# Accelerating Bayesian Optimization for Biological Sequence Design with Denoising Autoencoders

**Samuel Stanton** [1] **Wesley J. Maddox** [1] **Nate Gruver** [2] **Phillip Maffettone** [3] **Emily Delaney** [3]
**Peyton Greenside** [3] **Andrew Gordon Wilson** [1 2]

## Abstract

Bayesian optimization (BayesOpt) is a gold standard for query-efficient continuous optimization. However, its adoption for drug design has been hindered by the discrete, high-dimensional nature of the decision variables. We develop a new approach (LaMBO) which jointly trains a denoising autoencoder with a discriminative multi-task Gaussian process head, allowing gradient-based optimization of multi-objective acquisition functions in the latent space of the autoencoder. These acquisition functions allow LaMBO to balance the explore-exploit tradeoff over multiple design rounds, and to balance objective tradeoffs by optimizing sequences at many different points on the Pareto frontier. We evaluate LaMBO on two small-molecule design tasks, and introduce new tasks optimizing *in silico* and *in vitro* properties of large-molecule fluorescent proteins. In our experiments LaMBO outperforms genetic optimizers and does not require a large pretraining corpus, demonstrating that BayesOpt is practical and effective for biological sequence design.

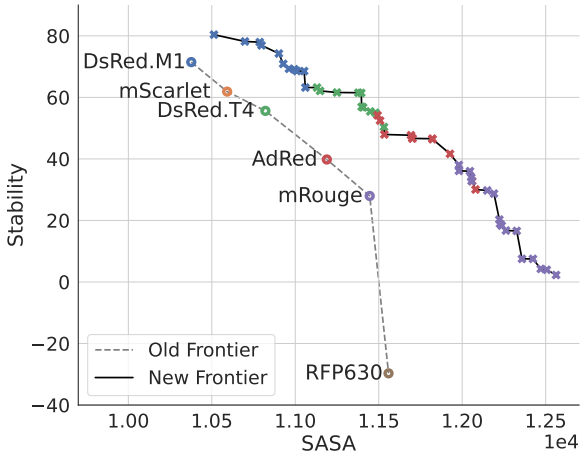



*Figure 1.* BayesOpt can be used to maximize the simulated folding stability (i.e. -dG, or the negative change in Gibbs free energy) and solvent-accessible surface area (SASA) of red-spectrum fluorescent proteins (RFPs). Higher is better for both objectives. The ancestor proteins are shown as colored circles, with corresponding optimized offspring shown as crosses. Stability correlates with protein function (e.g. how long the protein can fluoresce) while SASA is a proxy for fluorescent intensity.

## 1. Introduction

Modern drug development is a very costly endeavor, with estimates ranging from $310M to $2.8B for each new drug (Wouters et al., 2020). There are three major phases of drug development: 1) *target discovery*, the proposal of a biological mechanism hypothesized to treat a specific medical condition, 2) *drug design*, the specification of a molecular payload which will interact with the proposed mechanism, and 3) *clinical trials*, the evaluation of the efficacy and safety of the payload *in vivo*. Though each phase contributes to the total cost of development, in this work we focus on the drug design phase. In particular we will optimize real-valued target properties (such as neurotransmitter receptor affinity or protein folding stability) for payloads represented as discrete sequences (e.g. SMILES strings). Since the mappings from sequence to target are often unknown, expensive to observe *in vitro*, and difficult to simulate, we pose biological sequence design as a costly black-box optimization (BBO) problem over a large discrete search space.

Recent work applying deep learning to biological tasks has primarily focused on learning from a static, offline dataset (Rao et al., 2019; Jumper et al., 2021; Baek et al., 2021; Meier et al., 2021; Rives et al., 2021). When these models are used to select new sequences to label, they are applied in a one-shot fashion, without accounting for future design rounds (Gligorijevic et al., 2021; Biswas et al., 2021). Because labels are scarce for many important targets, it is

[1]Center for Data Science, New York University, New York, USA [2]Courant Institute of Mathematical Sciences, New York University, New York, USA [3]BigHat Biosciences, San Mateo, CA, USA. Correspondence to: Samuel Stanton <ss13641@nyu.edu>.

*Proceedings of the $39^{th}$ International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. 

important to account for model uncertainty to manage the explore-exploit tradeoff (O'Donoghue et al., 2018).

Bayesian optimization (BayesOpt) is a powerful class of BBO algorithms, explicitly designed to *coherently* reason about online decision-making based on incomplete information (Brochu et al., 2010). BayesOpt balances the explore-exploit tradeoff in a principled way, relying on a probabilistic discriminative model to prioritize decisions with the highest potential payoff. At each decision point the discriminative model produces a posterior distribution over the hypothesis space of all functions the model can represent. The posterior formally represents the degree to which any particular hypothesis is plausible given the data and model assumptions. To make a decision, BayesOpt selects the best decision as defined by an *acquisition function*, such as expected improvement (EI), with each hypothesis contributing to the acquisition value in proportion to its posterior probability (Jones et al., 1998). BayesOpt is not only philosophically appealing, it is provably a *no-regret* strategy under the right conditions (Srinivas et al., 2010).

Like many Bayesian methods, the barriers hindering widespread adoption of BayesOpt are not conceptual, but *practical*. Drug design in particular is a natural application domain, but introduces multiple challenges. *Discrete, high-dimensional inputs:* antibody therapeutic payloads are often RNA proteins which instruct the patient's immune system to produce the desired antibody. When proteins are represented as a sequence of residues, each identifying one of 20 possible amino acids, even a fairly small 200-residue protein is only one of $20^{200} \approx 1.6 \times 10^{260}$ options. By contrast, conventional BayesOpt works best on problems with 10 or fewer continuous decision variables, due to the properties of standard kernels, and the lack of gradients to maximize the acquisition function. *Batched, multi-objective experiments:* because considerations including efficacy, toxicity, and yield must all be taken into account, drug design is inherently multi-objective. Furthermore sequences are labeled in batches, necessitating the use of more sophisticated acquisition functions than standard workhorses like EI. *Data-scarcity:* wet lab experimental data is expensive and challenging to collect, so it is rare to have large-scale datasets with labels for the exact target properties of interest.

No single BayesOpt method has been shown to simultaneously address all of these challenges (see Section 3). As a result, previous methods have necessarily only been evaluated on very stylized tasks that fail to capture key aspects of real drug design problems. In this work we present **La**tent **M**ulti-Objective **B**ayes**O**pt (**LaMBO**) to address this deficiency, and propose a novel *in silico* task which emulates protein design tasks more closely than common open-source drug design benchmarks. We preview our results applying LaMBO to this new task in Figure 1, maximizing the stability and SASA of RFPs derived from the fpBase dataset (Lambert, 2019), with the initial Pareto frontier in objective space shown as a dashed line. The new Pareto frontier (the solid line) discovered by LaMBO is superior, as it is characterized more densely by new sequences that are Pareto improvements over their ancestors. In short, our contributions are as follows:

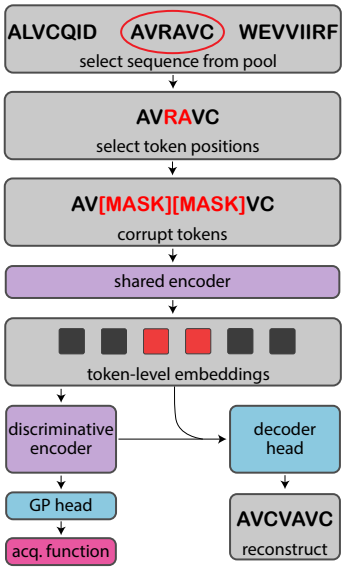



*Figure 2.* LaMBO with a non-autoregressive denoising autoencoder (DAE) architecture: a shared encoder produces continuous token-level embeddings $Z$ from corrupted inputs, which are passed to a discriminative encoder to produce target-specific token-level embeddings $Z'$. A generative decoder head receives both $Z$ and $Z'$ as input, and a discriminative Gaussian process (GP) head pools $Z'$ to predict the objective values. The GP head allows the use of principled acquisition functions to manage the explore-exploit tradeoff, and the DAE head allows LaMBO to follow the acquisition gradient in latent space when selecting new queries.

1. We propose the LaMBO architecture, a novel combination of a generative DAE with a discriminative deep kernel learning GP, with a simple joint training procedure and an effective decision optimization routine.

2. We propose a new *in silico* large-molecule task to augment existing open-source drug design benchmarks.

3. We evaluate LaMBO *in silico* on two small-molecule tasks and our new large-molecule task, comparing to both genetic and latent-space BBO baselines, showing improved sample efficiency and solution quality.[1]

4. We present *in vitro* wet lab results using LaMBO to discover brighter, more thermostable red fluorescent proteins.

[1]Code here: `github.com/samuelstanton/lambo`.

## 2. Preliminaries

We now introduce the problem setting and BayesOpt.

### 2.1. Discrete Multi-Objective Sequence Design

We first define the input space $\mathcal{X} = \times_{i=1}^{t} \mathcal{V}$, where $\mathcal{V}$ is an ordered, discrete vocabulary of $v$ tokens, $t$ is the max sequence length, and $\times$ is the Cartesian product. $\mathcal{V}$ includes a padding token to accommodate sequences of varying length. Because $|\mathcal{X}| = |\mathcal{V}|^t$ is exponential in $t$, $\mathcal{X}$ becomes too large to enumerate quickly as the sequence length grows, even if $|\mathcal{V}|$ is relatively small. As a result, sequence optimization usually starts with a library or pool of initial sequences (see Figure 2, top), which are modified to produce new candidate sequences. When posed in this way, the optimization problem is restructured into three nested decisions: (1) Choose a base sequence from the pool. (2) Choose which positions on the sequence to change. (3) Choose how the sequence will be changed at those positions.

We represent sequence design as a multi-objective optimization problem $\min_{\mathbf{x}\in\mathcal{X}}(f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$, where $k$ is the number of objectives and each $f_i : \mathcal{X} \to \mathbb{R}$ is an expensive, black-box function of decision variables $\mathbf{x} \in \mathcal{X}$. Given two feasible solutions $\mathbf{x}$ and $\mathbf{x}'$, $\mathbf{x}$ *dominates* $\mathbf{x}'$ ($\mathbf{x} \prec \mathbf{x}'$) if $f_i(\mathbf{x}) \leqslant f_i(\mathbf{x}')\ \forall i \in \{1, \ldots, k\}$, and $\exists i \in \{1, \ldots, k\}$ s.t. $f_i(\mathbf{x}) < f_i(\mathbf{x}')$. In general, there will not be a single dominating solution $\mathbf{x}^*$; instead, we define the set of non-dominated solutions (i.e. the true *Pareto frontier* $\mathcal{P}^*$),

$$\mathcal{P}^* := \{\mathbf{x} \in \mathcal{X} \mid \{\mathbf{x}' \in \mathcal{X} \mid \mathbf{x}' \prec \mathbf{x}, \mathbf{x}' \neq \mathbf{x}\} = \varnothing\}. \quad (1)$$

Since $\mathcal{P}^*$ is unknown, we seek a set of candidate solutions $\mathcal{P}$ that are close in objective space to those in $\mathcal{P}^*$. We find these solutions by maximizing the hypervolume bounded by the extremal points in $\mathcal{P} \cup \{\mathbf{x}_{\text{ref}}\}$, where $\mathbf{x}_{\text{ref}}$ is some reference solution.

### 2.2. BayesOpt

See Brochu et al. (2010) and Frazier (2018) for a more complete introduction to BayesOpt. BayesOpt constructs a probabilistic *surrogate model* $\hat{f} \in \mathcal{F}$ which is trained to emulate $f$ from a dataset $\mathcal{D}_n := \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$, where $\mathbf{y}_i$ are noisy observations of $f(\mathbf{x}_i)$ (e.g. $\mathbf{y}_i = f(\mathbf{x}_i) + \varepsilon_i,\ \varepsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_k)$). The surrogate posterior distribution is used to define an *acquisition function* $a : \mathcal{X} \times \mathcal{F} \to \mathbb{R}$, which in turn defines an *inner loop* optimization problem to select new query point(s). The objective function is then queried at the selected points and the surrogate is retrained on the augmented dataset, and the procedure repeats, forming an *outer loop* (Algorithm 1).

GPs (Williams & Rasmussen, 2006) are often preferred as surrogates because they have closed-form posteriors and work well in data-scarce regimes. The inductive biases of a GP are primarily determined by the choice of kernel, which defines a prior distribution over the values of $f$ for any finite collection of inputs. Most commonly used GP kernels (e.g. RBF or Matérn) rely on $\ell_2$ distance between inputs to determine the prior covariance between outputs. When the inputs are low-dimensional (e.g. $d = 10$) such kernels work well, but in high dimensions the $\ell_2$ norm is often a poor choice of distance metric (Srinivas et al., 2010; Wang et al., 2016). This limitation has motivated the development of *deep kernel learning* (DKL), which learns a low-dimensional continuous embedding via an encoder such as a convolutional neural network (CNN) (Wilson et al., 2016). Although GPs are kernel-based models, there is a range of well-known methods to scale them to large, online datasets, notably inducing point methods like stochastic variational GPs (SVGPs) which admit the use of stochastic optimizers (Hensman et al., 2013; Maddox et al., 2021c).

**Algorithm 1** The BayesOpt outer loop

**Inputs:** hypothesis space $\mathcal{F}$, acquisition $a$, dataset $\mathcal{D}_0$.
**for** $i = 0, \ldots, i_{\max} - 1$ **do**
  Fit $\hat{f} \in \mathcal{F}$ to $\mathcal{D}_i$.
  $\mathbf{x}_i^* = \min_{\mathbf{x}\in\mathcal{X}} a(\mathbf{x}, \hat{f})$. $\leftarrow$ the inner loop
  Observe $\mathbf{y}_i \sim p(\cdot|\mathbf{x}_i^*)$.
  $\mathcal{D}_{i+1} = \mathcal{D}_i \cup (\mathbf{x}_i^*, \mathbf{y}_i)$.
  $\mathcal{P}_{i+1} = \text{nondominated}(\mathcal{D}_{i+1})$.
**end**
**return** $\mathcal{P}_{i_{\max}}$

## 3. Related Work

**Discrete Optimization by Sampling:** genetic algorithms (GAs) such as NSGA-II (Deb et al., 2002) slowly evolve a good solution by random mutation. GAs are a simple, popular baseline for BBO problems, but are known for being inefficient (Turner et al., 2021). One solution is to continue generating mutations randomly, but screen the proposed queries with a discriminative model before labeling (Nigam et al., 2019; Yang et al., 2019b). Other solutions focus on proposing mutations more intelligently, including RL-based approaches (Angermueller et al., 2020a;b) and generative approaches (Jensen, 2019; Biswas et al., 2021; Zhang et al., 2021). In particular the generative approach described by Lee et al. (2018) and Gligorijevic et al. (2021) inspired the LaMBO architecture. All the approaches just discussed are greedy in the sense that they rely on point estimates of the objective values to select new queries.

**Discrete BayesOpt:** excluding library-based approaches such as Yang et al. (2019a), discrete BayesOpt methods can be categorized by how they structure the inner loop optimization problem. Some methods use substring kernels (SSKs), optimizing queries directly in sequence space with

a GA, and evaluating only on small tasks (Lodhi et al., 2002; Beck & Cohn, 2017; Moss et al., 2020). Khan et al. (2022) is an example of concurrent antibody design work in this vein, exploiting task-specific knowledge of complementarity-determining regions (CDRs) of the antibodies to make the problem tractable. See Appendix C.1 for more discussion and an experiment comparing SSK and DKL GPs in the offline regression setting.

Latent-space optimizers learn continuous sequence embeddings $Z$, which are shared by a generative decoder modeling $p(\mathbf{x}|Z)$ and the discriminative surrogate modeling $p(\mathbf{y}|Z)$ (Deshwal & Doppa, 2021; Grosnit et al., 2021; Maus et al., 2022). Thus $Z$ can be optimized with gradient-based methods to produce new sequences. LaMBO is most similar to Latent-Space BayesOpt (LSBO) (Gómez-Bombarelli et al., 2018), since both methods model $p(\mathbf{y}|Z)$ as a GP and model $p(\mathbf{x}|Z)$ via an autoencoder. LSBO uses a VAE pretrained on a large dataset to solve single-objective tasks, training the VAE weights and the auxiliary GP head separately. With a specialized architecture proposed by Jin et al. (2018) and a biased VAE objective proposed by Tripp et al. (2020), LSBO has been shown capable of solving simple small-molecule tasks such as maximizing penalized logP. Aside from LaMBO's use of DAEs rather than VAEs, this work improves upon LSBO in multiple ways, such as enabling the use of a general-purpose architecture for both small and large molecules, removing the need to pretrain the autoencoder, providing a reliable procedure for jointly training generative and GP heads with a shared encoder, and the introduction of multi-objective tasks and acquisition functions. See Appendix C.2 for a comparison between LSBO and LaMBO in the single-objective BBO setting.

**Multi-Objective BayesOpt:** Daulton et al. (2020) proposed a batch version of expected hypervolume improvement (EHVI) (Emmerich, 2005; Emmerich et al., 2011), an extension of EI to multiple objectives. In follow-up work, Daulton et al. (2021a) proposed the noisy expected hypervolume improvement (NEHVI) acquisition function, which extends noisy expected improvement (NEI) (Letham et al., 2019) to multiple objectives. Multi-task GPs (MTGPs) have previously been used as surrogates for multi-objective BayesOpt (Shah & Ghahramani, 2016), including recent work scaling MTGPs up to thousands of training examples or objectives in the continuous setting (Daulton et al., 2021b; Maddox et al., 2021a). We make use of NEHVI and MTGPs, including an efficient MTGP posterior sampling approach developed in Maddox et al. (2021b).

## 4. Latent-Space Multi-Objective BayesOpt

We now describe the key ideas behind LaMBO, summarized in Figure 2 and Algorithm 2.

### 4.1. Architecture Overview

We use a non-autoregressive denoising autoencoder to map discrete sequences to and from sequences of continuous token-level embeddings, with a multi-task GP head operating on pooled sequence-level embeddings. Unlike previous work combining GPs with deep generative models, we do not require a pretrained autoencoder, nor do we require the surrogate to be completely reinitialized after receiving new data. Furthermore, we demonstrate that both stochastic variational and exact GP inference can be used, alleviating concerns regarding computational scalability or applicability to noisy objectives with non-Gaussian likelihoods. See Appendix B for more details.

**Shared Encoder:** we use a non-autoregressive bidirectional encoder $g(\mathbf{x}, \theta_{\text{enc}}) = [\mathbf{z}_1, \dots, \mathbf{z}_t] = Z$, where $\mathbf{z}_i \in \mathbb{R}^d$ are latent token-level embeddings. In particular, our encoder is composed of 1D CNN layers, using standard vocabulary embeddings and sinusoidal position embeddings, padding token masking, skip-connections, and layernorm. A key advantage of using a DAE rather than a VAE is that projects like ChemBERTa, TAPE and ESM have already openly released large DAE models trained on large sequence corpora (Chithrananda et al., 2020; Rao et al., 2019; Rives et al., 2021). As a result, encoders from these models could be used as drop-in replacements for our small CNN encoder.

**Discriminative Head:** this head takes an encoded sequence $Z$ and outputs a scalar value indicating the utility of selecting that sequence as a query point. We pass $Z$ to a discriminative encoder $w$ to obtain transformed embeddings $Z'$, then pool $Z'$ into low-dimensional sequence-level features. The discriminative encoder is smaller than the shared encoder, for example a single residual CNN block. The pooling operation $(|\mathcal{I}(\mathbf{x})|)^{-1} \sum_{i \in \mathcal{I}(\mathbf{x})} \mathbf{z}'_i$ averages token-level embeddings over a restricted index set $\mathcal{I}(\mathbf{x})$ which excludes positions corresponding to padding tokens. We define a multi-task GP kernel by combining a $5/2$ Matérn kernel evaluated on these sequence features with an intrinsic model of coregionalization (ICM) kernel over $f_i$ (Goovaerts et al., 1997; Alvarez et al., 2011; Rasmussen & Williams, 2008). The resulting GP outputs a posterior predictive distribution $p(f|\mathcal{D})$, which is passed as input to the acquisition function. We use the noisy expected hypervolume improvement (NEHVI) acquisition from Daulton et al. (2021a) since some objectives (particularly those involving biological data) are inherently noisy.

**Generative Decoder Head:** this head ($h$) maps a sequence of token-level embeddings to a predictive distribution over $\mathcal{X}$ and can either be a simple masked language model (MLM) head (Devlin et al., 2018) or a full seq2seq latent non-autoregressive neural machine translation (LANMT) decoder (Shu et al., 2020). In this work we make use of both. An MLM head is simpler and easier to control than

an LANMT decoder, which allows for careful comparisons between LaMBO and discrete genetic optimizers. Despite their complexity, LANMT decoders accommodate insertions and deletions more gracefully than MLM heads by means of a length prediction head and length transform operation, allowing the same latent representation to be decoded to sequences of varying length. In either case, the decoder uses the same layer types as the shared encoder $g$, and takes both $Z$ and $Z'$ as input.

### 4.2. Initializing the Latent Embeddings

At each outer-loop iteration $i$, there are many choices of $\mathbf{x}$ we are confident will *not* improve our current solution $\mathcal{P}_i$, corresponding to large flat regions of the inner-loop loss surface. If the inner-loop is not initialized carefully, it is very likely the initial solution will fall in one of these flat regions, severely hampering gradient-based optimization. If $\mathbf{x}$ was continuous, we could use initialization heuristics from previous work to avoid this pitfall (Balandat et al., 2020; Daulton et al., 2021a). Instead we now show how the the same corruption process used to train and sample from DAEs can be used as a robust initialization procedure for discrete $\mathbf{x}$ by initializing the inner loop solution in latent-space ($Z_0$) with corrupted, encoded variants of the current outer-loop solution $\mathcal{P}_i$.

**Selecting base sequences:** we begin with a set of seed sequences $\mathcal{X}_{\text{pool}}$ and select a subset, $\mathcal{X}_{\text{base}} \subset \mathcal{X}_{\text{pool}}$. After each optimization round, the latest query sequences are added to $\mathcal{X}_{\text{pool}}$ and can serve as future base sequences. Choosing $\mathcal{X}_{\text{base}}$ well is critical for fast convergence. If too many low quality sequences are added, too much computation is spent optimizing them. Conversely, only selecting the current Pareto sequences could hinder exploration of sequences with potential for improvement. The interaction between the online queries and the decoder head must also be considered, since we want to prevent the generative samples from collapsing to a couple sequences.

We populate $\mathcal{X}_{\text{base}}$ first with the current Pareto sequences $\mathcal{P}_i$, then with random sequences (without replacement) that were on the Pareto frontier in previous optimization rounds ($\mathcal{P}_{<i}$), and finally fill any remaining space in the base set with random sequences from the entire optimization history. In practice, we took the size of the base set to be the same as the query batch size $b$. We perform multiple restarts during the inner loop, and each restart samples $b$ sequences from $\mathcal{X}_{\text{base}}$, with replacement. We do not sample any base sequences uniformly at random, but according to a weighted distribution $\Delta(\mathcal{X}_{\text{pool}})$ to ensure that high-scoring sequences are more likely to be optimized than low-scoring ones.

Let $r_i(\mathbf{x}_j, X)$ be the rank of $\mathbf{x}_j \in X$ w.r.t. the 0-indexed dense ranking of $X$ according to $f_i$, and let $r_{\max}(\mathbf{x}_j, X) = \max_i r_i(\mathbf{x}_j, X)$. The sampling weight $w_j$ of $\mathbf{x}_j \in \mathcal{X}_{\text{pool}}$ is

$$w_j = s_j(-\log(1+\mathbf{r})/\tau), \qquad \mathbf{r} = [r_{\max}(\mathbf{x}_1, X), \dots, r_{\max}(\mathbf{x}_p, X)], \tag{2}$$

where $s_j(\mathbf{v}) = \exp(v_j)/\sum_i \exp(v_i)$ is the softmax function and $\tau \in (0, +\infty)$ is the softmax temperature. In other words, we choose the least favorable ranking across all objectives for each $\mathbf{x}$ to determine its weight. This type of weighting is similar in spirit to a procedure used by Tripp et al. (2020) to bias a VAE loss in favor of high-scoring sequences.

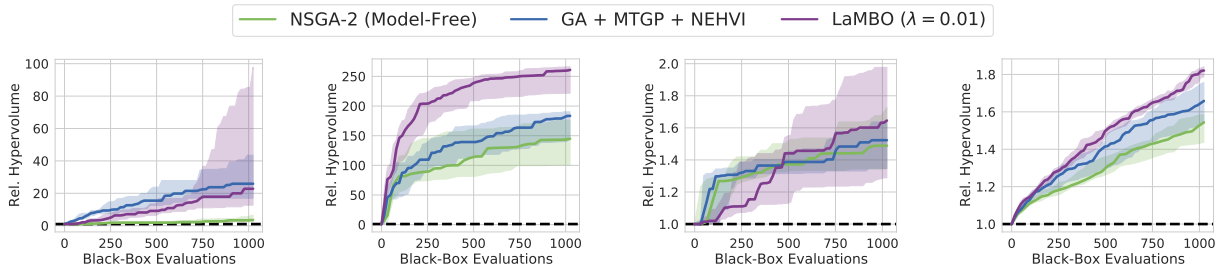
**Algorithm 2** The LaMBO inner loop. For clarity, steps where LaMBO differs from LSBO are shown in blue.

**Inputs:** acquisition $a$, corruption $c$, shared encoder $g$, discriminative encoder $w$, decoder $h$, base sequences $\mathcal{X}_{\text{base}}$, and batch size $b$.

$v^* \leftarrow +\infty$
$Z_0 = \{g \circ c(\mathbf{x}_0), \dots, g \circ c(\mathbf{x}_b)\},\ \mathbf{x}_m \in \mathcal{X}_{\text{base}}$
**for** $j = 0, \dots, j_{\max}$ **do**
  $Z'_j = w(Z_j)$
  $Z_{j+1} = Z_j - \eta \nabla_Z \left[a(Z'_j) - \lambda \mathbb{H}\left[h(Z_j, Z'_j)\right]\right]$
  $\mathcal{X}_{\text{cand}} \leftarrow \{\mathbf{x}'_1, \dots, \mathbf{x}'_b\} \sim h(Z_j, Z'_j)$
  $Z_{\text{cand}} \leftarrow [g(\mathbf{x}'_1), \dots, g(\mathbf{x}'_b)]^\top$
  $v_j = a(w(Z_{\text{cand}}))$
  **if** $v_j < v^*$ **then**
    $v^*, \mathcal{X}^* \leftarrow v_j, \mathcal{X}_{\text{cand}}$
  **end**
**end**
**return** $\mathcal{X}^*$

**Selecting base sequence positions:** after obtaining $\mathcal{X}_{\text{base}}$ we apply a corruption function $c$ to each element before passing them as input to the encoder, similar to the procedure proposed in Gligorijevic et al. (2021). The corruption function first selects positions in each sequence to modify, then selects a modification (substitution, insertion, deletion) at those positions. We uniformly sample positions not occupied by utility tokens, such as padding tokens.

**Selecting corruption operations:** once sequence positions have been selected, the corruption function chooses corruption operations to apply at those positions. For LaMBO, the corruption procedure differs depending on whether the decoder is an MLM head or a LANMT head. If the decoder is an MLM head then all operations are substitution operations, and the replacement tokens are all masking tokens. If the decoder is an LANMT head then the operation type is chosen randomly and replacement tokens are sampled uniformly from $\mathcal{V}$. Note we use a similar corruption procedure to train the decoder head.




**(a) Bigrams** **(b) logP + QED** **(c) DRD3 Docking + SA** **(d) Stability + SASA**

*Figure 3.* On all four tasks (described in Section 5.1), LaMBO outperforms genetic algorithm baselines, specifically NSGA-2 and a model-based genetic optimizer with the same surrogate architecture (GA + MTGP + NEHVI). Performance is quantified by the hypervolume bounded by the optimized Pareto frontier. The midpoint, lower, and upper bounds of each curve depict the 50%, 20%, and 80% quantiles, estimated from 10 trials. See Section 5.2 for more discussion.

### 4.3. Sequence Optimization

At a high level, we solve the inner-loop by treating the output of the decoder head $h$ as a proposal distribution. We iteratively refine the proposal distribution by following a regularized acquisition gradient in latent space, drawing and scoring batches of sequences along the way (Algorithm 2).

More precisely, once we have selected and corrupted the base sequences, we pass them through the encoder to produce latent embeddings $Z_0$ that serve as the initial solution for the inner-loop optimization problem. Then for each optimization step $0 \leqslant j < j_{\max}$, we take $Z_{j+1} = Z_j - \eta \nabla_Z [\ell_{\text{query}}(Z_j)]$, where $Z'_j = w(Z_j)$ (i.e. the output of the discriminative encoder $w$),

$$\ell_{\text{query}}(Z_j) = a(Z'_j) + \lambda \mathbb{H}[h(Z_j, Z'_j)], \tag{3}$$

$\mathbb{H}$ is the Shannon entropy, and $\eta$, $\lambda$ are hyperparameters controlling the step-size and regularization strength. We sample $\mathcal{X}_{\text{batch}} = \{\mathbf{x}'_0, \ldots, \mathbf{x}'_b\} \sim h(Z_j, Z'_j)$, and score $\mathcal{X}_{\text{batch}}$ by passing it *uncorrupted* through the shared encoder and discriminative head. If we see that $\mathcal{X}_{\text{batch}}$ has the best acquisition value so far, we store it and continue optimizing. Note that this procedure produces a different result than simply optimizing $Z$ and decoding once at the end. Recall that the decoder is stochastic, and the ultimate goal of the inner loop is to produce *sequences* with high acquisition value, not just high-scoring latent embeddings.

We observed that following the unregularized acquisition gradient caused the decoder entropy to quickly increase, resulting in very uniform proposal distributions. When the proposal distributions are uniform, LaMBO essentially performs a variant of random search. However, because $Z_0$ is produced in the same way the decoder head is trained (i.e. the same corruption process), we expect that it will be close to other latent embeddings seen during training, and the decoder entropy will be relatively low. These observations motivated us to include the proposal entropy penalty in Eq. (3), to encourage $Z_j$ to stay in a region of latent space where the decoder has non-uniform predictive distributions.

We also considered choosing a small step-size $\eta$ to implicitly confine $Z_j$ to a small region around $Z_0$, but we found it difficult to choose $\eta$ both large enough to improve the $\ell_{\text{query}}$ and small enough to prevent uniform proposals. The proposal entropy penalty also has the added benefit of smoothing the query loss surface when using acquisitions like NEHVI, which helps make the inner loop less dependent on a good initialization. Adding a regularization term for improved control of the acquisition function has been previously studied in the continuous setting by Shahriari et al. (2016).

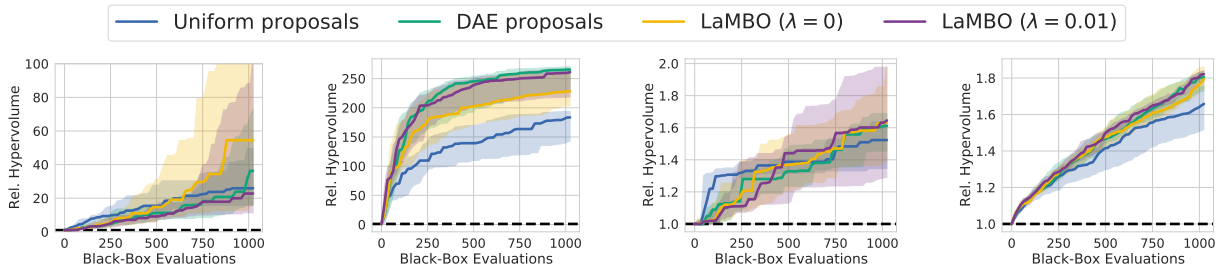
## 5. Empirical Evaluation

We now evaluate LaMBO on a suite of small-molecule and large-molecule sequence design tasks. In Section 5.1 describe our suite of *in silico* tasks, including a new multi-objective large-molecule task which in which we maximize the folding stability and SASA of RFPs. See Appendix C.1 for an experiment comparing SSK GPs and DKL GPs, and Appendix C.2 for an experiment comparing LSBO and LaMBO. In Section 5.2 we show that LaMBO outperforms strong genetic algorithm baselines in a carefully controlled comparison, followed by two investigative experiments in Section 5.3 which give insight into the design choices behind LaMBO. Finally in Section 5.4 we analyze molecules designed by LaMBO, including *in vitro* wet lab results showing the discovery of improved RFP variants.

### 5.1. Evaluation procedure

We consider the following *in silico* evaluation tasks, with full descriptions in the appendix:

- **Bigrams**: optimize short strings (around 32 tokens) uniformly sampled from $\mathcal{X}$ to maximize the counts of three predetermined bigrams (Appendix A.1).




**(a) Bigrams** **(b) logP + QED** **(c) DRD3 Docking + SA** **(d) Stability + SASA**

*Figure 4.* An ablation of LaMBO's main components. Starting from our model-based genetic algorithm baseline (uniform proposals), we cumulatively add the elements described in Section 4.3: (1) DAE-generated proposals, (2) DAE proposal optimization following $\nabla_Z[\ell_{\text{query}}]$ with $\lambda = 0$. (see Eq. (3)), and (3) DAE proposal optimization with $\lambda = 0.01$. DAE proposals improve performance on all tasks. Proposal optimization is most helpful on **Bigrams** where the starting sequence distribution is very unlikely to produce high-scoring queries. The entropy penalty is detrimental when working with random sequences **(a)**, but is helpful when working with biological sequences **(b-d)**. The midpoint, lower, and upper bounds of each curve depict the 50%, 20%, and 80% quantiles, estimated from 10 trials.

- **logP + QED:** optimize SELFIES-encoded small molecules (50-100 tokens) w.r.t. logP and QED (Appendix A.2).
- **DRD3 docking + SA:** optimize SELFIES-encoded small molecules (50-100 tokens) w.r.t. DRD3 docking and synthetic accessibility (SA) (Appendix A.3).
- **Stability + SASA:** optimize large-molecule RFPs (around 200 tokens) w.r.t. folding stability and SASA. Both objectives require the 3D protein structure to compute, however we treat the folding simulator as part of the black-box objective (Appendix A.4).

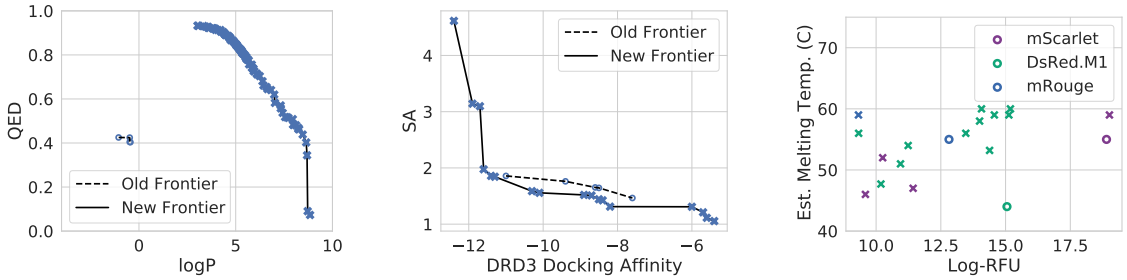
Unless otherwise noted, each task begins with 512 examples in its start pool, and collects a total of 1024 online queries in batches of 16. No additional pretraining data is used. Each method uses the same architecture and hyperparameters for all tasks (Appendix B). We evaluate all methods by comparing the relative improvement of the hypervolume bounded by the Pareto front after $x$ black-box function calls compared to the starting hypervolume.

### 5.2. Comparing to Multi-Objective Genetic Optimizers

In Figure 3 we compare LaMBO with a MLM decoder head against two genetic algorithm (GA) baselines. For this experiment we set the entropy penalty at $\lambda = 0.01$. The simplest baseline is NSGA-2, a robust model-free multi-objective GA, which effectively simply randomly mutates solutions along the Pareto frontier. The other baseline is a model-based GA which also randomly mutates solutions but also screens new queries with a discriminative model, for which we use the same architecture and acquisition function (NEHVI) as LaMBO. Both GA baselines use a uniform mutation proposal distribution. In this experiment all optimizers are only allowed to change a single token per optimization round, and each optimizer selects base sequences and token positions in the same way. The model-based GA differs from LaMBO primarily in two respects: (1) the encoder is trained only through the supervised loss, and (2) the proposal distribution is uniform rather than generated by a DAE. In fact LaMBO can be viewed as a generalization of the model-based GA, where the proposal distribution is learned and optimized, rather than fixed *a priori*. The effect of (2) is most strongly seen in **logP + QED**, since the task requires very little exploration and the SELFIES vocabulary for small molecules is significantly larger than the amino acid vocabulary for proteins. LaMBO performs well on all four tasks, particularly those involving natural sequences **(b-d)**. In contrast the starting distribution over sequences in the **Bigrams** task has the highest possible entropy, so LaMBO learns a good sampling distribution more slowly.

### 5.3. Analyzing LaMBO

Having demonstrated that LaMBO compares favorably to genetic optimizers, we now examine the different components of LaMBO and how each contributes to performance. We first disentangle the effect of replacing a uniform proposal distribution with a DAE-generated one, and the effect of optimizing the proposal distribution by gradient descent on $\ell_{\text{query}}$. In Figure 4 we interpolate between the model-based GA baseline in Figure 3 and LaMBO by cumulatively adding DAE-generated proposals, proposal optimization, and the proposal entropy penalty. We find that unoptimized DAE proposals with NEHVI screening is a strong baseline on all tasks. Proposal optimization is the most useful in **Bigrams**, where the true non-dominated solutions $\mathcal{P}^*$ lie far outside the starting sequence distribution, requiring more exploration. For the same reason the entropy penalty is somewhat detrimental specifically in **Bigrams**, since it keeps new queries near those previously seen. See Figure




**(a) logP + QED** **(b) DRD3 docking + SA** **(c) RFP Wet Lab Results**

*Figure 5.* Here we show the old and new Pareto fronts in objective space discovered by LaMBO in Section 5.2. For **(a)** higher is better for both objectives, and for **(b)** lower is better. For all tasks every point in the old frontier is strictly dominated by at least one point in the new frontier, and solutions are evenly spread across the new frontiers. In **(c)** we provide wet lab measurements of brightness ($x$ axis) and thermostability ($y$ axis) for proteins optimized for **Stability + SASA**. Higher is better for both objectives. We discovered new, non-dominated protein variants for three of the five ancestor proteins in our evaluation set.

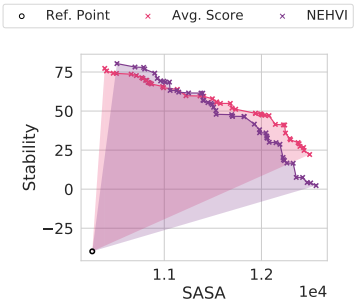



*Figure 6.* Pareto fronts of LaMBO and a variant that optimizes the expected average objective value on the RFP task. NEHVI incentivizes more exploration of the frontier extremities, resulting in solutions representing more diverse functional tradeoffs and slightly higher hypervolume.

11 in Appendix C for more discussion.

In Figure 6, we evaluate the sensitivity of LaMBO to the choice of acquisition function on the RFP task. We compare the final Pareto frontier obtained with a simple multi-objective scalarization (averaging normalized scores) to the frontier obtained with NEHVI. Score averaging focuses optimization on solutions with similar tradeoffs, pushing the interior of the frontier out quickly. This behavior leads to less exploration than NEHVI and thus a slightly lower hypervolume of 1.51 as compared to NEHVI's hypervolume of 1.57. Our findings corroborate similar results in previous work comparing scalarization and hypervolume-based acquisitions in different problem settings (Emmerich, 2005; Emmerich et al., 2011; Daulton et al., 2020).

### 5.4. Analyzing Sequence Designs

Though metrics like hypervolume are useful for conducting baseline comparisons and ablation studies, such metrics do not give much insight into more qualitative aspects of the sequences we are designing. In Figure 5(a-b) we show the Pareto fronts found by LaMBO for our two small-molecule tasks, as we did in Figure 1 for the large-molecule task. For every task every sequence on the old frontier is strictly dominated by at least one sequence on the new frontier.

So far we have shown that LaMBO can optimize *in silico* objectives effectively, and argued in Appendix A.4 that the **Stability + SASA** objectives are likely correlated with properties of RFPs we actually wish to optimize in the real world, specifically brightness and thermostability. In Figure 5(c) we evaluate *in vitro* protein sequences designed by LaMBO to optimize **Stability + SASA**, reporting brightness (log relative fluorescence units, log-RFU) and thermostability (protein melting temperature). We discovered non-dominated variants of three of the five ancestor sequences in our evaluation set, including strictly dominant variants of mScarlet and DsRed.M1, with inconclusive results for the other two ancestor sequences. See Appendix A.5 for more details about our experimental procedure and results. Our results indicate that **Stability + SASA** are good *in silico* proxy objectives for a real-world protein design task.

In Figure 7 we visualize solutions at different points on the optimized Pareto frontiers found by LaMBO for our two small-molecule tasks. Intriguingly, we find that although we did not specifically optimize for solution diversity in sequence space, the sequences change significantly as we move along the frontiers. In contrast, many generative-only approaches use heuristics to explicitly encourage diversity in sequence space and prevent solution collapse.

Some of the proposed molecules shown have features such as large macrocycles in 7(a)-2 and 7(b)-1, or three-carbon rings in 7(a)-4 and 7(a)-5, that are likely not synthetically accessible (Gao et al., 2021). However, we note that **logP + QED** does not explicitly incentivize accessibility, and re-

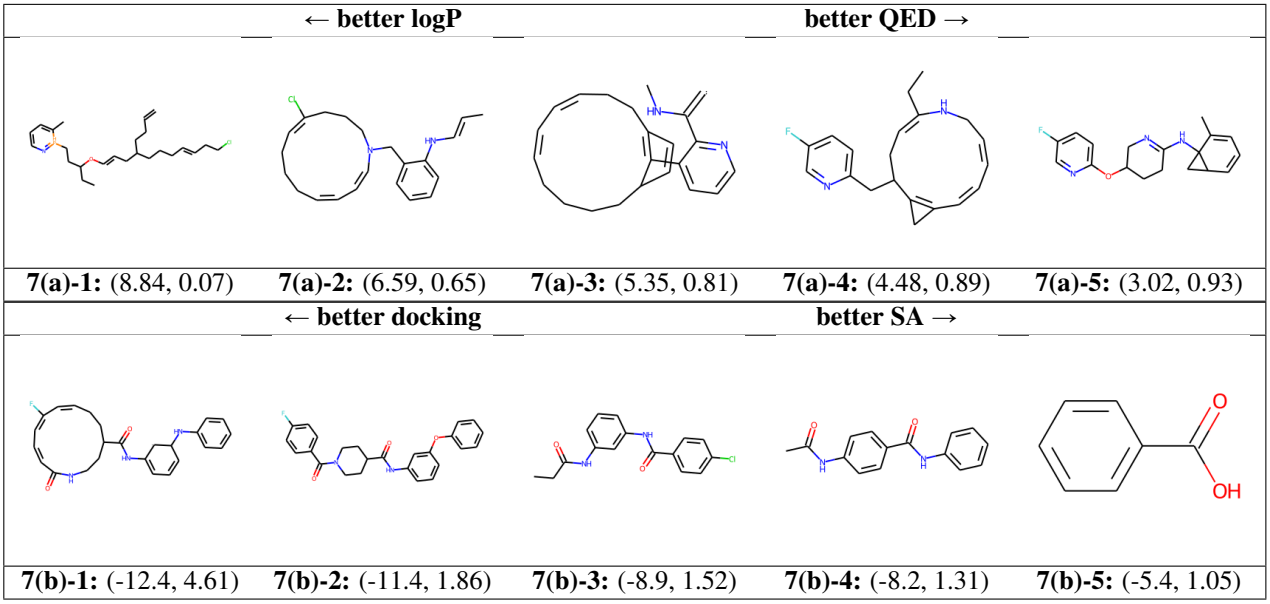



*Figure 7.* Example molecules at varying points on the new Pareto fronter discovered by LaMBO in Section 5.2 for **logP + QED** (top) and **DRD3 docking + SA** (bottom). Under each molecule we show the objective values as index : $(y_1, y_2)$. From left to right $y_1$ worsens and $y_2$ improves. The Pareto frontier is a rich, low-dimensional space that can be analyzed much more easily than the whole search space.

markably one can deduce that molecules with large macrocycles are difficult to synthesize even with little knowledge of chemistry by simply comparing molecules 7(b)-1 and 7(b)-2. Our results highlight the multi-objective nature of drug design, the need for careful objective selection (i.e. target discovery), and the human-interpretable insights that can be gained by studying the differences between non-dominated solutions found by machine learning methods.

## 6. Discussion

Drug design is quickly emerging as an extremely important application of machine learning. BayesOpt has extraordinary potential for this domain, but existing approaches struggle to extend to high-dimensional, discrete, multi-objective design tasks. We have shown how deep generative models can be integrated with BayesOpt to address these challenges, achieving good sample efficiency and solution quality across a range of design tasks. Moreover, we introduced a new large-molecule task, which provides a challenging and realistic benchmark with which to evaluate new methods for biological sequence design. Finally we successfully optimized red fluorescent proteins *in vitro*, and showed how characterizing the Pareto frontier can lead to useful, interpretable scientific insights.

In the short term, we are excited to combine LaMBO with large specialized pretrained generative models for antibodies (Ruffolo et al., 2021). The selection of mutation sites in the initialization procedure in Section 4.2 could also be improved beyond uniform random sampling. In the longer term, there are also many promising developments in BayesOpt methodology that have yet to be explored for sequence design, such as non-myopic acquisition functions (Jiang et al., 2020), multi-fidelity acquisition functions to determine the type of experimental assay and number of replications (Kandasamy et al., 2017; Wu et al., 2019), rigorous treatment of design constraints (Eriksson et al., 2019), and the coordination of many parallel drug development campaigns by optimizing risk measures across a whole compound portfolio (Cakmak et al., 2020). BayesOpt with multimodal inputs is a particularly exciting direction, allowing scientists to combine many different sources of experimental data, including 3D structure and raw instrument output (Jin et al., 2021). BayesOpt itself can also be developed for greater resilience to model misspecification, miscalibration, and covariate shift, all of which are important in drug design tasks. Finally, our work also the highlights the need for better benchmarks to evaluate drug design methods.

While there are still many challenges to overcome, there is a path for Bayesian optimization to revolutionize drug design, profoundly improving our lives, and changing the way we approach scientific discovery.

**Acknowledgements.** We would like to thank Sait Cakmak, Andres Potapczynski and Sanyam Kapoor for helpful discussions. This research is supported by an Amazon Research Award, Facebook Research, Google Research, NSF I-DISRE 193471, NIH R01DA048764-01A1, NSF IIS-1910266, and NSF 1922658 NRT-HDR.

## References


Alvarez, M. A., Rosasco, L., and Lawrence, N. D. Kernels for vector-valued functions: A review. *arXiv preprint arXiv:1106.6251*, 2011.

Angermueller, C., Belanger, D., Gane, A., Mariet, Z., Dohan, D., Murphy, K., Colwell, L., and Sculley, D. Population-based black-box optimization for biological sequence design. In *International Conference on Machine Learning*, pp. 324–334. PMLR, 2020a.

Angermueller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., and Colwell, L. Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=HklxbgBKvr.

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html.

Beck, D. and Cohn, T. Learning kernels over strings using gaussian processes. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 67–73, 2017.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M., and Church, G. M. Low-n protein engineering with data-efficient deep learning. *Nature Methods*, 18(4):389–396, 2021.

Blank, J. and Deb, K. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

Bonilla, E. V., Chai, K., and Williams, C. Multi-task gaussian process prediction. *Advances in neural information processing systems*, 20, 2007.

Brochu, E., Cora, V. M., and De Freitas, N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Cakmak, S., Astudillo Marban, R., Frazier, P., and Zhou, E. Bayesian optimization of risk measures. *Advances in Neural Information Processing Systems*, 33:20130–20141, 2020.

Chithrananda, S., Grand, G., and Ramsundar, B. Chemberta: Large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Chudakov, D. M., Matz, M. V., Lukyanov, S., and Lukyanov, K. A. Fluorescent proteins and their applications in imaging living cells and tissues. *Physiological Reviews*, 90(3):1103–1163, 2010. doi: 10.1152/physrev.00038.2009. URL https://doi.org/10.1152/physrev.00038.2009. PMID: 20664080.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

Coley, C. W., Eyke, N. S., and Jensen, K. F. Autonomous discovery in the chemical sciences part ii: outlook. *Angewandte Chemie International Edition*, 59(52):23414–23436, 2020.

Dance, A. The hunt for red fluorescent proteins. *Nature*, 596:152–153, August 2021. (Online) https://doi.org/10.1038/d41586-021-02093-6.

Daulton, S., Balandat, M., and Bakshy, E. Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

Daulton, S., Balandat, M., and Bakshy, E. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *arXiv preprint arXiv:2105.08195*, 2021a.

Daulton, S., Eriksson, D., Balandat, M., and Bakshy, E. Multi-objective bayesian optimization over high-dimensional search spaces. *arXiv preprint arXiv:2109.10964*, 2021b.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Deshwal, A. and Doppa, J. Combining latent space and structured kernels for bayesian optimization over combinatorial spaces. *Advances in Neural Information Processing Systems*, 34, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Emmerich, M. Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels. *dissertation, Universität Dortmund*, 2005.

Emmerich, M. T., Deutz, A. H., and Klinkenberg, J. W. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 2147–2154. IEEE, 2011.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local bayesian optimization. *Advances in Neural Information Processing Systems*, 32:5496–5507, 2019.

Ertl, P. and Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):1–11, 2009.

Frazier, P. I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Fu, T., Xiao, C., Li, X., Glass, L. M., and Sun, J. Mimosa: Multi-constraint molecule sampling for molecule optimization. *arXiv preprint arXiv:2010.02318*, 2020.

Gao, W., Mercado, R., and Coley, C. W. Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. *arXiv preprint arXiv:2110.06389*, 2021.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in Neural Information Processing Systems*, 31:7576–7586, 2018.

Gligorijevic, V., Berenberg, D., Ra, S., Watkins, A., Kelow, S., Cho, K., and Bonneau, R. Function-guided protein design by deep manifold sampling. *bioRxiv*, 2021.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Goovaerts, P. et al. *Geostatistics for natural resources evaluation*. Oxford University Press on Demand, 1997.

Grosnit, A., Tutunov, R., Maraval, A. M., Griffiths, R.-R., Cowen-Rivers, A. I., Yang, L., Zhu, L., Lyu, W., Chen, Z., Wang, J., et al. High-dimensional bayesian optimisation with variational autoencoders and deep metric learning. *arXiv preprint arXiv:2106.03609*, 2021.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

Høie, M. H., Cagiada, M., Frederiksen, A. H. B., Stein, A., and Lindorff-Larsen, K. Predicting and interpreting large scale mutagenesis data using analyses of protein stability and conservation. *bioRxiv*, 2021. doi: 10.1101/2021.06.26.450037. URL https://www.biorxiv.org/content/early/2021/06/28/2021.06.26.450037.

Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C. W., Xiao, C., Sun, J., and Zitnik, M. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*, 2021.

Jensen, J. H. 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.

Jiang, S., Jiang, D., Balandat, M., Karrer, B., Gardner, J., and Garnett, R. Efficient nonmyopic bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33:18039–18049, 2020.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Jin, W., Wohlwend, J., Barzilay, R., and Jaakkola, T. Iterative refinement graph neural network for antibody sequence-structure co-design. *arXiv preprint arXiv:2110.04624*, 2021.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kandasamy, K., Dasarathy, G., Schneider, J., and Póczos, B. Multi-fidelity bayesian optimisation with continuous approximations. In *International Conference on Machine Learning*, pp. 1799–1808. PMLR, 2017.

Khan, A., Cowen-Rivers, A. I., Deik, D.-G.-X., Grosnit, A., Dreczkowski, K., Robert, P. A., Greiff, V., Tutunov, R., Bou-Ammar, D., Wang, J., et al. Antbo: Towards real-world automated antibody design with combinatorial bayesian optimisation. *arXiv preprint arXiv:2201.12570*, 2022.

Krenn, M., Häse, F., Nigam, A., Friederich, P., and Aspuru-Guzik, A. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Lambert, T. J. Fpbase: a community-editable fluorescent protein database. *Nature methods*, 16(4):277–278, 2019.

Lázaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 2010.

Lee, J., Mansimov, E., and Cho, K. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*, 2018.

Letham, B., Karrer, B., Ottoni, G., and Bakshy, E. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

Lipinski, C. A., Lombardo, F., Dominy, B. W., and Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews*, 23(1-3):3–25, 1997.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444, 2002.

Lyu, J., Wang, S., Balius, T. E., Singh, I., Levit, A., Moroz, Y. S., O'Meara, M. J., Che, T., Algaa, E., Tolmachova, K., et al. Ultra-large library docking for discovering new chemotypes. *Nature*, 566(7743):224–229, 2019.

Maddox, W., Feng, Q., and Balandat, M. Optimizing high-dimensional physics simulations via composite bayesian optimization. *arXiv preprint arXiv:2111.14911*, 2021a.

Maddox, W. J., Balandat, M., Wilson, A. G., and Bakshy, E. Bayesian optimization with high-dimensional outputs. *Advances in Neural Information Processing Systems*, 34, 2021b.

Maddox, W. J., Stanton, S., and Wilson, A. G. Conditioning sparse variational gaussian processes for online decision-making. *Advances in Neural Information Processing Systems*, 34, 2021c.

Maus, N., Jones, H. T., Moore, J. S., Kusner, M. J., Bradshaw, J., and Gardner, J. R. Local latent space bayesian optimization over structured inputs. *arXiv preprint arXiv:2201.11872*, 2022.

Meier, J., Rao, R., Verkuil, R., Liu, J., Sercu, T., and Rives, A. Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*, 2021.

Mishra, A., Ranganathan, S., Jayaram, B., and Sattar, A. Role of solvent accessibility for aggregation-prone patches in protein folding. *Scientific Reports*, 8(1):12896, 2018. doi: 10.1038/s41598-018-31289-6. URL https://doi.org/10.1038/s41598-018-31289-6.

Moss, H. B., Beck, D., Gonzalez, J., Leslie, D. S., and Rayson, P. Boss: Bayesian optimization over string spaces. *arXiv preprint arXiv:2010.00979*, 2020.

Nigam, A., Friederich, P., Krenn, M., and Aspuru-Guzik, A. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. *arXiv preprint arXiv:1909.11655*, 2019.

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. The uncertainty bellman equation and exploration. In *International Conference on Machine Learning*, pp. 3836–3845, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, X., Canny, J., Abbeel, P., and Song, Y. S. Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32:9689, 2019.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass., 3. print edition, 2008. ISBN 978-0-262-18253-9.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

Ruffolo, J. A., Gray, J. J., and Sulam, J. Deciphering antibody affinity maturation with language models and weakly supervised learning. *arXiv preprint arXiv:2112.07782*, 2021.

Schymkowitz, J., Borg, J., Stricher, F., Nys, R., Rousseau, F., and Serrano, L. The foldx web server: an online force

field. *Nucleic acids research*, 33(suppl_2):W382–W388, 2005.

Shah, A. and Ghahramani, Z. Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, pp. 1919–1927. PMLR, 2016.

Shahriari, B., Bouchard-Côté, A., and Freitas, N. Unbounded bayesian optimization via regularization. In *Artificial intelligence and statistics*, pp. 1168–1176. PMLR, 2016.

Shrake, A. and Rupley, J. A. Environment and exposure to solvent of protein atoms. lysozyme and insulin. *Journal of molecular biology*, 79(2):351–371, 1973.

Shu, R., Lee, J., Nakayama, H., and Cho, K. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8846–8853, 2020.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1015–1022, 2010.

Tripp, A., Daxberger, E., and Hernández-Lobato, J. M. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33, 2020.

Tripp, A., Simm, G. N., and Hernández-Lobato, J. M. A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*, 2021.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., and Guyon, I. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv preprint arXiv:2104.10201*, 2021.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., and de Feitas, N. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

Wouters, O. J., McKee, M., and Luyten, J. Estimated research and development investment needed to bring a new medicine to market, 2009-2018. *Jama*, 323(9):844–853, 2020.

Wu, J., Toscano-Palmerin, S., Frazier, P. I., and Wilson, A. G. Practical multi-fidelity bayesian optimization for hyperparameter tuning. *arXiv pre-print arXiv:1903.04703*, 2019.

Yang, K. K., Chen, Y., Lee, A., and Yue, Y. Batched stochastic bayesian optimization via combinatorial constraints design. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3410–3419. PMLR, 2019a.

Yang, K. K., Wu, Z., and Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8):687–694, 2019b.

Zhang, D., Fu, J., Bengio, Y., and Courville, A. Unifying likelihood-free inference with black-box sequence design and beyond. *arXiv preprint arXiv:2110.03372*, 2021.

# Appendices

In Appendix A we outline the four evaluation tasks: **Bigrams**, **logP + QED**, **DRD3 docking + SA**, and **Stability + SASA**. In Appendix B, we describe in detail the neural network architecture, the DKL GP implementation, the denoising autoencoder implementation, the string kernel implementation, and the hyperparameters used for our experiments. Finally, in Appendix C, we present additional experimental results to supplement the figures in the main text.

## A. Evaluation Task Details

### A.1. Bigrams Task

The bigrams task is a simple toy example of discrete sequence optimization. We draw random strings from an alphabet $\mathcal{V}$ and count the occurrence of $k$ predetermined bigrams, which we use as proxy fitness targets. The task is to maximize the counts of each bigram in the sequence, restricting the sequence length to 36 tokens including utility tokens ("[PAD]", "[CLS]", "[SEP]", "[UNK]", "[MASK]"). For our experiments we used the same amino acid vocabulary as our protein task and chose 3 complementary bigrams, "AV", "VC" and "CA". The initial sequences were sampled with lengths between 32 and 36 tokens We ensured there were an equal number of positive examples (sequences with at least one occurrence of one of the bigrams) as negative examples in the starting pool.

### A.2. logP + QED Task

The original ZINC logP optimization task, popularized in the BayesOpt community by Gómez-Bombarelli et al. (2018), is to optimize the octanol-water partition coefficient of a small molecule. Molecules with high logP values are hydrophobic and molecules with low values are hydrophilic. Hydrophobicity can be desirable for absorption and solubility, for example in pharmaceuticals. As a property that is easy to calculate, it has risen to prominence despite being undesirable on its own. Very high logP can result in molecules with limited practical application, and moreover finding molecules with high logP reduces trivially to the problem of finding long hydrocarbon chains, as these compounds are extremely hydrophobic relative to the size of the molecule. The *penalized* logP objective adds auxiliary terms measuring synthetic accessibility (Ertl & Schuffenhauer, 2009) and the number of cycles. Unfortunately these terms do not fix the underlying problem, and so penalized logP is similarly vulnerable to optimization hacking, as we discuss in Section C.2.

Because logP in itself is a deeply flawed objective, both in its relevance to real-world drug design and its ability to be hacked by optimizers, we also consider a multi-objective optimization task that is closer in form to real design problems. Instead of solely optimizing for logP, we jointly optimize for logP and QED (Quantitative Estimate of Druglikeness), a composite metric that captures many elements of druglikeness, with bioaavailability among the most prominent (Bickerton et al., 2012). Unfortunately QED has its own limitations as an objective, since it is a simple parametric model trained on a small dataset to emulate heuristics such as Lipinski's Rule of Five (Lipinski et al., 1997).

The shortcomings of objectives like logP and QED appear to be well-known (Nigam et al., 2019; Coley et al., 2020; Fu et al., 2020; Tripp et al., 2021; Maus et al., 2022), but superior alternatives have not yet been accepted by the research community. For example, at the time of writing the only molecule generation benchmark in TorchDrug is maximization of QED and logP of ZINC-like molecules.[2] Angermueller et al. (2020a) evaluated BBO algorithms on a substantial number of *in silico* sequence design tasks, however the large molecule tasks they considered were relatively simple, single-objective problems (e.g. maximization of the likelihood of a hidden Markov model). The vacuum of rigorous *in silico* evaluation tasks for large-molecule design motivated us to propose our RFP task as a new benchmark.

We construct the start pool by inverting the scores and selecting the top-$k$ non-dominating sequences (i.e. we found the $k$ *most* dominated sequences in the ZINC dataset w.r.t. logP and QED). Constructing the task in this way is better than simply sampling randomly from ZINC because QED is bounded above by 1 and many ZINC sequences already score fairly close to 1. Starting with dominated sequences ensures that there is sufficient headroom for improvement to observe variations in optimizer behavior. We capped the max sequence length at 128 SELFIES tokens, including utility tokens. The SELFIES vocabulary was precomputed from the entire ZINC dataset (Krenn et al., 2020).

[2] https://torchdrug.ai/docs/benchmark/generation.html

### A.3. DRD3 Docking + SA Task

We use the DRD3 docking score oracle from Huang et al. (2021), and the following quotation is reproduced from https://tdcommons.ai/benchmark/docking_group/overview:

" Docking is a theoretical evaluation of affinity between a ligand (a small molecular drug) and a target (a protein involved in the disease). As a molecule with higher affinity is more likely to have higher bioactivity, docking is widely used for virtual screening of compounds (Lyu et al., 2019). "

Because optimizing solely for docking may produce molecules that are difficult to synthesize, we also optimize for synthetic accessibility (SA) (Ertl & Schuffenhauer, 2009).

To construct the start pool for this task we selected 512 molecules from ZINC uniformly at random and labeled them with the objective oracles. We then used the same start pool and SELFIES encodings for all experimental trials and all optimization methods.

### A.4. Stability + SASA Task

In this work we present a new *in silico* benchmark task designed to simulate searching for improved red fluorescent protein (RFP) variants *in vitro*, a problem of significant interest to biomedical researchers (Dance, 2021). We optimize red-spectrum proteins with known structures for stability (-dG or negative change in Gibbs free energy) and solvent-accessible surface area (SASA) (Shrake & Rupley, 1973; Cock et al., 2009) in simulation, using the FoldX suite (Schymkowitz et al., 2005) and BioPython to evaluate our objective function. Stability as evaluated by FoldX—particularly in the negative case—has been shown to correlate with protein function (Høie et al., 2021). Solvent-accessible surface area will correlate with factors that influence the brightness and photostability of the fluorescent protein: aggregation propensity due to exposed hydrophobic surface (Mishra et al., 2018) and shielding by the beta-barrel, which encapsulates the fluorophore (Chudakov et al., 2010). Since both of these benchmark tasks are functions of the protein's three-dimensional structure, it is expected that training a model on these tasks will require the model to learn a latent representation for structure, which in turn determines function.

We constructed the start pool in two phases. First we searched FPBase for all red-spectrum (defined in this context as having an emission wavelength at least 580 nm) proteins with at most 244 residues with known 3D structures, selecting the highest resolution structure if more than one was available. If more than one chain was present in the structure, we selected the longest chain as the representative residue sequence. Starting from these base proteins, we used NSGA-2 to collect additional labelled sequences to use in the start pool for subsequent experiments.

Although this task is a significant step forward for *in silico* evaluation of discrete sequence design, it is currently limited by the capabilities of FoldX, which can only compute structures from substitution mutations (i.e. the sequence length cannot change). Deep learning structure oracles such as AlphaFold (Jumper et al., 2021) or RoseTTAFold (Baek et al., 2021) could also be used, but we found FoldX to be simpler and more amenable for rapid prototyping.

### A.5. Wet lab experimental procedure

We synthesized fluorescent proteins with PURExfrex 2.1 (Cosmo Bio LTD) in 50 uL reactions from linear DNA purchased from IDT as eBlock dsDNA gene fragments. We ran reactions at 30°C overnight in black, half-area microplates (Corning #3993) with optically clear plate adhesive and measured excitation and emission through a series of sweeps (fixing the excitation wavelength and scanning emission every 1–2 nm, or vice versa). We determined peak excitation and peak emission as the wavelength that gave maximum fluorescence units. Using NanoDSF on an Uncle instrument (Unchained Labs), we measured protein thermostability as $T_m$, defined as the midpoint transition temperature of the thermal melt curve.

## B. Implementation Details

Our models are implemented in PyTorch (Paszke et al., 2019), BoTorch (Balandat et al., 2020), and GPyTorch (Gardner et al., 2018). Our genetic optimizer baselines are implemented in PyMOO (Blank & Deb, 2020). Our code is publicly available at https://github.com/samuelstanton/lambo. Hyperparameters are summarized in Appendix B.4.

### B.1. Architecture Details

We used the same base architecture for all experiments, relying on 1D convolutions (masking positions corresponding to padding tokens). We used standard pre-activation residual blocks with two conv layers, layernorm, and swish activations. We used a kernel size of 5, 64 intermediate channels and 16 latent channels.

The shared encoder and decoder each were composed of 3 of these residual blocks (for a total of 6 convolutional layers each). The shared encoder embeds input sequences with standard vocabulary and sinusoidal position embeddings. The discriminative encoder was composed of a single residual block.

Note that transformer encoder layers could be substituted as a drop-in replacement for these convolutional residual blocks, we used small convolutional layers because they are fast to train and performed adequately in our experiments.

For multi-task GPs, we chose ICM kernels for their efficiency, particularly in sampling (Bonilla et al., 2007; Maddox et al., 2021b), their flexibility and popularity.

### B.2. DKL Implementation Details

Training DKL models is an art. Some best practices apply to both stochastic variational and exact GP inference, others are specific to the former.

**Applicable to exact and variational GP inference**

1. *Kernel hyperparameter priors matter.* Allowing the DKL GP to easily change both the inputs to the final conventional GP kernel (e.g. RBF) and the lengthscale of that kernel doesn't work well. We placed a tight Gaussian prior ($\sigma = 0.01$) around the initial lengthscale value and forced the encoder to learn features appropriate for that lengthscale. Note that this is distinct from simply fixing the kernel hyperparameters *a priori*.

2. *Optimizer hyperparameters matter.* Adam is really convenient to avoid too much learning rate tuning, but it can cause unexpected issues when jointly training supervised and unsupervised heads. We almost completely disabled the running estimates of the first two moments in Adam, using $\beta_1 = 0.$, and $\beta_2 = 0.01$.

3. *Normalization matters.* This is more of an issue for SVGPs than exact GPs, but in both cases batchnorm can cause undesirable and unexpected behavior. Use layernorm.

**Applicable to variational GP inference**

1. *Initialization matters.* We use the procedure described in Maddox et al. (2021c) to reinitialize the inducing point locations and the variational parameters every time the model was retrained. This trick significantly improves results and saves computation, since the GP training does not completely start over every outer loop iteration.

2. One final trick that is very useful for SVGPs is to turn off gradients to all GP-related parameters every other epoch (so half the epochs are only train the encoder).

As we show in the main text and Figure 9, DKL SVGPs can consistently be trained to similar levels of accuracy as exact DKL GPs with very little trouble, once the proper training procedures are in place. With these practical insights we were able to jointly train supervised GP heads and unsupervised language model heads on a shared encoder simply by taking one gradient step on the supervised GP loss and one gradient step on the unsupervised DAE loss per minibatch, using the same optimizer and learning rate schedule. We used diagonal Gaussian likelihoods for all our experiments, with the noise variance initialized at $0.25$.

We found that DKL GPs (both exact and variational) were not immune to overfitting, so we used weight decay (1e-4) and reserved 10% of all collected data (including online queries) as validation data for early stopping.

### B.3. DAE Implementation Details

**MLM Head** We used a mask ratio of $0.125$ for all experiments when training MLM heads. The MLM loss is computed by randomly masking input tokens, and computing the empirical cross-entropy between the original sequence and the predictive

distribution of the MLM head at the masked positions. During sequence optimization the MLM predictive distribution is modified to prevent sampling the original token (to encourage diversity) and to prevent the sampling of special tokens.

**LANMT Head** Our LANMT head is identical to our MLM head, except for the addition of a length prediction head and length transform module (Shu et al., 2020), a different corruption procedure and training objective. We used a max length change of 8 in our experiments for Figure 10, so the corruption function randomly sampled a length change $\Delta t$ between -8 and 8. $\Delta t$ tokens were subsequently deleted, replaced, or inserted into the sequence. The corrupted sequence was forwarded through the model, which was also given the original sequence length as a label during training. A training step takes a gradient step on the cross-entropy between the predicted length and the actual length, and on the cross-entropy between the predictive distribution over the whole decoded sequence and the original sequence.

### B.4. Hyperparameters

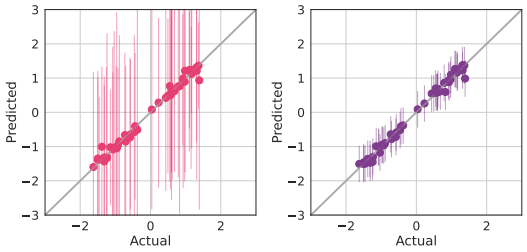
| Sequence Optimization | |
|---|---|
| **Name** | **Value** |
| $\lvert\mathcal{D}_0\rvert$ | 512 |
| Query batch size ($b$) | 16 |
| $\lvert\mathcal{X}_{\text{base}}\rvert$ | $b$ |
| # Optimization rounds ($i_{\max}$) | 64 |
| # Inner loop restarts | 16 |
| # Inner loop gradient steps ($j_{\max}$) | 32 |
| Inner loop step size ($\eta$) | 0.1 |
| Entropy penalty ($\lambda$) | 1e-2 |
| # MC acquisition samples | 2 |
| Random seeds | $\{0, \ldots, 9\}$ |

| DAE Architecture | |
|---|---|
| **Name** | **Value** |
| Shared enc. depth (# residual blocks) | 3 |
| Disc. enc. depth (# residual blocks) | 1 |
| Decoder depth (# residual blocks) | 3 |
| Conv. kernel width (# tokens) | 5 |
| # conv. channels | 64 |
| Latent dimension | 16 |
| GP likelihood variance init | 0.25 |
| GP lengthscale prior | $\mathcal{N}(0.7, 0.01)$ |
| # inducing points (SVGP head) | 64 |

| DAE Training | |
|---|---|
| **Name** | **Value** |
| DAE corruption ratio (training) | 0.125 |
| DAE learning rate (MTGP head) | 5e-3 |
| DAE learning rate (SVGP head) | 1e-3 |
| DAE weight decay | 1e-4 |
| Adam EMA params ($\beta_1, \beta_2$) | (0., 1e-2) |
| Early stopping holdout ratio | 0.1 |
| Early stopping relative tolerance | 1e-3 |
| Early stopping patience (# epochs) | 32 |
| Max # training epochs | 256 |

# C. Additional Results

## C.1. What About Substring Kernels?




(a) SSK GP (b) DKL GP

*Figure 8.* Evaluating the effect of kernel choice on exact GP regression with a discrete SSK **(left)** and a deep Matérn kernel with a CNN encoder **(right)** when predicting the SASA property of RFP large molecules. The SSK GP is under-confident and less accurate than the DKL GP, suggesting SSK GPs would not improve optimization performance.


*Figure 9.* Negative log-likelihood (NLL) and root mean squared error (RMSE) for various discriminative models in the offline regression setting. DKL MTGPs and SVGPs have good performance across the board, while bootstrapped CNN ensembles (DE) are very overconfident. Exact GP inference with a substring kernel (SK) is very underconfident and has poor accuracy when predicting logP and QED.

Since the start pools used in the evaluation in Figure 3 are fairly small (i.e. 512 examples), it is natural to wonder how a substring kernel (SSK) GP (e.g., Moss et al., 2020)) would compete with our DKL-based GPs. Due to constraints on time and computation, and SSK scalability issues, we do not directly compare SSK GPs and DKL GPs in the online setting. However, in Figure 8, we compare SSK GPs and DKL GPs in the offline regression setting, training both models only through the supervised loss to predict the SASA property of RFP large molecules, using 410 examples for training and 102 examples for validation and test. The SSK GP performs fairly well, but is very under-confident when compared to the DKL GP. At an empirical level, our results do not support the claim that SSK GPs are superior models for biological sequences. SSKs have further drawbacks which make it hard to justify additional investment into SSK GPs for drug design.

**Lack of Positional Information:** at a conceptual level, SSKs cannot identify regions of the sequence that have little effect on the objective values, since matching substrings increase the prior covariance between sequences regardless of where the substrings are found. Simply put, an SSK just counts the occurrences of every possible $n$-gram across every possible combination of $n$ positions in a sequence. The gap decay hyperparameter downweights occurrences corresponding to position combinations with elements that are not closely colocated. Hence SSKs have very limited positional awareness in the sense that sequences with similar $n$-gram counts have high prior covariance, regardless of where the $n$-grams actually occurred. Positional awareness is important when dealing with biological sequences from some subpopulation (e.g. a family of fluorescent proteins) since they have many identical subsequences, only varying at positions that strongly affect function.

**Difficult to Scale:** at a practical level, SSKs are difficult to integrate with deep generative models since they do not operate on continuous embeddings (Nigam et al., 2019), and standard methods for scaling GPs — inducing point methods (e.g., Hensman et al., 2013), random feature expansions (e.g., Lázaro-Gredilla et al., 2010), and CG-based methods (e.g., Gardner

et al., 2018) — are difficult to apply. In particular inducing points methods are impractical because the inducing point domain would be the discrete input space, introducing a challenging discrete optimization subproblem just to train the surrogate. Furthermore SSKs struggle to scale not just to large datasets, but also to long sequences. The dynamic programming algorithm used by Moss et al. (2020) to compute their SSK is parallelizable, but becomes prohibitively memory intensive for sequences longer than 100 tokens, even when chunking the sequence into smaller pieces. In fact, we used an Nvidia RTX 8000 GPU with 48 GB of memory just to produce Figure C.1. We also implemented a memory-efficient trie-based SSK, which could handle longer sequences but was prohibitively slow and difficult to parallelize.

### C.2. Comparison to LSBO for single-objective BBO

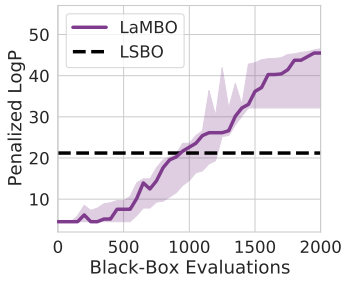



*Figure 10.* LaMBO reaches a higher objective value than the best reported LSBO result. When constrained to 500 evaluations LSBO is more sample efficient (ignoring pretraining data). The midpoint, lower, and upper bounds of the LaMBO curve depict the 60%, 40%, and 80% quantiles, estimated from 5 trials.

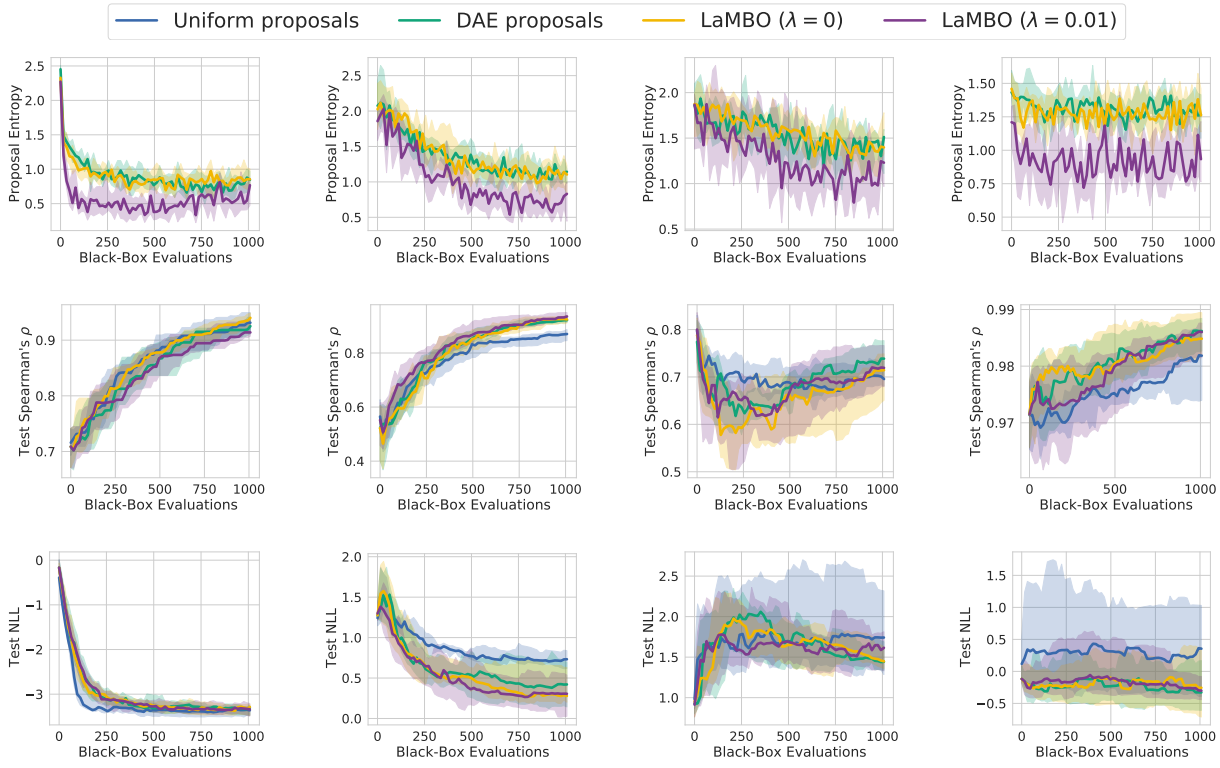
Here we evaluate LaMBO in the single-objective setting, using the *penalized* logP task described in Tripp et al. (2020). In contrast to SSK-based methods, LaMBO can successfully be scaled to large datasets with standard variational GP inference, allowing us to compare to the popular latent space BayesOpt approach (LSBO) (Gómez-Bombarelli et al., 2018) on a larger-scale problem. The start pool for this task is composed of the 2000 highest scoring sequences from ZINC, and 8000 random sequences, replicating the setup in Tripp et al. (2020). To accommodate the larger dataset, the discriminative head uses $k$ independent stochastic variational GPs (SVGPs) with 64 shared inducing points rather than an exact MTGP.

In Figure 10 we demonstrate that LaMBO is competitive with a variant of LSBO specifically designed for this task, requiring about twice as many online observations before reaching the reported median best score attained by LSBO (Tripp et al., 2020). As noted in Section 3, LSBO uses the entire ZINC dataset for pretraining, so we do not directly compare sample efficiency. In addition to the differences between LaMBO and LSBO already noted, we use SELFIES encodings rather than SMILES. We use a seq2seq LANMT-style decoder head for this task, since logP heavily favors large molecules. High-scoring molecules such as those found by LSBO are larger than any found in ZINC, so it is important that the optimizer allow insertions.

Overall, LaMBO outperforms the best reported LSBO score (27.84) by a wide margin, reaching scores as high as 50 for some seeds, while using a more general architecture and requiring less data (counting both pretraining data and online queries). The factor that ultimately bounds the penalized logP objective in practice is the max sequence length constraint imposed by the positional encoding scheme we use. Therefore we note that, despite its widespread use, unconstrained logP (penalized or otherwise) is a poor optimization benchmark, since it can be manipulated by altering the positional encoding to permit longer sequences (Nigam et al., 2019; Tripp et al., 2021; Fu et al., 2020).

In short, while LaMBO is designed to facilitate multi-objective optimization — a central feature of drug design — it can also outperform the widely used single-objective LSBO, even in a single-objective setting.

### C.3. Ablating LaMBO: Proposal Entropy and Discriminative Performance




**(a) Bigrams** **(b) logP + QED** **(c) DRD3 docking + SA** **(d) Stability + SASA**

*Figure 11.* Additional metrics for the ablation experiment described in Section 5.3 and Figure 4, where we cumulatively add the elements described in Section 4.3: (1) DAE-generated proposals, (2) DAE proposal optimization following $\nabla_Z[\ell_{\text{query}}]$ with $\lambda = 0$. (see Eq. (3)), and (3) DAE proposal optimization with $\lambda = 0.01$. The **top** row shows the average entropy of the generative DAE proposal distributions over time. As expected, the entropy penalty decreases the proposal entropy. The **middle** and **bottom** rows show the discriminative Spearman's $\rho$ and NLL (averaged across objectives) on heldout data over time. Training the LaMBO architecture with both the unsupervised DAE objective and the supervised GP objective improves discriminative performance compared to the same model trained only through the supervised objective. Otherwise the methods behave similarly, verifying that better solutions are the result of better proposals, rather than a better discriminative model. The midpoint, lower, and upper bounds of each curve depict the 50%, 20%, and 80% quantiles, estimated from 10 trials.