Combining Latent Space and Structured Kernels for Bayesian Optimization over Combinatorial Spaces

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider the problem of optimizing combinatorial spaces (e.g., sequences, trees, and graphs) using expensive black-box function evaluations. For example, optimizing molecules for drug design using physical lab experiments. Bayesian optimization (BO) is an efficient framework for solving such problems by intelligently selecting the inputs with high utility guided by a learned surrogate model. A recent BO approach for combinatorial spaces is through a reduction to BO over continuous spaces via learning a latent representation from a database of unsupervised structures. The selected input from the continuous space is decoded into a discrete structure for performing function evaluation. However, this BO procedure doesn't take into account the corresponding decoded structures of points in the latent space resulting in a disconnect between the actual function evaluation and the BO process. To overcome this drawback, this paper proposes a principled approach referred as LADDER (**La**tent Space BO guided by **D**eco**de**d St**r**uctures). The key idea is to integrate the structural information from decoded structures with the learned latent space representation for better surrogate modeling. Our experiments on real-world benchmarks show that LADDER significantly improves over baseline methods.

1 Introduction

A huge range of science and engineering applications involve optimizing combinatorial spaces (e.g., sequences, trees, graphs) using *expensive* black-box function evaluations. For example, in drug design application, each candidate structure is a molecule and evaluation involves performing an expensive physical lab experiment. Bayesian optimization (BO) [44] is an effective framework for optimizing expensive black-box functions. The key idea is to learn a *cheap-to-evaluate* surrogate statistical model, e.g., Gaussian process (GP), from past function evaluations and employ it to select inputs with high-utility for evaluation. BO for combinatorial spaces is a relatively less-studied research topic with many challenges in the *small-data* setting (number of function evaluations is small).

A recent approach to solve some of these challenges is through a reduction to BO over continuous spaces, which we refer as *BO over latent space* [15, 52]. This method relies on two ideas. First, we learn a latent space representation from a database of unsupervised structures (i.e., function evaluations are not available) using a encoder-decoder style deep generative model [25, 21]. Second, we build a statistical model over the learned latent space to perform BO. Each selected input from the latent space is decoded into a structured object for performing function evaluation. BO over latent space has shown good success when the number of function evaluations is large [15, 52, 11, 22]. However, this approach may not be effective in small-data settings for the following reasons. **1)** While searching the latent space, it does not take into account the structured objects corresponding to the decoding of points in the latent space. This results in a fundamental disconnect between the actual function evaluation and the BO process. **2)** The learned statistical model may not generalize well

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

beyond the training instances from the latent space. Indeed, our experiments on real-world problems demonstrate the ineffectiveness of this Naïve surrogate model over the latent space.

In this paper, we propose a novel approach referred as LADDER (**La**tent Space BO guided by **D**eco**de**d St**r**uctures) to overcome the above drawbacks for the small-data setting. Intuitively, LADDER provides a bridge between the combinatorial space and the learned latent space to improve surrogate modeling by combining the best of both representations. We provide a principled Gaussian process based method to integrate the latent space representation with the structural information from decoded structures using a novel *structure-coupled kernel*. The key insight is to extend a kernel over the latent space (with its hyper-parameters estimated on the evaluated points) to non-evaluated points in the space by utilizing the features exploited in structured kernels defined over combinatorial spaces (e.g., string kernels and graph kernels). LADDER has multiple advantages. First, we can leverage a large body of research on kernels over structured data. Second, allows the use of deep generative models (DGMs) to learn latent space as a plug-and-play technology. This means advances in DGMs will directly improve the effectiveness of LADDER. Our experiments on real-world benchmarks show that LADDER performs better or similar to state-of-the-art methods, and significantly better than the Naïve latent space BO approach in our problem setting. We also empirically demonstrate that superiority of LADDER's performance is due to better surrogate model resulting from the proposed method to combine representations.

Contributions. The key contribution of this paper is the development and evaluation of the LADDER approach to perform BO over combinatorial spaces in the *small-data setting*. Specific list includes:

- Identifying the key reasons for the ineffectiveness of the Naïve latent space BO method in the small-data setting and providing empirical evidence on real-world problems.
- Development of a principled Gaussian process approach for improved surrogate modeling by combining the structural information from decoded structures with the learned latent space representation using a novel structure-coupled kernel.
- Experiments on real-world benchmarks to show the efficacy of LADDER over prior methods in our problem setting. The code and data are available on a GitHub repository and will be made public. Our anonymized code is in Appendix for review purposes.

2 Problem Setup and Background

Let $\mathcal{X}$ be a space of combinatorial structures (e.g., sequences, trees, and graphs). We assume the availability of a black-box objective function $f : \mathcal{X} \mapsto \mathbb{R}$ defined over the combinatorial space $\mathcal{X}$. Evaluating each candidate structure $\mathbf{x} \in \mathcal{X}$ using function f (also called an experiment) is *expensive* in terms of the resources consumed and produces an output $y = f(\mathbf{x})$. For example, in the drug design application, each $\mathbf{x} \in \mathcal{X}$ is a molecule, and $f(\mathbf{x})$ corresponds to running a physical lab experiment. Our overall goal is to find a structure $\mathbf{x} \in \mathcal{X}$ that approximately optimizes f by minimizing the number of experiments and observing their outcomes.

We are also provided with a database of *unsupervised* structures $\mathcal{X}_u \subset \mathcal{X}$. Unsupervised means that we do not know the function evaluations $f(x)$ for structures in $\mathcal{X}_u$. This assumption is satisfied by many scientific applications including chemical design and material design. We assume the availability of a *latent space* $\mathcal{Z}$ learned from unsupervised structures $\mathcal{X}_u$ using a encoder-decoder style deep generative model, e.g., variational autoencoders (VAEs) for structured data such as junction tree VAE [21] and grammer VAE [25]. Formally, the encoder denoted by Υ embeds a given combinatorial structure $\mathbf{x} \in \mathcal{X}$ into a point in the latent space $\mathbf{z} \in \mathbb{R}^d = \Upsilon(x)$ where d is the number of dimensions of the latent space $\mathcal{Z}$ and the decoder denoted by Φ converts a given point from latent space $\mathbf{z}' \in \mathcal{Z}$ into a structured object $\mathbf{x}' \in \mathcal{X} = \Phi(\mathbf{z}')$. Encoder Υ and decoder Φ are typically realized by neural networks.

Bayesian Optimization. To solve optimization problems using black-box evaluations of expensive objective functions, Bayesian optimization (BO) is an efficient framework. BO algorithms select inputs with high utility guided by a learned surrogate statistical model from past observations. The three key elements of BO are as follows: **1)** Statistical model of the black-box function $f(\mathbf{x})$, e.g., Gaussian process; **2)** Acquisition function to score the utility of evaluating candidate inputs using the statistical model, e.g., expected improvement; and **3)** Acquisition function optimizer to select the input with maximum utility. In each BO iteration, we select one input for function evaluation and update the statistical model based on the new training example. BO has shown great success for continuous spaces, but BO over combinatorial spaces is a relatively less-studied problem space.

Naïve Latent Space BO. This approach reduces the problem of BO over combinatorial space $\mathcal{X}$ to BO over the latent space $\mathcal{Z}$, which is continuous. We construct statistical model $\mathcal{M}_{\mathcal{Z}}$ over the latent space $\mathcal{Z}$ using training points selected from $\mathcal{Z}$ for evaluation. Each selected input $\mathbf{z} \in \mathcal{Z}$ is decoded into a structured object using the decoder Φ. We perform function evaluation using the decoded structure $\Phi(\mathbf{z})$ and the statistical model $\mathcal{M}_{\mathcal{Z}}$ is updated using the new training example ($\mathbf{z}$, $f(\Phi(\mathbf{z}))$). This reduction approach allows us to leverage prior research on BO for continuous spaces and has shown success when the the number of function evaluations is large [15, 52, 11, 22].. In this paper, we study the *small data* setting (number of function evaluations is small) by addressing the drawbacks of this Naïve latent space BO approach in a principled manner.

3 LADDER: Latent Space BO guided by Decoded Structures

In this section, we first discuss the challenges with the Naïve latent space BO approach. Next, we describe our proposed LADDER approach with a focus on the novel surrogate model by combining kernels over structured data and latent space representation, which is our key technical contribution.

3.1 Challenges with the Naïve latent space BO approach

As mentioned above, the Naïve latent space BO (LSBO) approach builds a statistical model over the latent space using kernels for continuous spaces (e.g., Radial Basis Function or Squared Exponential kernel) and performs acquisition function optimization in the latent space using optimizers for continuous spaces (e.g., gradient-based methods) to select point $\mathbf{z} \in \mathcal{Z}$ for evaluation. However, the expensive objective function $f(\mathbf{x})$ is defined over the space of combinatorial structures $\mathcal{X}$ and *not* the latent space $\mathcal{Z}$. Therefore, we need to decode this point $\mathbf{z}$ using the decoder Φ to get the corresponding combinatorial structure $\mathbf{x}$=$\Phi(z)$ for function evaluation $f(\mathbf{x})$. All the existing work on BO over latent space do not account for this decoding process. As a consequence, we need to deal with two inter-related challenges, which are especially significant for small-data settings, where we need to rely on a small number of function evaluations.

- **Challenge #1:** Ignoring the decoding process in this iterative loop results in a fundamental disconnect between the modeling process and the objective function. Therefore, we are not able to leverage the potentially useful inductive bias and rich structural information available in decoded structures.
- **Challenge #2:** The statistical model itself might not be expressive enough to generalize well beyond the training examples from the latent space (set of points in the latent space and their corresponding function evaluations). This is especially true in the small-data setting for latent spaces learned using DGMs in real-world scientific applications, where the number of dimensions can be large when compared to the standard BO setup.

Indeed, we provide strong empirical evidence to demonstrate these challenges and our key hypothesis in Figure 2. We show that by incorporating the rich structural information from the decoded output, we can address both these challenges to improve the overall BO performance.

3.2 Overview of LADDER algorithm and key advantages

LADDER is an instantiation of the latent space BO framework that employs a novel surrogate statistical model to address the two challenges with the Naïve method. The surrogate model is a Gaussian process that combines the complementary strengths of the latent space representation with the rich structural information from decoded outputs using structured kernels (e.g., string kernels and graph kernels). The key idea is to define a *structure-coupled kernel* that extends the continuous kernel on the evaluated points in the latent space to unknown points using the rich information from structured kernels. We employ expected improvement (EI) as the acquisition function. For optimizing the acquisition function to select high utility inputs from the latent space, we employ evolutionary search due to its recent successes [27] including policy search in high-dimensional spaces [40].

Figure 1 shows a high-level illustration of LADDER and Algorithm 1 provides the complete pseudo-code. We use a small set of initial training data in the form of points in the latent space and their corresponding function evaluations to bootstrap the GP based surrogate model using the structure-coupled kernel. In each iteration t, we optimize the acquisition function to select a point $\mathbf{z}_t$ from the

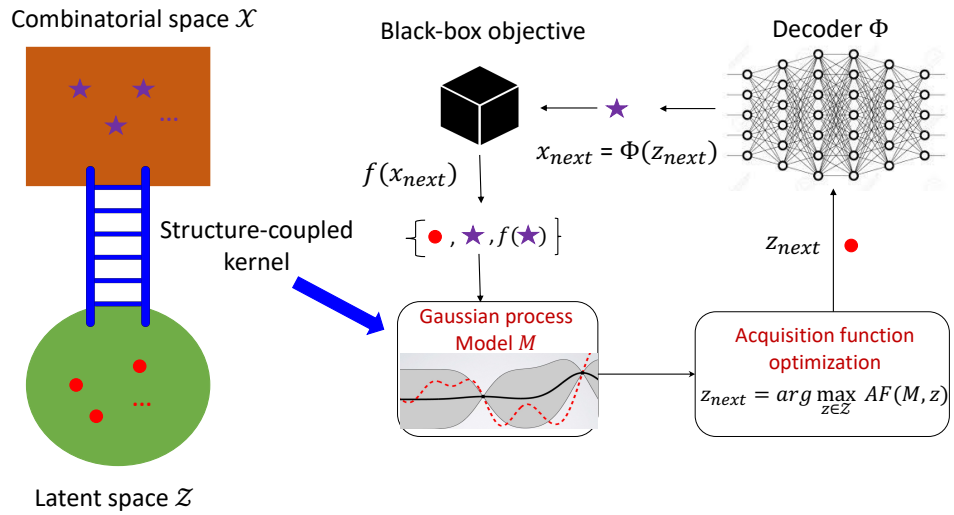


Figure 1: High-level conceptual illustration of our proposed LADDER approach, which acts as a "ladder" in connecting the rich structural information of each structure in the combinatorial space with its corresponding latent space representation. Structure-coupled kernel is the key element that enables this connection to build an effective Gaussian process based surrogate model.

latent space $\mathcal{Z}$ for evaluation. The corresponding decoded structure $\mathbf{x}_t$=$\Phi(\mathbf{z}_t)$ is evaluated to measure the outcome $f(\mathbf{x}_t)$. The GP model with structure-coupled kernel is updated using the new 3-tuple training example $\{\mathbf{z}_t, \mathbf{x}_t, f(\mathbf{x}_t)\}$. We repeat these sequence of steps until covergence or maximum query budget and then return the best uncovered combinatorial structure $\hat{\mathbf{x}} \in \mathcal{X}$ as the output.

Advantages of LADDER. Some of the key advantages of our proposed approach are listed below.

- We are allowed to employ any existing trained deep generative model for structured data in a *plug-and-play* manner with LADDER. Therefore, any advances in the latent-space generative modeling technology will directly improve the overall BO performance.
- LADDER is a generic approach that is applicable to any combinatorial space of structures (e.g., sequences, trees, graphs, sets, and permutations). This method just requires an appropriate structured kernel over the given combinatorial space and is agnostic to the particular choice of kernel. This allows us to leverage a large body of research on kernels over structured data (e.g., string kernels and graph kernels) and hand-designed kernels based on domain knowledge for specific applications. For example, in our experiments, we employ sub-sequence string kernel and fingerprint kernel to concretely instantiate the LADDER approach for strings and molecules respectively to demonstrate its flexibility.
- Combines the complementary strengths of latent space representations and structured kernels in a principled manner to create highly-effective surrogate statistical models.

3.3 Novel surrogate statistical model via structure-coupled kernel

We consider *Gaussian process (GP)* [37] as the surrogate model of the expensive black-box objective function $f(\mathbf{x} \in \mathcal{X})$. GPs are known to have excellent statistical properties including principled uncertainty quantification, which is critical for the effectiveness of Bayesian optimization algorithms. A GP model defines a prior distribution on functions which is entirely characterized by a kernel defined over a pair of inputs. Most of the existing work on latent space BO employs a standard continuous space kernel represented by $l(\mathbf{z_i} \in \mathcal{Z}, \mathbf{z_j} \in \mathcal{Z})$, e.g., RBF and Matern kernels, over points in the latent space to create the GP model. However, this surrogate model is highly-ineffective for small data settings, especially when the number of dimensions of latent space is large, which is the common case for deep generative models for scientific applications.

Algorithm 1 Latent Space Bayesian Optimization guided by Decoded Structures (LADDER)

Input: Objective function $f(\mathbf{x})$, Encoder (Υ) - Decoder (Φ) style model for latent space $\mathcal{Z}$, Kernel for latent space $l(\mathbf{z_i} \in \mathcal{Z}, \mathbf{z_j} \in \mathcal{Z})$, Kernel for combinatorial space $k(\mathbf{x_i} \in \mathcal{X}, \mathbf{x_j} \in \mathcal{X})$
Initialize dataset $\mathcal{D}_0$ by evaluating few random points: $\mathcal{D}_0 \leftarrow \{\mathbf{Z}_0, \mathbf{X}_0, f(\mathbf{X}_0)\}$; $t \leftarrow 0$
// slight abuse of notation here since $\mathbf{Z}_0$ is the set of initial points from latent space $\mathcal{Z}$ and $\mathbf{X}_0$ is the set of corresponding decoded structures from $\mathcal{X}$ with function evaluations $f(\mathbf{X}_0)$
repeat
  Learn Gaussian process model on the dataset $\mathcal{D}_t$ with the proposed kernel in Equation 4
  Optimize acquisition function over the latent space $\mathcal{Z}$ to find the next point $\mathbf{z}_t$ for evaluation
  Compute the decoded structure $\mathbf{x}_t$ for point $\mathbf{z}_t$ using decoder Φ
  Evaluate the combinatorial structure $\mathbf{x}_t$ to get $f(\mathbf{x}_t)$
  Add new training triple: $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{\mathbf{z}_t, \mathbf{x}_t, f(\mathbf{x}_t)\}$; increment the iteration $t \leftarrow t+1$
until convergence or maximum iterations
Output: best uncovered structure $\hat{\mathbf{x}}$ and the corresponding function value $f(\hat{\mathbf{x}})$

Structure-coupled kernel. We propose to utilize the rich structural information that is available from the decoded combinatorial structure $\mathbf{x}$ corresponding to each point $\mathbf{z}$ from the latent space $\mathcal{Z}$. The key idea behind our approach is to integrate the structure information from the decoded outputs with the learned representation of inputs from the latent space to achieve better surrogate modeling performance. To include this structure information in a principled manner witin a GP model, we leverage the Generalized Nystrom extension idea [42, 53] to *extrapolate* the eigenfunctions of the kernel matrix over latent space ($\mathbf{L} = \{L_{ij} = l(\mathbf{z_i}, \mathbf{z_j}) | \mathbf{z_i}, \mathbf{z_j} \in Z\}$) with basis functions from a kernel k defined over the decoded combinatorial structures.

Without loss of generality, let m be the number of evaluated inputs from the latent-space, which are denoted as $\mathbf{Z} = \{\mathbf{z_1}, \mathbf{z_2}, \cdots, \mathbf{z_m}\}$. For example, these are the points accumulated after m iterations of the latent space BO approach. Let the corresponding set of decoded structures $\mathbf{X} = \{\mathbf{x_1} = \Phi(\mathbf{z_1}), \mathbf{x_2} = \Phi(\mathbf{z_2}), \cdots, \mathbf{x_m} = \Phi(\mathbf{z_m})\}$ with their function evaluations $\{f(\mathbf{x_1}), f(\mathbf{x_2}), \cdots, f(\mathbf{x_m})\}$. Given kernels l and k with $\mathbf{L}$ and $\mathbf{K}$ representing their corresponding kernel matrices, Generalized Nystrom extension generates an m-dimensional feature vector $\xi(\mathbf{z})$ for a point $\mathbf{z}$ in the latent space. The ith component of $\xi(\mathbf{z})$ is given as follows:

$$\xi_i(\mathbf{z}) = \mathbf{k_z}^T \mathbf{K}^{-1} v_i \quad i \in \{1, 2, \cdots, m\} \tag{1}$$

where $\mathbf{k_z}$ is an m-dimensional vector evaluated between $\mathbf{x}$ (decoded output of latent space input $\mathbf{z}$) and other combinatorial structures from $\mathbf{X}$:

$$\mathbf{k_z} = [k(\Phi(\mathbf{z}), \Phi(\mathbf{z_1})), \cdots, k(\Phi(\mathbf{z}), \Phi(\mathbf{z_m})) = [k(\mathbf{x}, \mathbf{x_1}), \cdots, k(\mathbf{x}, \mathbf{x_m})]] \tag{2}$$

$\mathbf{K}$ is an $m \times m$ kernel matrix for combinatorial structures in the set $\mathbf{X}$, i.e., $K_{ij} = k(\Phi(\mathbf{z_i}), \Phi(\mathbf{z_j})) = k(\mathbf{x_i}, \mathbf{x_j})$, and v_i is the eigenvector of the kernel matrix $\mathbf{L}$ defined over the latent space inputs in the set $\mathbf{Z}$. The reader should note that the term $k(\Phi(\mathbf{z_i}), \Phi(\mathbf{z_j}))$ means that the structured kernel is applied to the decoded structures $\mathbf{x_i} = \Phi(\mathbf{z_i})$ and $\mathbf{x_j} = \Phi(\mathbf{z_j})$ of latent space inputs $\mathbf{z_i}$ and $\mathbf{z_j}$ respectively. This is a key term in the above expression because it integrates each input from latent space with its decoded structure in the new feature map $\xi(\mathbf{z})$. For any two inputs $\mathbf{z}$ and $\mathbf{z}'$ in the latent space, the resulting structure-coupled kernel denoted as $c(\mathbf{z}, \mathbf{z}'))$ is defined as the dot product of their corresponding feature vectors $\xi(\mathbf{z})$ and $\xi(\mathbf{z}')$:

$$c(\mathbf{z}, \mathbf{z}') = \xi(\mathbf{z})^{\mathbf{T}} \xi(\mathbf{z}') \tag{3}$$

$$c(\mathbf{z}, \mathbf{z}') = \mathbf{k_z}^T \mathbf{K}^{-1} \mathbf{L} \mathbf{K}^{-1} \mathbf{k_{z'}} \tag{4}$$

Intuitively, by this construction, we are extending the kernel over the latent space l on the evaluated points $\mathbf{Z}$ to non-evaluated points in the space by utilizing the rich structural features exploited in kernels (k) commonly defined over combinatorial spaces[1]. More importantly, this general construction of structure-coupled kernel allows us to leverage extensive research on kernel methods for highly-structured data, which tries to exploit structural features of combinatorial objects. For example,

[1] In fact, $\mathbf{k_z}$ is the well-known empirical kernel feature map for $\Phi(\mathbf{z})$ [41]. Technically, we need to orthonormalize this feature map but core concept remains the same.

string kernels [26] count the number of common sub-strings in string inputs, fingerprint kernels [36] capture neighborhood-aggregated properties of molecular inputs, and graph features such as number of random walks and shortest paths are utilized by kernels defined on graph inputs [4].

4 Experiments and Results

In this section, we empirically evaluate the effectiveness of the proposed LADDER approach on real-world benchmarks, and perform comparison with baseline methods.

4.1 Real-world benchmarks

We employ two widely used real-world benchmarks for combinatorial Bayesian optimization.

Arithmetic expressions optimization. In this benchmark, the goal is to search in the space of uni-variate arithmetic expressions (generated from a given grammar) to find the best expression that fits a given target expression [25]. As described in [25], the latent space model is trained on 100K randomly generated expressions from the following grammar:

```
S → S '+' T | S '*' T | S '/' T | T
T → '(' S ')' | 'sin(' S ')' | 'exp(' S ')'
T → 'v' | '1' | '2' | '3'
```

We follow the same setup as discussed in the state-of-the-art paper for this benchmark [52]. We consider the log mean-squared error between an expression $\mathbf{x}$ and the target expression $\mathbf{x}^* = 1/3 \cdot v \cdot \sin(v^2)$ (computed over 1000 evenly-spaced values of v in the interval $[-10, 10]$) as the objective function which should be minimized.

Chemical design optimization. This benchmark considers finding molecules with best drug-like properties [25] and is similar in prototype for many scientific applications. Specifically, the goal is to maximize the water-octanol partition coefficient (logP) over the space of molecules. The latent space model is trained on the Zinc molecule dataset of 250K molecules. For consistency purposes, all our results are shown as minimization obtained by taking a negation of the logP objective.

4.2 Experimental setup

Configuration of algorithms. The Bayesian optimization part of the source code is written using the popular GpyTorch [12] and BoTorch [2] libraries for all BO methods including LADDER. We employ ARD (automatic relevance determination) Matern kernel for the latent space inputs in all our experiments. Matern kernel is commonly advocated as a better choice than RBF kernel for BO algorithms since the sample functions from the latter are impractically smooth [48]. Hyperparameters of Gaussian process models are fitted by marginal likelihood maximization after every BO iteration. We employed Junction tree VAE [21] and Grammar VAE [25] as the latent-space model for chemical design and arithmetic expression optimization benchmarks respectively. Both pretrained encoder-decoder models are taken from the source code provided by the authors' of [52] [2]. We employed expected improvement (EI) as the acquisition function for all the BO methods. All experiments are performed on a machine with the following configuration: Intel(R) Core(TM) i9-7960X CPU @ 2.80GHz with 128 GB RAM.

LADDER instantiations. In addition to the encoder-decoder style latent space model, which is same as the Naïve latent space BO (LSBO), LADDER also requires an appropriate structured kernel for the given combinatorial space. To demonstrate the generality of our proposed approach, we instantiate LADDER with two different kernels for our two optimization benchmarks. We employed sub-sequence string kernel for the arithmetic expressions task and fingerprints kernel for the chemical design task. We briefly describe both kernels below.

- *Sub-sequence string kernel.* This kernel captures the similarity between two strings by counting the number of matching substrings, where the substrings can be non-contiguous [26, 7]. Following the notation in [29], given an alphabet Π, the kernel between two strings s_1 and s_2 is given as: $k(s_1, s_2) = \sum_{u \in \Pi^n} \rho(s_1)\rho(s_2)$ where $\rho(s) =$

[2] https://github.com/cambridge-mlg/weighted-retraining/

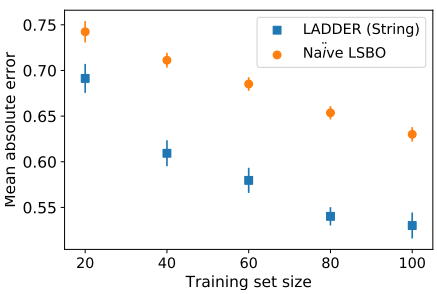
$\lambda_m^{|u|} \cdot \sum_{1<i_1<\cdots<i_{|u|}<|s|} \left(\lambda_g^{i_{|u|}-i_1} \mathbf{1}_u(s_{i_1}, \cdots, s_{i_{|u|}})\right)$ for any string s where λ_g and λ_m are gap decay and match decay hyper-parameters. Since arithmetic expressions are naturally represented as strings, we use this kernel for the arithmetic expressions task.

- *Fingerprints kernel.* There is a large body of work in the chemical informatics literature for designing structural features for molecular inputs. These hand-engineered features by domain experts are commonly known as molecular fingerprints [28]. We employ Morgan fingerprints [38] which are high-dimensional binary features to capture different substructures in a molecule while being invariant to atom relabeling. Since combinatorial structures in the chemical design task are molecules, given two molecules m_1 and m_2, we consider the dot product of their Morgan fingerprints as the structured kernel for the chemical design task. Hence, we refer to this kernel as the Fingerprints kernel.

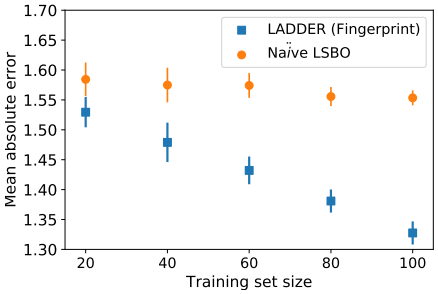
In all our results and figures, the corresponding structured kernel for LADDER is denoted in the parenthesis, e.g., LADDER (String). We employed the evolutionary search algorithm CMA-ES [17] as the acquisition function optimizer for LADDER. The parameters of CMA-ES[3] were fixed with sigma = 0.2 and a population size of 50. We found the performance of CMA-ES to be highly robust to different choice of these parameters. We ran CMA-ES from 10 different starting inputs for 10 iterations each and picked the best optimizer found. As discussed later, we consider one instance of Naïve LSBO with the same configuration of CMA-ES optimizer for fair comparison and to test our key hypothesis on surrogate modeling.

Evaluation metric. We evaluate all methods on the best objective (log MSE for the arithmetic expressions task and logP for the chemical design task) value uncovered as a function of the number of experiments (expensive function evaluations). The method that finds high-performing structures with less number of function evaluations is considered better.

We ran each method on all benchmarks for 10 different runs with the same initialization (small number of randomly selected structures and function evaluations). We plot the mean and two times the standard error for all our experimental results.



(a) Arithmetic expression optimization



(b) Chemical design optimization

Figure 2: Mean absolute error results comparing the quality of model fit for varying sizes of training sets with two models: GP model with Matern kernel (Naïve LSBO) and GP model with the proposed structure-coupled kernel (LADDER). Lower MAE values means better surrogate model.

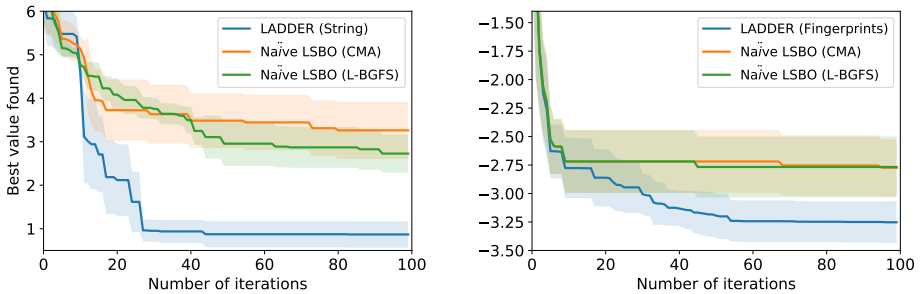
4.3 Results and discussion

Comparison of surrogate models. Recall that the key hypothesis of this paper is that surrogate model employed by the Naïve LSBO approach (GP model with Matern kernel in our case) is ineffective and GP model with our proposed structure-coupled kernel for small-data settings. To test this hypothesis, we compare the quality of the model fit for these two GP models (Naïve LSBO and LADDER) by evaluating the mean absolute error (MAE) of their predictions on a testing set. To perform this experiment, we generate 50 random training sets of different sizes and evaluate the models on 20 random testing sets. The averaged MAE results over these 1000 training and testing set pairs are shown in Figure 2. We make following observations. **1)** The standard GP model on latent space denoted by Naïve LSBO shows some improvement in MAE for the arithmetic expressions

[3] https://github.com/CMA-ES/pycma

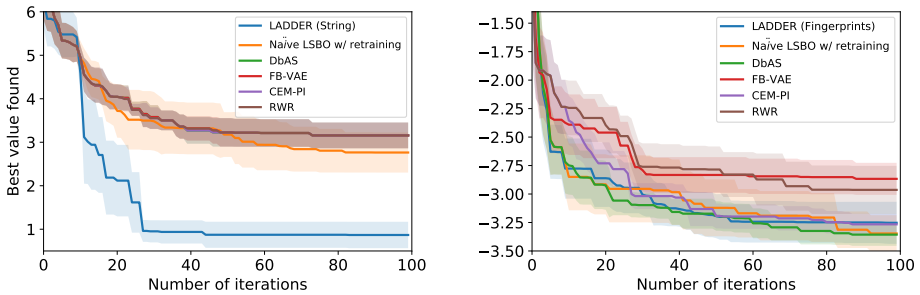
task while the improvement is minimal for the chemical design task. **2)** Our proposed surrogate model, i.e., GP with structure-coupled kernel, denoted by LADDER has significantly lower MAE than Naïve LSBO on both the benchmarks and continuously decreases as a function of the training set size. These strong results corroborate our key hypothesis and demonstrate the effectiveness of our proposed surrogate model with the structure-coupled kernel.

Naïve latent space BO vs. LADDER. A natural question based on the above results is whether improved surrogate modeling results in better overall BO performance. To answer this question, we compare the BO performance (best uncovered objective value vs. number of function evaluations or iterations) of Naïve LSBO and LADDER. We consider two choices for acquisition function optimizer within the Naïve LSBO approach: zeroth order CMA based optimizer (same as LADDER) and second-order gradient based optimizer (L-BFGS). We make the following observations from the results shown in Figure 3. **1)** LADDER consistently uncovers significantly better structures than those from the Naïve LSBO approach on both benchmarks. This is a direct consequence of the better surrogate model of LADDER since the acquisition function optimizer is kept the same for both methods (i.e., CMA). **2)** L-BFGS based acquisition function optimizer slightly improves the BO performance of Naïve LSBO but still cannot match the performance of LADDER.



(a) Arithmetic expression dataset (b) Chemical design dataset

Figure 3: Results comparing the BO performance of LADDER and Naïve latent space BO.



(a) Arithmetic expression dataset (b) Chemical design dataset

Figure 4: Results comparing the BO performance of LADDER and different state-of-the-art methods.

LADDER vs. State-of-the-art. To further analyze LADDER, we compare its performance with a set of state-of-the-art methods. We include *weighted retraining* approach which was recently proposed [52] to update the latent space model as BO algorithm progresses (Naïve LSBO w/ retraining). This method is clearly complementary to our proposed LADDER approach. We also consider design by adaptive sampling (DbAS) [5], the cross-entropy method with probability of improvement (CEM-PI) [39], the feedback VAE (FB-VAE) [16] and reward-weighted regression (RWR) [34]. We employ the publicly available implementation of these methods. We make the following observations from the results shown in Figure 4. **1)** LADDER performs significantly better on the arithmetic expressions task and similar to the best method on the chemical design task. We verify the similar performance of LADDER on the chemical design task by performing a two-sided paired Wilcoxon test at 1%

significance. The performance of LADDER and Naïve LSBO w/ retraining is statistically similar on the chemical design task (p-value = 0.0489). **2)** Retraining the latent space model helps in improving the performance of Naïve LSBO on both tasks. **3)** DbAS and CEM-PI perform similar to Naïve LSBO w/ retraining since they are special case of retraining method as described in [52]. The discrepancy in LADDER's performance on the chemical design benchmark can be attributed to the low-flexibility of fingerprint kernel which doesn't include any hyper-parameters (as opposed to string kernel) to tune it for a specific dataset.

5 Related Work

In this section, we discuss the most closely related work for the problem setting studied in this paper.

Deep generative models for structured data. Scientific domains including physics, chemistry, and molecular biology involve highly structured data [43]. To address the unique challenges of structured data, there is a growing body of work on deep generative models (DGMs). Some notable examples include grammar VAE [25], junction tree VAE [21] and syntax-directed VAE [8]. This line of work is complementary to our proposed method as advances in DGMs will automatically improve the BO performance of LADDER.

Kernels over structured data. There is a large body of research on generic kernels over structured data including strings [26, 7], trees [46], graphs [4], sets [6], and permutations [20]. LADDER allows us to leverage this prior research to perform sample-efficient BO over structured spaces.

BO over latent space. The use of DGMs to learn a latent space and to perform BO in this continuous space was first started by Gomez-Bombarelli et al., for chemical design [15]. This led to a lot of followup work motivated by applications from multiple domains [11, 22, 21]. A recent work by Tripp et al., [52] noted that this decoupled approach of training DGM and BO process can make the overall optimization problem hard; and proposed an approach referred as weighted retraining. The key idea is to periodically retrain the DGM by giving more weight to inputs with high function values. Our LADDER approach and weighted retraining have complementary strengths. For small-data settings, LADDER is much more effective than weighted retraining as demonstrated by our experiments.

BO over combinatorial space. We can categorize these methods based on the choice of surrogate model and search strategies for acquisition function optimization. Linear models [3], Gaussian processes with kernels over discrete structures [31, 14, 24], and random forest [19] are considered for surrogate modeling. Search strategies include mathematical optimization [3], heuristic search methods [19, 31], and combination thereof [10, 50, 30]. LADDER is complementary to this line of work. First, it allows to combine the complementary strengths of two pieces of knowledge, namely, generic/domain-specific kernels over structured data and data-driven representations learned using DGMs, for improved surrogate modeling. Second, it is well known in the AI search literature that effective search strategies varies from one problem to another. For example, a recent work on BO over string spaces [29], showed that we need to specialize genetic algorithms for different problems. In contrast, we can use generic search strategies for continuous spaces with LADDER.

Reinforcement learning Reinforcement learning (RL) formulations are also studied to solve this optimization problem [54, 56, 32, 47, 35, 1, 45]. The key idea is to encode the problem as a Markov decision process in terms of states, actions, and reward function to be able to apply RL algorithms to learn a policy to guide the selection of structures for evaluation. RL is very effective for simulated environments where collection of large amount of data to improve the policy is possible. Unfortunately, this requirement is impractical when we need to perform expensive experiments to collect new training examples.

6 Summary and Future Work

We introduced a sample-efficient Bayesian optimization (BO) approach for combinatorial spaces called LADDER. The key idea behind LADDER is a novel Gaussian process based surrogate model that combines the complementary strengths of latent space representation with rich information about decoded outputs using structured kernels. We showed that the BO performance of LADDER is better or similar than state-of-the-art methods and significantly better than the Naïve latent space BO method. One potential limitation of LADDER might be selecting the right choice of structured kernel for a given task. Since LADDER's key contribution is in the surrogate model part of the BO procedure, it provides the flexibility to use any acquisition function, which opens up an avenue for various type of extensions including multi-objective [9, 33, 49], multi-fidelity [51, 23, 55], constrained BO [18, 13].

References

[1] Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*, 2019.

[2] 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, 2020.

[3] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. In *Proceedings of the 35th International Conference on Machine Learning*, pages 462–471, 2018.

[4] Karsten Borgwardt, Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: state-of-the-art and future challenges. *arXiv preprint arXiv:2011.03854*, 2020.

[5] David H Brookes and Jennifer Listgarten. Design by adaptive sampling. *arXiv preprint arXiv:1810.03714*, 2018.

[6] Poompol Buathong, David Ginsbourger, and Tipaluck Krityakierne. Kernels over sets of finite sets using RKHS embeddings, with application to bayesian (combinatorial) optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2731–2741. PMLR, 2020.

[7] Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. Word sequence kernels. *The Journal of Machine Learning Research*, 3:1059–1082, 2003.

[8] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018.

[9] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. *arXiv preprint arXiv:2006.05078*, 2020.

[10] Aryan Deshwal, Syrine Belakaria, Janardhan Rao Doppa, and Alan Fern. Optimizing discrete spaces via expensive evaluations: a learning to search framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3773–3780, 2020.

[11] Stephan Eissman, Daniel Levy, Rui Shu, Stefan Bartzsch, and Stefano Ermon. Bayesian optimization and attribute adjustment. In *Proc. 34th Conference on Uncertainty in Artificial Intelligence*, 2018.

[12] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[13] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *International Conference on Machine Learning*, volume 2014, pages 937–945, 2014.

[14] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380:20–35, 2020.

[15] 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.

[16] Anvita Gupta and James Zou. Feedback GAN for DNA optimizes protein functions. *Nature Machine Intelligence*, 1(2):105–111, 2019.

[17] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.

[18] José Miguel Hernández-Lobato, Michael A. Gelbart, Ryan P. Adams, Matthew W. Hoffman, and Zoubin Ghahramani. A general framework for constrained bayesian optimization using information-based search. *Journal of Machine Learning Research*, 17(1):5549–5601, 2016.

[19] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on Learning and Intelligent Optimization*, pages 507–523, 2011.

[20] Yunlong Jiao and Jean-Philippe Vert. The kendall and mallows kernels for permutations. In *International Conference on Machine Learning*, pages 1935–1944. PMLR, 2015.

[21] 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.

[22] Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization. In *International Conference on Machine Learning*, pages 3183–3191. PMLR, 2019.

[23] Kirthevasan Kandasamy, Gautam Dasarathy, Junier Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process optimisation with multi-fidelity evaluations. In *Proceedings of the 30th/International Conference on Advances in Neural Information Processing Systems (NIPS'30)*, 2016.

[24] Jungtaek Kim, Michael McCourt, Tackgeun You, Saehoon Kim, and Seungjin Choi. Bayesian optimization over sets. *arXiv preprint arXiv:1905.09780*, 2019.

[25] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR, 2017.

[26] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444, 2002.

[27] Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Bi-population CMA-ES agorithms with surrogate models and line searches. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 1177–1184, 2013.

[28] Andrea Mauri, Viviana Consonni, Manuela Pavan, and Roberto Todeschini. Dragon software: an easy approach to molecular descriptor calculations. *Match*, 56(2):237–248, 2006.

[29] Henry B Moss, Daniel Beck, Javier González, David S Leslie, and Paul Rayson. BOSS: Bayesian optimization over string spaces. *arXiv preprint arXiv:2010.00979*, 2020.

[30] Dang Nguyen, Sunil Gupta, Santu Rana, Alistair Shilton, and Svetha Venkatesh. Bayesian optimization for categorical and category-specific continuous inputs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5256–5263, 2020.

[31] Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial bayesian optimization using the graph cartesian product. In *Advances in Neural Information Processing Systems*, pages 2910–2920, 2019.

[32] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

[33] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A flexible framework for multi-objective bayesian optimization using random scalarizations. In *Uncertainty in Artificial Intelligence*, pages 766–776. PMLR, 2020.

[34] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, pages 745–750, 2007.

[35] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science Advances*, 4(7):eaap7885, 2018.

[36] Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005.

[37] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

[38] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

[39] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

[40] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[41] Bernhard Scholkopf, Sebastian Mika, Chris JC Burges, Philipp Knirsch, K-R Muller, Gunnar Ratsch, and Alexander J Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[42] Anton Schwaighofer, Volker Tresp, and Kai Yu. Learning gaussian process kernels via hierarchical bayes. In *Advances in Neural Information Processing Systems*, pages 1209–1216, 2005.

[43] Daniel Schwalbe-Koda and Rafael Gómez-Bombarelli. Generative models for automatic chemical design. In *Machine Learning Meets Quantum Physics*, pages 445–467. Springer, 2020.

[44] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: a review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[45] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

[46] Kilho Shin, Marco Cuturi, and Tetsuji Kuboyama. Mapping kernels for trees. In *International Conference on Machine Learning*, 2011.

[47] Gregor Simm, Robert Pinsler, and José Miguel Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. In *International Conference on Machine Learning*, pages 8959–8969. PMLR, 2020.

[48] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems - Volume 2*, page 2951–2959, 2012.

[49] Shinya Suzuki, Shion Takeno, Tomoyuki Tamura, Kazuki Shitara, and Masayuki Karasuyama. Multi-objective bayesian optimization using pareto-frontier entropy. In *International Conference on Machine Learning*, pages 9279–9288. PMLR, 2020.

[50] Kevin Swersky, Yulia Rubanova, David Dohan, and Kevin Murphy. Amortized bayesian optimization over discrete spaces. In *Conference on Uncertainty in Artificial Intelligence*, pages 769–778. PMLR, 2020.

[51] Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity bayesian optimization with max-value entropy search and its parallelization. In *International Conference on Machine Learning*, pages 9334–9345. PMLR, 2020.

[52] 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, 2020.

[53] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Annual Conference on Neural Information Processing Systems*, 2001.

[54] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in neural information processing systems*, pages 6410–6421, 2018.

[55] Yehong Zhang, Trong Nghia Hoang, Bryan Kian Hsiang Low, and Mohan Kankanhalli. Information-based multi-fidelity bayesian optimization. In *NIPS Workshop on Bayesian Optimization*, 2017.

[56] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1):1–10, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discussed potential limitation in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work?
   [Yes] Since LADDER is a general expensive black-box optimization algorithm, it's social impact is determined by the nature of end-application it is employed in and the optimization objective for a particular application. For example, accelerating the design of drugs/vaccines vs. gain of function experimental selection.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)?
   [Yes] We provide our code in the supplementary material and a thorough description of the experimental setup is given in Section 4.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)?
   [Yes] All the training details are given in Section 4.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)?
   [Yes] We show the error bars in all our figures.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] we provide the details of the computing resource used for our experiments in Section 4.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators?
   [Yes] We cited the original authors' for code libraries and datasets that we used in Section 4.
   (b) Did you mention the license of the assets?
   [Yes] The license is kept in the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL?
   [Yes] We added our own Bayesian optimization code in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating?
   [Yes] We use all open-source codes available on public Github repositories.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? N/A

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]