation of the metric construction in Section 4.2, where we interpret the inverse encoder covariance $\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)$ as the value of a Riemannian metric tensor at the latent mean $\boldsymbol{\mu}(\boldsymbol{x}_i)$, following [70].

Curve length and geodesic distance. Let $\gamma : [0, 1] \to \mathcal{M}$ be a smooth curve with $\gamma(0) = \boldsymbol{z}_1$ and $\gamma(1) = \boldsymbol{z}_2$. Its length on the manifold is

$$L(\gamma) \;=\; \int_0^1 \sqrt{\dot\gamma(t)^{\top}\,\mathbf{G}\big(\gamma(t)\big)\,\dot\gamma(t)}\;\mathrm{d}t, \tag{46}$$

where $\dot\gamma(t) = \mathrm{d}\gamma/\mathrm{d}t$ is the velocity. A curve that minimizes L with fixed endpoints and is parameterized proportionally to arc length is called a *geodesic*. On a connected manifold the *Riemannian distance* between two points is

$$\mathrm{dist}_{\mathbf{G}}(\boldsymbol{z}_1, \boldsymbol{z}_2) \;=\; \inf_{\gamma}\; L(\gamma), \qquad \gamma(0) = \boldsymbol{z}_1,\ \gamma(1) = \boldsymbol{z}_2. \tag{47}$$

In Euclidean space this reduces to the ℓ_2 norm; with a constant metric $\boldsymbol{\Sigma}^{-1}$ it becomes the Mahalanobis distance [67]. In Section 4.2 the Mahalanobis distance induced by $\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)$ serves as the local approximation of the Riemannian distance that enters the interpolation weights $\omega_i(\boldsymbol{z})$ in Eq. (8).

The pull-back metric induced by a generator. Deep generative models define a nonlinear decoder mapping $f : \mathcal{Z} \to \mathcal{X}$ from a low-dimensional latent space to a high-dimensional observation space. A first-order Taylor expansion shows that a small displacement $\Delta \boldsymbol{z}$ in latent space is mapped to $\mathbf{J}_{\boldsymbol{z}}\, \Delta \boldsymbol{z}$ in observation space, where $\mathbf{J}_{\boldsymbol{z}} = \partial f / \partial \boldsymbol{z}$ is the Jacobian. The squared observation-space length of this displacement equals $\Delta \boldsymbol{z}^{\top} (\mathbf{J}_{\boldsymbol{z}}^{\top} \mathbf{J}_{\boldsymbol{z}})\, \Delta \boldsymbol{z}$, so the matrix

$$\mathbf{G}(\boldsymbol{z}) \;=\; \mathbf{J}_{\boldsymbol{z}}^{\top}\, \mathbf{J}_{\boldsymbol{z}} \tag{48}$$

is a natural Riemannian metric on $\mathcal{Z}$, commonly called the *pull-back metric* [67, 82]. It quantifies how the generator distorts infinitesimal volumes when mapping from latent space to observation space. Computing $\mathbf{J}_{\boldsymbol{z}}$ requires differentiating the entire decoder, which can be expensive when $\mathcal{X}$ is high-dimensional. Chadebec and Allassonnière [70] showed that the encoder-based metric $\mathbf{G}(\boldsymbol{\mu}(\boldsymbol{x}_i)) = \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)$ approximates the pull-back metric near $\boldsymbol{\mu}(\boldsymbol{x}_i)$ under mild regularity conditions, while completely avoiding the Jacobian computation. Section 4.2 adopts this approximation to construct a tractable metric for sampling.

Volume element and the magnification factor. Given the metric tensor $\mathbf{G}(\boldsymbol{z})$ a canonical volume element on $\mathcal{M}$ is defined as [83]

$$\mathrm{d}\mathcal{M}_{\boldsymbol{z}} \;=\; \sqrt{\det \mathbf{G}(\boldsymbol{z})}\; \mathrm{d}\boldsymbol{z}, \tag{49}$$

where $\mathrm{d}\boldsymbol{z}$ denotes the Lebesgue measure. The scalar $\mathrm{MF}(\boldsymbol{z}) = \sqrt{\det \mathbf{G}(\boldsymbol{z})}$ is known as the *magnification factor* and quantifies how much an infinitesimal region around $\boldsymbol{z}$ is stretched or compressed under the generator mapping. This volume element directly enters the sampling density $p_{\text{Riem}}(\boldsymbol{z})$ constructed in Eq. (9) of Section 4.2. Regions with a large magnification factor receive higher sampling probability, thereby concentrating candidate generation on geometrically well-supported areas of the latent space.

Probability distributions on Riemannian manifolds. The volume element in Eq. (49) serves as the reference measure against which densities on $\mathcal{M}$ can be defined [83]. Two distributions are particularly relevant to our work.

Riemannian Gaussian. Replacing the Euclidean distance with the Riemannian distance of Eq. (47) yields

$$\mathcal{N}_{\text{riem}}(\boldsymbol{z} \mid \mu, \sigma) \;=\; \frac{1}{C}\, \exp\!\Big(-\frac{\mathrm{dist}_{\mathbf{G}}(\boldsymbol{z}, \mu)^2}{2\sigma}\Big), \quad C = \int_{\mathcal{M}} \exp\!\Big(-\frac{\mathrm{dist}_{\mathbf{G}}(\boldsymbol{z}, \mu)^2}{2\sigma}\Big) \mathrm{d}\mathcal{M}_{\boldsymbol{z}}. \tag{50}$$

When $\mathbf{G}$ is everywhere equal to the constant $\boldsymbol{\Sigma}^{-1}$ this reduces to an ordinary multivariate Gaussian with covariance $\boldsymbol{\Sigma}$ [70]. This observation justifies viewing the VAE encoder posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_i) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_i), \boldsymbol{\Sigma}(\boldsymbol{x}_i))$ as a Riemannian Gaussian under the locally constant metric $\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i)$.

Riemannian uniform. If the total volume of $\mathcal{M}$ is finite one can define a uniform distribution on the manifold [83, 70]

$$\mathcal{U}_{\text{riem}}(\boldsymbol{z}) \;=\; \frac{\sqrt{\det \mathbf{G}(\boldsymbol{z})}}{\displaystyle\int_{\mathbb{R}^d} \sqrt{\det \mathbf{G}(\boldsymbol{z})}\; \mathrm{d}\boldsymbol{z}}, \tag{51}$$

whose density with respect to the Lebesgue measure is proportional to the magnification factor. Sampling from this distribution explores the manifold in a geometry-aware fashion: high-density regions correspond to areas with well-supported latent geometry. This is the target density $p_{\text{Riem}}(\boldsymbol{z})$ in Eq. (9) from which we draw latent candidates in Section 4.2 via the NUTS sampler [25].

D Additional Experiments

In this section, we provide additional experimental results on the PMO benchmark under both the small-budget setting (Appendix D.1) and the large-budget setting (Appendix D.2).

D.1 Small Budget Setting on PMO Benchmark

As shown in Figure 6, we further evaluate MarKBO under the small-budget setting on twelve additional PMO tasks (Albuterol Similarity, Valsartan SMARTS, Isomer C9H10N2O2PF2Cl, Isomer

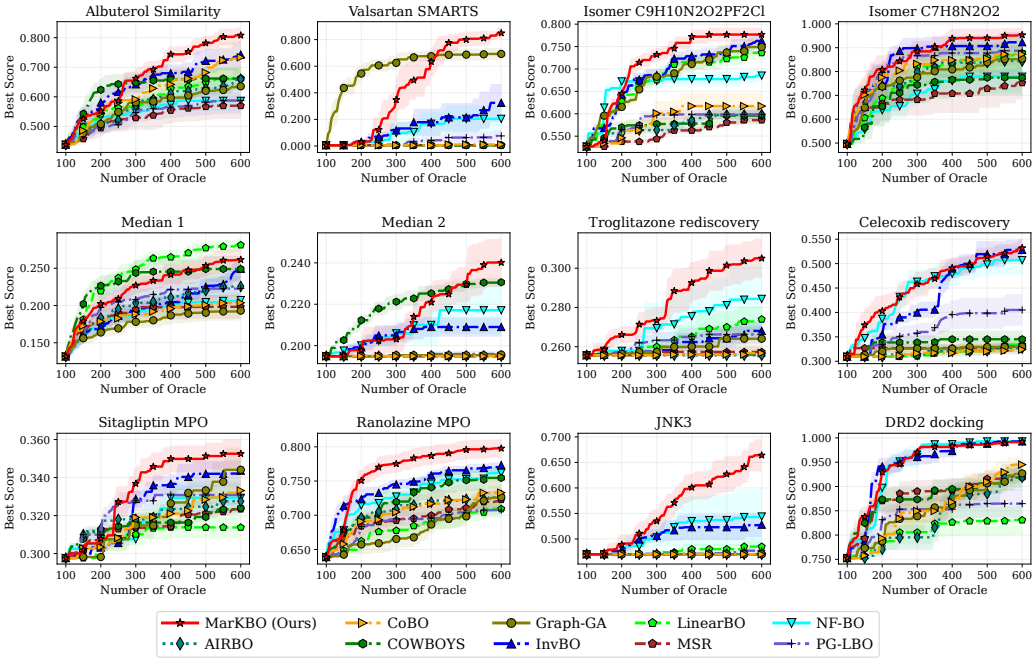


Figure 6: Performance of MarKBO against nine baseline methods on 12 small-budget PMO tasks. The solid curve shows the mean best score over 10 random seeds, and the shaded region indicates the standard deviation. The x-axis denotes the number of objective function evaluations.

C7H8N2O2, Median 1, Median 2, Troglitazone Rediscovery, Celecoxib Rediscovery, Sitagliptin MPO, Ranolazine MPO, JNK3 and DRD2 Docking). Each method starts from 100 initial molecules and is allowed 500 additional oracle calls with a batch size of 5. Overall, MarKBO achieves the highest final best score on 11 out of 12 tasks. On Valsartan SMARTS and Median 2, MarKBO starts below some baselines during the early optimization stage but surpasses all baselines by the end of the budget. The one exception is Median 1, where LinearBO maintains a higher best score than all baselines across the entire budget. The results confirm that MarKBO provides strong performance under the small-budget setting across a diverse set of molecular optimization objectives.

Figure 7 presents the complete violin plots of oracle scores for all compared methods on the four small-budget PMO tasks in Sec. 5.1. Each violin aggregates the scores of all deduplicated molecules generated across 10 independent runs. On all four tasks, MarKBO achieves the highest median score among all baselines. Across all four tasks, the score distributions of MarKBO are not only shifted toward higher values but also exhibit relatively compact shapes with less mass in the low-score tail. This indicates that MarKBO can generate diverse high-quality candidates.

D.2 Large Budget Setting on PMO Benchmark

As shown in Figure 8, we further evaluate MarKBO under the large-budget setting on four additional PMO tasks. MarKBO achieves the highest best scores across all four tasks as the number of oracle evaluations increases from 10K to 20K. Each method starts from 10000 initial molecules and is allowed 10000 additional oracle calls with a batch size of 100. On Amlodipine MPO, Median 2 and Zaleplon MPO, MarKBO surpasses all baselines throughout the optimization process. On Ranolazine MPO, MarKBO also reaches better performance compared to competitive baselines such as NF-BO and InvBO. This scalability benefits from two design choices. The marginalized kernel admits a closed-form expression that avoids costly posterior sampling at kernel evaluation. The Riemannian geometry-aware NUTS sampler operates on the metric $\mathbf{G}(\boldsymbol{z})$ interpolated in closed form from encoder outputs, so incorporating new observations only requires updating the centroid set rather than retraining the VAE. These results confirm that MarKBO scales effectively to large-budget setting across diverse molecular optimization objectives.

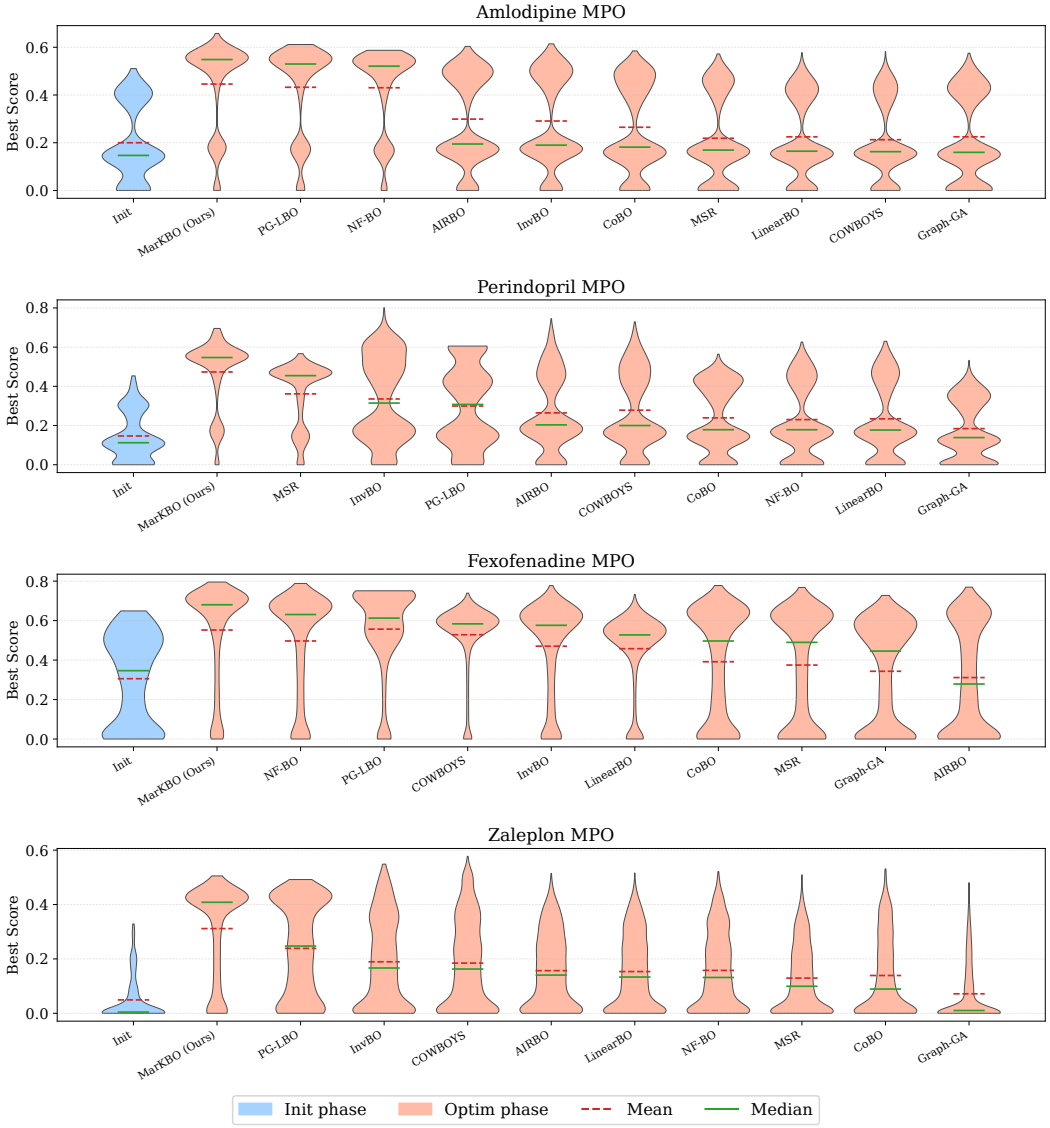


Figure 7: Violin plots of oracle scores for all deduplicated molecules generated during optimization on four small-budget PMO tasks over 10 seeds. Methods are sorted by median score in descending order from left to right. Green solid lines indicate medians and red dashed lines indicate means.

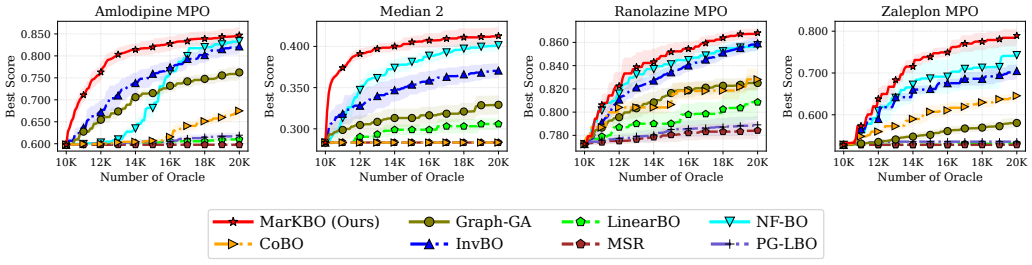


Figure 8: Performance of MarKBO against seven baseline methods on four large-budget PMO tasks. The solid curve shows the mean best score over 10 random seeds, and the shaded region indicates the standard deviation. The x-axis denotes the number of objective function evaluations.

E Experimental Details

In this section, we provide the implementation details underlying all experiments in Section 5. Section E.1 introduces the deep generative models used for each task. Section E.2 introduces the surrogate model and Section E.3 introduces the acquisition strategies. Section E.4 introduces the baseline methods. Section E.5 reports the computational resources. Section E.6 lists the licenses for existing assets used in this work.

E.1 Deep Generative Models

E.1.1 Molecular Design

The generative model employed for molecular optimization in Sec. 5.1 is a Transformer-based variational autoencoder (SELFIES-VAE [45]) that establishes a mapping between discrete SELFIES [74] token sequences and a continuous latent space.

Encoder. Given a SELFIES token sequence $\boldsymbol{x}_{1:T}$, the encoder maps each token through an embedding matrix $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{V\times 2d}$ into a $2d$-dimensional space. After adding sinusoidal positional encodings, the resulting representations are passed through an L_{enc}-layer Transformer encoder to produce hidden states $\mathbf{H} \in \mathbb{R}^{T\times 2d}$. The feature dimension of $\mathbf{H}$ is split in half and only the first B sequence positions are retained, yielding the posterior parameters $\boldsymbol{\mu} = \mathbf{H}_{[1:B,\,:d]}$ and $\boldsymbol{\sigma} = \text{softplus}(\mathbf{H}_{[1:B,\,d:2d]}) + \epsilon_{\min}$. These define a diagonal Gaussian posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_{1:T}) = \mathcal{N}(\boldsymbol{z};\, \boldsymbol{\mu},\, \text{diag}(\boldsymbol{\sigma}^2))$ with $\boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{B\times d}$. Latent variables are obtained via the reparameterization trick as $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}$ with $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. During latent space Bayesian optimization, $\boldsymbol{z}$ is flattened into an $\mathbb{R}^{Bd}$ vector that serves as the search space.

Decoder. The decoder is an L_{dec}-layer Transformer decoder that treats $\boldsymbol{z}$ as memory. At each generation step it attends to the latent representation through cross-attention, modeling the autoregressive distribution $p_\theta(\boldsymbol{x}_{1:T} \mid \boldsymbol{z}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{<t}, \boldsymbol{z})$. At inference time, tokens are sampled autoregressively until a stop token is produced or the maximum sequence length is reached.

Training objective. The model is trained by minimizing a weighted negative ELBO [24]:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}, \boldsymbol{z}) \;+\; \beta\, D_{\text{KL}}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_{1:T}) \,\big\|\, p(\boldsymbol{z})\big), \tag{52}$$

where the prior is a standard Gaussian $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Hyperparameters. Table 6 summarizes the hyperparameters of the SELFIES-VAE. All settings follow the publicly released code implementation of [45]. In our experiments, we initialize the VAE with the pretrained weights provided in the public repository of [45].[2]

E.1.2 Automated Robotic Design

The generative model employed for robot design optimization in Sec. 5.1 is a graph variational autoencoder adapted from the Junction Tree VAE (JT-VAE [7]), which establishes a mapping between tree-structured robot design graphs and a continuous latent space [14].[3]

Data Generation. Training data is collected using the `collect_data.py` script released by [14], which stochastically samples robot morphologies from the RoboGrammar production grammar [75]. Starting from the initial nonterminal start symbol, grammar rules are applied in uniformly random order up to a maximum derivation depth of $D_{\max} = 40$ steps. Each candidate design undergoes several validity filters: designs with fewer than $N_{\min} = 11$ links are discarded, self-colliding initial configurations (detected via physics pre-simulation in Bullet [84]) are rejected, and designs must exhibit at least four non-degenerate ground-contact points. Valid designs are converted into symmetric adjacency matrices $\mathbf{A} \in \{0,1\}^{n\times n}$ and discrete node-type vectors $\boldsymbol{f} \in \{0, \ldots, |\mathcal{V}| - 1\}^n$, where n

[2] Our implementation follows the publicly released code at https://github.com/nataliemaus/lolbo.

[3] Our implementation follows the publicly released code at https://github.com/JiahengHu/GLSO.

Table 6: Hyperparameters of the SELFIES-VAE [45] for experiment.

Notation	Description	Value
d	Transformer hidden dimension	128
$2d$	Encoder embedding dimension	256
B	Bottleneck size	2
Bd	Flattened latent dimension	256
L_{enc}	Number of encoder layers	6
L_{dec}	Number of decoder layers	6
$n_{\text{head}}^{\text{enc}}$	Number of encoder attention heads	8
$n_{\text{head}}^{\text{dec}}$	Number of decoder attention heads	8
$d_{\text{ff}}^{\text{enc}}$	Encoder feed-forward dimension	512
$d_{\text{ff}}^{\text{dec}}$	Decoder feed-forward dimension	256
$p_{\text{drop}}^{\text{enc}}$	Encoder dropout rate	0.1
$p_{\text{drop}}^{\text{dec}}$	Decoder dropout rate	0.1
β	KL divergence weight	0.1
$\epsilon_{\min}$	Minimum posterior standard deviation	10^{-4}

is the number of nodes and each type index jointly encodes one of three capsule link geometries and one of fourteen joint configurations, yielding $|\mathcal{V}| = 42$ distinct node types in total. Ground-contact locations $\boldsymbol{c} \in \mathbb{R}^C$ consisting of the horizontal-plane coordinates of up to eight contact points sorted by the x-value and zero-padded to a fixed length $C = 16$, are additionally recorded for auxiliary supervision. A total of 500,000 valid designs are collected in this manner. For the optimization tasks in Sec. 5.1, the first 100 designs from this dataset are used as initial evaluations for all methods.

Encoder. The encoder is a message-passing neural network operating on the tree-structured robot graph $\mathcal{G} = (\mathbf{A}, \boldsymbol{f})$. Each node carries a discrete type $f_i \in \{0, \dots, |\mathcal{V}| - 1\}$, which is mapped to a d-dimensional embedding $\boldsymbol{x}_i = \mathbf{W}_{\text{emb}}[f_i]$ via a learned embedding matrix $\mathbf{W}_{\text{emb}} \in \mathbb{R}^{|\mathcal{V}| \times d}$. A Graph GRU then performs D rounds of message passing along the tree edges. Let $\boldsymbol{x}_m$ denote the embedding of the source node of message m, let $\mathcal{N}(m)$ denote the set of messages feeding into m, and let all message states be initialized as $\mathbf{h}_m^{(0)} = \mathbf{0}$. Each round $\ell = 1, \dots, D$ updates

$$\mathbf{s}_m^{(\ell)} = \textstyle\sum_{k \in \mathcal{N}(m)} \mathbf{h}_k^{(\ell-1)}, \tag{53}$$

$$\mathbf{u}_m^{(\ell)} = \sigma\big(\mathbf{W}_u \, [\boldsymbol{x}_m;\ \mathbf{s}_m^{(\ell)}]\big), \tag{54}$$

$$\mathbf{r}_k^{(\ell)} = \sigma\big(\mathbf{W}_r \, \boldsymbol{x}_m + \mathbf{U}_r \, \mathbf{h}_k^{(\ell-1)}\big), \tag{55}$$

$$\mathbf{h}_m^{(\ell)} = \big(1 - \mathbf{u}_m^{(\ell)}\big) \odot \mathbf{s}_m^{(\ell)} + \mathbf{u}_m^{(\ell)} \odot \tanh\big(\mathbf{W}_h \, [\boldsymbol{x}_m;\ \textstyle\sum_{k \in \mathcal{N}(m)} \mathbf{r}_k^{(\ell)} \odot \mathbf{h}_k^{(\ell-1)}]\big), \tag{56}$$

where $\sigma(\cdot)$ is the sigmoid function, $[\cdot\,;\,\cdot]$ denotes vector concatenation, $\mathbf{u}_m^{(\ell)}$ is the update gate, and $\mathbf{r}_k^{(\ell)}$ is the reset gate. After D rounds, each node representation is formed by concatenating its embedding with the sum of its incoming final messages and projecting through a linear layer with ReLU activation. A graph-level representation $\mathbf{g}$ is obtained by summing the representations of all leaf nodes, defined as nodes with exactly one neighbor whose link type is not the body capsule. The posterior parameters are computed as $\boldsymbol{\mu} = \mathbf{W}_\mu \, \mathbf{g}$ and $\log \boldsymbol{\sigma}^2 = -|\mathbf{W}_\sigma \, \mathbf{g}|$, where the negative absolute value constrains every component of $\boldsymbol{\sigma}^2$ to be at most one. These define a diagonal Gaussian posterior $q_\phi(\boldsymbol{z} \mid \mathcal{G}) = \mathcal{N}(\boldsymbol{z};\, \boldsymbol{\mu},\, \mathrm{diag}(\boldsymbol{\sigma}^2))$ with $\boldsymbol{z} \in \mathbb{R}^L$. Latent samples are obtained via the reparameterization trick as $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}$ with $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. During latent space optimization, the L-dimensional vector $\boldsymbol{z}$ serves as the continuous search space.

Decoder. The decoder autoregressively reconstructs a tree-structured design graph conditioned on the latent vector $\boldsymbol{z}$. Generation proceeds in depth-first order: starting from a predicted root node type, the decoder at each visited node i first decides whether to expand (add a child) or backtrack to its parent; if expanding, it predicts the child's node type from the vocabulary $\mathcal{V}$. Both decisions rely on GRU-based hidden states that aggregate messages from already-generated neighbors. Concretely, let $\boldsymbol{e}_i$ denote the embedding of the current node, let $\mathbf{o}_i = \sum_{k \to i} \mathbf{h}_{k,i}$ denote the sum of incoming

neighbor messages, and let $\mathbf{h}_{\text{new}}$ be the hidden state obtained by applying the GRU to neighbors excluding the target direction. The node-type distribution and expand probability are

$$p(v_{\text{child}} \mid \mathbf{h}_{\text{new}}, \boldsymbol{z}) = \text{softmax}\big(\mathbf{W}_o^w \,\text{ReLU}\big(\mathbf{W}^w\,[\mathbf{h}_{\text{new}};\ \boldsymbol{z}]\big)\big), \tag{57}$$

$$p(\text{expand} \mid \boldsymbol{e}_i, \mathbf{o}_i, \boldsymbol{z}) = \sigma\big(\mathbf{U}_o^s \,\text{ReLU}\big(\mathbf{U}^s\,\big[\text{ReLU}\big(\mathbf{U}_i^s\,[\boldsymbol{e}_i;\ \mathbf{o}_i]\big);\ \boldsymbol{z}\big]\big)\big), \tag{58}$$

where $\mathbf{W}^w$, $\mathbf{W}_o^w$ are the node-type prediction weights and $\mathbf{U}_i^s$, $\mathbf{U}^s$, $\mathbf{U}_o^s$ are the backtrack prediction weights. Generation terminates when the decoder backtracks to the root or a maximum of $T_{\max}$ decoding steps is reached.

Contact Predictor. An auxiliary multi-layer perceptron referred to as the property prediction network (PPN) [14] predicts ground-contact locations $\hat{\boldsymbol{c}} \in \mathbb{R}^C$ directly from the posterior mean $\boldsymbol{\mu}$. By encouraging designs with similar physical footprints to lie close together in the latent space, this module facilitates more efficient downstream Bayesian optimization.

Training Objective. The model is trained by minimizing a composite loss:

$$\mathcal{L} = \alpha\,\mathcal{L}_{\text{word}} + \mathcal{L}_{\text{topo}} + \beta\,D_{\text{KL}}\big(q_\phi(\boldsymbol{z} \mid \mathcal{G})\ \big\|\ p(\boldsymbol{z})\big) + \gamma\,\mathcal{L}_{\text{pred}}, \tag{59}$$

where $\mathcal{L}_{\text{word}}$ is the cross-entropy loss for node-type prediction, $\mathcal{L}_{\text{topo}}$ is the binary cross-entropy loss for expand/backtrack decisions, D_{KL} denotes the Kullback–Leibler divergence from the Gaussian prior $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\mathcal{L}_{\text{pred}} = \|\hat{\boldsymbol{c}} - \boldsymbol{c}\|_2^2$ is the mean squared error for contact-location prediction.

Training. The composite loss $\mathcal{L}$ is minimized using the Adam optimizer with a learning rate of 10^{-3}. An exponential learning rate decay with factor 0.9 is applied every $40{,}000$ iterations. Gradient norms are clipped to 50.0, and the batch size is set to 32. The KL weight β follows an annealing schedule. It remains at zero for a warmup period of $\tau_{\text{warm}} = 40{,}000$ iterations, after which it increases by $\Delta\beta = 0.002$ every $\tau_{\text{KL}} = 2{,}000$ iterations until reaching $\beta_{\max} = 1.0$. Training proceeds for $400{,}000$ iterations in total. The resulting weights are used directly as the pretrained VAE for latent space optimization. The VAE weights are kept frozen throughout the optimization phase.

Table 7 summarizes the architectural hyperparameters and fixed loss weights of the graph VAE for robot design. All settings follow the publicly released implementation of [14].

Table 7: Architectural hyperparameters and loss weights of the GLSO-VAE [14] for experiment.

Notation	Description	Value
d	Hidden dimension	450
L	Latent dimension	28
D	Message passing depth	20
$\lvert\mathcal{V}\rvert$	Vocabulary size (node types)	42
C	Contact vector dimension	16
$T_{\max}$	Maximum decoding steps	100
n_{ppn}	Number of PPN hidden layers	2
d_{ppn}	PPN hidden layer width	128
α	Node-type loss weight	1.0
γ	Contact prediction loss weight	20

E.2 Surrogate Model

Following prior works [45, 17–19], we employ a sparse variational Gaussian process (SVGP) [52] as the surrogate model implemented with GPyTorch [85] and BoTorch [86] for MarKBO and other BO baselines. The SVGP adopts a Cholesky variational distribution with inducing points randomly sampled from the current training inputs. The offset c in the quadratic base kernel is a learnable kernel hyperparameter. Following the parameterization of GPyTorch's `PolynomialKernel`, we enforce positivity through $c = \text{softplus}(\rho) = \ln(1 + \exp(\rho))$, where $\rho \in \mathbb{R}$ is a trainable parameter. The raw parameter is initialized to $\rho = 0$ corresponding to an initial offset $c_0 = \log 2$. Inducing point locations are jointly optimized during training. All variational parameters and kernel hyperparameters are optimized by maximizing the predictive log-likelihood (PLL) with the Adam optimizer.

For the training schedule we adopt a two-phase strategy [45]. At the beginning of optimization the surrogate is trained on all available initial observations for a larger number of epochs. In all subsequent iterations the model is updated for fewer epochs using only the newly acquired batch.

E.3 Acquisition Strategy

For the acquisition function of each BO baseline we follow the choice made in its original public implementation. The baselines considered in this work involve two acquisition strategies. The first is Expected Improvement (EI), for which we adopt `qLogExpectedImprovement` (qLogEI) [54] from BoTorch [86]. The qLogEI acquisition function is optimized via multi-start L-BFGS-B through the `optimize_acqf` utility in BoTorch.

The second strategy is Thompson Sampling (TS) which draws a sample function from the surrogate posterior and returns its maximizer as the next evaluation point. We follow the discrete TS [45] where a large set of candidate points is generated and the candidates with the highest posterior sample values are selected as the next batch. This is realized through the `MaxPosteriorSampling` class in BoTorch [86]. We adopt this strategy for MarKBO similar to recent LSBO works [45, 17–19].

E.4 Baselines

We briefly describe the idea of each baseline in Section 5.2 and refer readers to the original papers for full details. Unless otherwise specified, our implementations of all baseline methods are based on the official publicly available codebases released by the respective authors. Where modifications are made (e.g., acquisition strategy), we explicitly note them in the corresponding descriptions below.

E.4.1 Molecular Design

For the PMO benchmark, we compare our method against the following nine baselines:

Graph-GA [76]. Graph-GA is a graph-based genetic algorithm that operates directly on molecular graph representations, using graph-based crossover and mutation to optimize molecular properties.

CoBO [17]. CoBO addresses the inherent gap in latent space Bayesian optimization by fine-tuning a correlated latent space where Euclidean distances between latent vectors are correlated with differences in objective function values. It introduces Lipschitz regularization to encourage smoothness of the latent landscape with respect to the objective function, a loss weighting scheme that prioritizes reconstruction fidelity in promising high-value regions, and trust region recoordination to recenter the search space after latent space updates.

InvBO [18]. InvBO is a plug-and-play module for VAE-based latent space Bayesian Optimization that addresses the "misalignment problem" where reconstruction errors cause a single latent vector $\boldsymbol{z}$ to be associated with conflicting objective values y. It introduces an optimization-based inversion method that finds latent points $\boldsymbol{z}$ perfectly reconstructing original data $\boldsymbol{x}$.

PG-LBO [42]. PG-LBO enhances VAE-based latent space Bayesian optimization by leveraging unlabeled data through pseudo-labeling. It uses the GP posterior mean to assign pseudo-labels to unlabeled points sampled in the latent space, then applies rank-based weighting to these pseudo-labeled points during VAE retraining, encouraging the model to focus on high-scoring regions.

COWBOYS [21]. Categorical Optimisation With Belief Of underlYing Structure (COWBOYS) decouples the VAE and GP surrogate in latent space Bayesian optimisation by fitting the GP directly in the structure space using a Tanimoto kernel on molecular fingerprints rather than in the latent space. It combines the VAE's prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ with the GP's predictive distribution via a Bayesian update rule, sampling candidate structures from the GP-conditioned VAE using Preconditioned Crank-Nicolson MCMC (PCN-MCMC).

NF-BO [19]. Normalizing Flow-based Bayesian Optimization (NF-BO) addresses the value discrepancy problem in latent space Bayesian optimization. Similar to InvBO, this problem arises because VAE reconstruction errors cause inconsistency between objective values y evaluated at original inputs $\boldsymbol{x}$ and their decoded reconstructions $\hat{\boldsymbol{x}}$. NF-BO introduces SeqFlow, an autoregressive normalizing flow for discrete sequences that establishes a one-to-one mapping from the input space to the latent space to ensure better reconstruction.

AIRBO [36]. AIRBO models uncertain inputs of arbitrary distributions by replacing the standard kernel in GP with one based on Maximum Mean Discrepancy (MMD), which measures the distance between input distributions in a reproducing kernel Hilbert space. It further applies Nyström approximation to reduce the space complexity of MMD estimation and accelerate posterior inference.

MSR [51]. MLE Scaled with RAASP (MSR) addresses the vanishing gradient problem that arises when fitting Gaussian Process lengthscales in high-dimensional spaces. It initializes the lengthscale at $1/10\sqrt{d}$ to avoid gradient vanishing in the marginal log-likelihood optimization and combines MLE with RAASP sampling to promote local search behavior during acquisition function maximization.

LinearBO [77]. LinearBO uses a GP with a linear kernel for high-dimensional Bayesian optimization. To address the boundary seeking pathology of standard linear models, it applies an inverse stereographic projection to bijectively map the hypercube input space onto a unit hypersphere before computing the kernel. In our experiments, we replace the original EI acquisition function with `qLogExpectedImprovement` [86, 54] to support batch evaluation.

E.4.2 Automated Robotic Design

For the Automated Robot Design task, in addition to the baselines compared in the GLSO paper [14], we further include state-of-the-art VAE-LSBO (InvBO), HDBO (LinearBO, MSR) and RBO (AIRBO) methods as additional comparisons. Since these methods have already been described in Section E.4.1, we do not repeat their descriptions here.

LSBO [14]. Grammar-guided Latent Space Optimization (GLSO) trains a graph variational autoencoder to map the combinatorial robot design space into a low-dimensional continuous latent space. It then performs BO in the latent space using a GP surrogate and EI as the acquisition function. In our experiments, we replace the original EI implementation with `qLogExpectedImprovement` [86, 54] to support batch evaluation.

Random [14]. Random Search generates candidate robot designs by randomly applying graph grammar rules until a complete design graph consisting of only terminal symbols is obtained.

MCTS [75]. Monte Carlo Tree Search operates on a search tree defined by the graph grammar rules and uses the Upper Confidence bounds to balance exploration and exploitation when selecting which partial designs to expand.

GHS [75]. Graph Heuristic Search trains a graph neural network (GNN) as a heuristic function to predict the best achievable reward for partial robot designs. Design candidates are generated via ϵ-greedy rule selection guided by this GNN.

GA [78]. The Genetic Algorithm follows [78] and maintains a population of candidate robot designs that are iteratively evolved through graph mutation operations. A population of 50 individuals is mutated and selected each generation until the evaluation budget is exhausted.

E.5 Compute Resources

All experiments were conducted on a single workstation equipped with an AMD Ryzen Threadripper PRO 3975WX 32-Core processor (64 virtual CPUs), 128 GB of RAM, and a single NVIDIA L40S GPU (48 GB VRAM).

E.6 Licenses for Existing Assets

We list the existing assets used in our implementation along with their licenses below.

Datasets.

- **GuacaMol** [22]: Released under the MIT License. Available at `https://github.com/BenevolentAI/guacamol`.
- **RoboGrammar** [75]: Released under the MIT License. Available at `https://github.com/allanzhao/RoboGrammar`.

Libraries.

- **BoTorch** [86]: Released under the MIT License. Available at https://github.com/pytorch/botorch.
- **GPyTorch** [85]: Released under the MIT License. Available at https://github.com/cornellius-gp/gpytorch.
- **Pyro** [71]: Released under the Apache License 2.0. Available at https://github.com/pyro-ppl/pyro.
- **Poli** [73]: Released under the MIT License. Available at https://github.com/MachineLearningLifeScience/poli.

F Pseudocode and Hyperparameters of MarKBO

Algorithm 1 describes the vanilla VAE-LSBO. The method begins by sampling n_0 structures from $\mathcal{X}$ and encoding each into the latent space $\mathcal{Z}$ via the reparameterization trick. The black-box objective is evaluated on all initial structures to form the observation dataset $\mathcal{D}_0$ and its latent counterpart $\mathcal{D}_0^{\boldsymbol{z}}$. At each iteration t a GP surrogate is fitted on $\mathcal{D}_{t-1}^{\boldsymbol{z}}$ with kernel hyperparameters optimized by maximizing the marginal log-likelihood. A batch of B latent candidates is then selected by jointly maximizing the acquisition function α_t over $\mathcal{Z}^B$. Each candidate $\boldsymbol{z}_{t,j}$ is decoded by sampling from the generative distribution $p_\theta(\boldsymbol{x} \mid \boldsymbol{z}_{t,j})$ and the resulting structure is evaluated under f. The B evaluations within each batch can be performed in parallel to reduce wall-clock time. Setting $B = 1$ recovers the standard sequential optimization.

Algorithm 1 VAE-based Latent Space Bayesian Optimization (VAE-LSBO)

Input: Structured domain $\mathcal{X}$, black-box objective $f : \mathcal{X} \to \mathbb{R}$, pre-trained VAE encoder $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and decoder $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$, GP prior $\mathcal{GP}(m(\cdot), k(\cdot,\cdot))$, acquisition utility function $u_t(\cdot)$, initial dataset size n_0, batch size B, number of iterations T

Output: Best observed input $\boldsymbol{x}^*$ and value y^*

Sample $\{\boldsymbol{x}_i\}_{i=1}^{n_0}$ from $\mathcal{X}$
for $i = 1, \dots, n_0$ **do**
  $\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i \leftarrow q_\phi(\cdot \mid \boldsymbol{x}_i)$
  $\boldsymbol{z}_i \leftarrow \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \odot \boldsymbol{\epsilon}_i,\ \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  $y_i \leftarrow f(\boldsymbol{x}_i)$
end for
$\mathcal{D}_0 \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_0},\ \mathcal{D}_0^{\boldsymbol{z}} \leftarrow \{(\boldsymbol{z}_i, y_i)\}_{i=1}^{n_0}$
$(\boldsymbol{x}^*, y^*) \leftarrow \arg\max_{(\boldsymbol{x},y)\in\mathcal{D}_0} y$
for $t = 1, \dots, T$ **do**
  Fit GP on $\mathcal{D}_{t-1}^{\boldsymbol{z}}$: $\hat{\boldsymbol{\psi}} \leftarrow \arg\max_{\boldsymbol{\psi}} \log p(\boldsymbol{y} \mid \boldsymbol{Z}, \boldsymbol{\psi})$
  $p(f \mid \mathcal{D}_{t-1}^{\boldsymbol{z}}) = \mathcal{GP}\big(m_{t-1}(\cdot),\ k_{t-1}(\cdot,\cdot)\big)$
  $\alpha_t(\boldsymbol{z}_{1:B}) \leftarrow \mathbb{E}_{f\sim p(f\mid\mathcal{D}_{t-1}^{\boldsymbol{z}})}[u_t(f,\ \boldsymbol{z}_{1:B})]$
  $\boldsymbol{z}_{t,1:B} \leftarrow \arg\max_{\boldsymbol{z}_{1:B}\in\mathcal{Z}^B} \alpha_t(\boldsymbol{z}_{1:B})$
  for $j = 1, \dots, B$ **do**
    $\hat{\boldsymbol{x}}_{t,j} \sim p_\theta(\boldsymbol{x} \mid \boldsymbol{z}_{t,j})$
    $y_{t,j} \leftarrow f(\hat{\boldsymbol{x}}_{t,j})$
  end for
  $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\hat{\boldsymbol{x}}_{t,j},\ y_{t,j})\}_{j=1}^{B},\ \mathcal{D}_t^{\boldsymbol{z}} \leftarrow \mathcal{D}_{t-1}^{\boldsymbol{z}} \cup \{(\boldsymbol{z}_{t,j},\ y_{t,j})\}_{j=1}^{B}$
  $j^* \leftarrow \arg\max_{j\in\{1,\dots,B\}} y_{t,j}$
  if $y_{t,j^*} > y^*$ **then**
    $\boldsymbol{x}^* \leftarrow \hat{\boldsymbol{x}}_{t,j^*},\ y^* \leftarrow y_{t,j^*}$
  end if
end for
return $\boldsymbol{x}^*,\ y^*$

Algorithm 2 describes latent geometry-aware sampling in Sec. 4.2. Given a labeled dataset and a pre-trained VAE, the algorithm first encodes all N observations to obtain their latent means $\boldsymbol{\mu}(\boldsymbol{x}_i)$ and covariances $\boldsymbol{\Sigma}(\boldsymbol{x}_i)$. It then ranks the observations by their objective values and retains the top $p\%$ as centroids ($K = \lceil pN/100 \rceil$), from which the Riemannian metric tensor $\mathbf{G}(\boldsymbol{z})$ is constructed following Eqs. (7)–(8). Next, n_{cand} latent points are drawn from the Riemannian uniform distribution (Eq. (9)) using NUTS [25], which adaptively sets the trajectory length in HMC and requires only a target potential energy and its gradient. In our setting the potential energy $U(\boldsymbol{z})$ is given by Eq. (10) and the gradient by Eq. (11). All NUTS internal details follow the procedure described in [25]. Finally, sampled latent points are decoded through $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ to produce the generated data set $\mathcal{X}_{\text{gen}}$.

Algorithm 2 Latent Geometry-Aware Sampling from a VAE

Input: Pre-trained VAE encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, labeled dataset $\{(\boldsymbol{x}_i,\, y_i)\}_{i=1}^N$, top percentage $p \in (0, 100]$, desired number of generated samples n_{cand}, regularization strength $\lambda > 0$, decay parameter $\tau \geq 0$.
Output: Generated data samples $\mathcal{X}_{\text{gen}}$

for $i = 1, \ldots, N$ **do**
  $\boldsymbol{\mu}(\boldsymbol{x}_i),\ \boldsymbol{\Sigma}(\boldsymbol{x}_i) \leftarrow q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_i)$
end for

Rank $\{(\boldsymbol{x}_i,\, y_i)\}_{i=1}^N$ by y_i in descending order
$K \leftarrow \lceil\, pN/100 \,\rceil$
Select the top-K entries as centroids with latent posteriors $\{(\boldsymbol{\mu}(\boldsymbol{x}_i),\, \boldsymbol{\Sigma}(\boldsymbol{x}_i))\}_{i=1}^K$

$\rho \leftarrow \max_i \min_{j \neq i} \|\boldsymbol{\mu}(\boldsymbol{x}_i) - \boldsymbol{\mu}(\boldsymbol{x}_j)\|_2$ ◁ kernel bandwidth
Construct metric tensor $\mathbf{G}(\boldsymbol{z})$ via Eqs. (7)–(8)

Draw n_{cand} latent samples Z_{gen} from $p_{\text{Riem}}(\boldsymbol{z}) \propto \sqrt{\det \mathbf{G}(\boldsymbol{z})}$ using NUTS [25] ◁ Eq. (9)

$\mathcal{X}_{\text{gen}} \leftarrow \emptyset$
for $\boldsymbol{z} \in Z_{\text{gen}}$ **do**
  $\hat{\boldsymbol{x}} \sim p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$
  $\mathcal{X}_{\text{gen}} \leftarrow \mathcal{X}_{\text{gen}} \cup \{\hat{\boldsymbol{x}}\}$
end for

return $\mathcal{X}_{\text{gen}}$

Table 8 summarizes the hyperparameters for the Riemannian metric $\mathbf{G}(\boldsymbol{z})$ construction and the NUTS sampling used for MarKBO. Following [51], the top percentage is set to $p = 5\%$, so the number of centroids is $K = \lceil pN/100 \rceil$, yielding $K = 5$ in small-budget settings ($N = 100$) and $K = 500$ in large-budget settings ($N = 10{,}000$). Accordingly, the number of candidates is set to $n_{\mathrm{cand}} = 500$ for small-budget and $n_{\mathrm{cand}} = 5{,}000$ for large-budget settings. Each MCMC chain is initialized from the latent mean of a centroid randomly drawn from the K selected centroids [70] with $n_{\mathrm{chains}} = 10$ parallel chains for small-budget and $n_{\mathrm{chains}} = 50$ for large-budget settings. The number of parallel MCMC chains follows the practice of [21]. The metric regularization coefficient $\lambda = 10^{-2}$ and the norm-decay rate $\tau = 10^{-6}$ follow the default settings recommended in [70] and its implementation. The target acceptance probability $p_{\mathrm{accept}} = 0.8$ and the maximum tree depth $D_{\max} = 10$ follow the default configuration of the NUTS implemented in Pyro [71].

Table 8: Hyperparameters of the latent geometry-aware sampling strategy. Values are reported as small-budget / large-budget where they differ.

Notation	**Description**	**Value**
Riemannian metric (Eq. (7))		
p	Top percentage for centroid selection	5%
K	Number of centroids ($\lceil pN/100 \rceil$)	5 / 500
λ	Metric regularization coefficient	10^{-2}
τ	Norm-decay rate	10^{-6}
NUTS sampler		
n_{chains}	Number of parallel MCMC chains	10 / 50
n_{warm}	Warmup steps per chain	100
n_{cand}	Total candidate samples drawn	500 / 5,000
p_{accept}	Target acceptance probability	0.8
$D_{\max}$	Maximum NUTS tree depth	10

Algorithm 3 presents MarKBO, which combines the marginalized kernel (Sec. 4.1) with the latent geometry-aware sampling (Sec. 4.2) of Algorithm 2. At each iteration the algorithm selects training data for the GP by taking the top-K observations ranked by objective value. It then fits a GP surrogate equipped with the marginalized kernel. A candidate set is generated via Algorithm 2 by drawing n_{cand} latent points from the Riemannian uniform distribution via NUTS and decoding them into the input space. The decoded candidates are re-encoded to obtain their distributional representations $(\boldsymbol{\mu}, \boldsymbol{\sigma})$. Finally B query points are selected through TS following [87] by drawing B independent posterior function samples and choosing the candidate that maximizes each sample. The selected inputs are evaluated by the black-box oracle f and the dataset is augmented for the next iteration.

Algorithm 3 Latent Space Bayesian Optimization with Marginalized Kernel (MarKBO)

Input: Structured domain $\mathcal{X}$, black-box objective $f : \mathcal{X} \to \mathbb{R}$, pre-trained VAE encoder $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and decoder $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$, marginalized kernel k_{marg} (Eq. (4)), initial dataset size n_0, batch size B, number of iterations T, GP training set size K, number of candidates n_{cand}, top percentage for centroids p, number of anchors n_{anchor}

Output: Best observed input $\boldsymbol{x}^*$ and value y^*

Sample $\{\boldsymbol{x}_i\}_{i=1}^{n_0}$ from $\mathcal{X}$ and evaluate $y_i \leftarrow f(\boldsymbol{x}_i)$
for $i = 1, \dots, n_0$ **do**
    $\boldsymbol{\mu}_i,\ \boldsymbol{\sigma}_i \leftarrow q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_i)$
end for
$\mathcal{D}_0 \leftarrow \{(\boldsymbol{x}_i,\ y_i,\ \boldsymbol{\mu}_i,\ \boldsymbol{\sigma}_i)\}_{i=1}^{n_0}$
$(\boldsymbol{x}^*, y^*) \leftarrow \arg\max_{(\boldsymbol{x}, y) \in \mathcal{D}_0} y$
Fit initial GP surrogate $\mathcal{M}$ with kernel k_{marg} on $\mathcal{D}_0$

for $t = 1, \dots, T$ **do**
    $\mathcal{T}_t \leftarrow \mathrm{TopK}(\mathcal{D}_{t-1},\ K)\ \cup\ \mathrm{Recent}(\mathcal{D}_{t-1},\ B)$
    Refit GP hyperparameters on $\mathcal{T}_t$ by maximizing $\log p(\boldsymbol{y} \mid \boldsymbol{M}, \boldsymbol{\Sigma}, \boldsymbol{\psi})$
    Generate candidate set $\mathcal{X}_{\mathrm{gen}} = \{\hat{\boldsymbol{x}}_j\}_{j=1}^{n_{\mathrm{cand}}}$ via Algorithm 2
    for $\hat{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{gen}}$ **do**
        $\boldsymbol{\mu}_{\hat{\boldsymbol{x}}},\ \boldsymbol{\sigma}_{\hat{\boldsymbol{x}}} \leftarrow q_\phi(\boldsymbol{z} \mid \hat{\boldsymbol{x}})$
    end for
    for $b = 1, \dots, B$ **do**
        Sample $\tilde{f}_b \sim p(f \mid \mathcal{T}_t)$
        $\boldsymbol{x}_{t,b} \leftarrow \arg\max_{\hat{\boldsymbol{x}} \in \mathcal{X}_{\mathrm{gen}}} \tilde{f}_b\big([\boldsymbol{\mu}_{\hat{\boldsymbol{x}}},\ \boldsymbol{\sigma}_{\hat{\boldsymbol{x}}}]\big)$
    end for
    for $b = 1, \dots, B$ **do**
        $y_{t,b} \leftarrow f(\boldsymbol{x}_{t,b})$
        $\boldsymbol{\mu}_{t,b},\ \boldsymbol{\sigma}_{t,b} \leftarrow q_\phi(\boldsymbol{z} \mid \boldsymbol{x}_{t,b})$
    end for
    $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_{t,b},\ y_{t,b},\ \boldsymbol{\mu}_{t,b},\ \boldsymbol{\sigma}_{t,b})\}_{b=1}^{B}$
    if $y_{t,b^*} > y^*$ **then**
        $\boldsymbol{x}^* \leftarrow \hat{\boldsymbol{x}}_{t,b^*},\ y^* \leftarrow y_{t,b^*}$
    end if
end for

return $\boldsymbol{x}^*,\ y^*$

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: The abstract and introduction clearly state our main contributions.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: We briefly discuss the limitations of the proposed algorithm in Section 6. A more detailed discussion is provided in Appendix A, covering the reliance on deep generative models and the computational overhead.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: All assumptions are explicitly stated (Assumptions 1 and 2) and the complete proof of Proposition 1 is provided in Appendix B.1.

Guidelines:

- The answer [N/A] 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 the algorithmic description in Section 4 and Appendix F. All experimental settings, hyperparameters, baseline configurations, and training procedures are detailed in Section 5 and Appendix E.

Guidelines:

- The answer [N/A] 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]

   Justification: Our code will be available at https://github.com/scut-eipi/markbo.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the experimental setup in Section 5, including evaluation budgets, batch sizes, and initial data specifications. Full details on hyperparameters, model architectures and baseline configurations are provided in Appendix E.

   Guidelines:

   - The answer [N/A] 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: All experiments in Section 5 are conducted with 10 random seeds. The shaded regions in the figures represent the standard deviation across these runs, and the values reported in the tables are the mean $\pm$ standard deviation over the 10 independent runs.

   Guidelines:

   - The answer [N/A] 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: We report the hardware specifications (CPU, GPU, memory) in Section E.5 and the compute time for experiments in Section A.

   Guidelines:

   - The answer [N/A] 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 reviewed the NeurIPS Code of Ethics and confirm that the research conducted in this paper conforms with it.

   Guidelines:

   - The answer [N/A] 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: [Yes]

    Justification: We briefly discuss broader impacts in Section 6. We highlight the potential to reduce reliance on expensive experimental evaluations as a positive societal impact. We also acknowledge possible misuse risks when the optimization framework is applied to objectives that conflict with safety considerations.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper proposes a Bayesian optimization method evaluated on standard benchmarks. It does not pose a high risk for misuse.

    Guidelines:

    - The answer [N/A] 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: We properly cite all original assets used in this work and include their licenses. Details are provided in Appendix E.6.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] 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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.