UNIFYING LIKELIHOOD-FREE INFERENCE WITH BLACK-BOX SEQUENCE DESIGN AND BEYOND

Anonymous authors
Paper under double-blind review

ABSTRACT

Black-box optimization formulations for biological sequence design have drawn recent attention due to their promising potential impact on the pharmaceutical industry. In this work, we propose to unify two seemingly distinct worlds: likelihood-free inference and black-box sequence design, under one probabilistic framework. In tandem, we provide a recipe for constructing various sequence design methods based on this framework. We show how previous drug discovery approaches can be "reinvented" in our framework, and further propose new probabilistic sequence design algorithms. Extensive experiments illustrate the benefits of the proposed methodology.

1 INTRODUCTION

Discovering new drugs to fulfill specific criteria, such as binding affinity towards a given molecular target, is a fundamental problem in chemistry and the pharmaceutical industry (Hughes et al., 2011). In this work, we focus on an important subdomain: *de novo* biological sequence design. This task is challenging for two reasons: (1) the exploration space for sequences is combinatorially large; and (2) sequence usefulness is evaluated via a complicated process which usually involves time-consuming and expensive wet-lab experiments.

Despite the difficulty of this task, many approaches have been developed over the past few decades thanks to recent advances in biochemistry and machine learning. The Nobel Prize wining paradigm, *directed evolution* (Chen & Arnold, 1991), which conducts local evolutionary search under human guidance, is one of the popular techniques. Unfortunately, it is limited by its sample inefficiency and reliance on strong prior knowledge, *e.g.*, about where to mutate (Ahn et al., 2020). Furthermore, to compete with other machine learning methods (Gottipati et al., 2020), guided evolution (Yoshikawa et al., 2018; Jensen, 2019; Nigam et al., 2019) heavily relies on human intuition for designing domain-specific evolutionary operators, which may not always apply to tasks at hand.

In this work, we deem sequence design to be a black-box optimization problem, tasked with maximizing an unknown oracle function. We assume that oracle queries are limited due to the constraint on resources, such as the budgets for evaluating queries in a wet-lab. Thus, sample efficiency is crucial. We develop a probabilistic framework by reformulating the aforementioned black-box optimization target as a posterior modeling problem. With this framework, we draw a surprising connection between likelihood-free inference and sequence design, and thus linking two fields which are previously considered as unrelated. The key observation we leverage here for establishing this connection is that both settings share similar elements and targets which will be elaborated in Section 2.2. This connection facilitates our understanding of both fields and provides a recipe for developing sequence design algorithms. Going beyond, we also combine different probabilistic modeling insights and develop three novel *composite* probabilistic algorithms. We point out that our framework could actually be applied to any black-box optimization settings, but in this work we focus on its application to biological sequence design.

To demonstrate the empirical effectiveness of our methods, we conduct systematical experiments to evaluate their performance on four *in-silico* sequence design benchmarks. Our proposed methods achieve at least comparable results to existing baselines, and the proposed composite methods behave consistently better than all other ones across various sequence design tasks.

We summarize our contribution as follows:

- We develop a probabilistic framework that unifies likelihood-free inference and black-box sequence design.
- Based on this framework, we provide a recipe for designing algorithms for black-box problems. We apply these ideas to propose a series of composite design algorithms.
- We perform systematical evaluation and find that these algorithms achieves consistently better or comparable results compared to previous ones, thus illustrating the benefit of the proposed unified framework.

2 A Unifying Probabilistic Framework

2.1 Background

Likelihood-free inference (LFI). We use $\boldsymbol{\theta} \in \Theta$ and $\mathbf{x} \in \mathcal{X}$ to separately denote the parameters and the data generated via the mechanism $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$. In this scenario, LFI refers to a special kind of Bayesian inference setting where the likelihood function is not tractable but sampling (by simulation) from the likelihood is feasible. Consider the objective of modeling the Bayesian posterior when we cannot compute the likelihood $p(\mathbf{x}_o|\boldsymbol{\theta})$:

$$p(\boldsymbol{\theta}|\mathbf{x}_o) \propto p(\boldsymbol{\theta}) \underbrace{p(\mathbf{x}_o|\boldsymbol{\theta})}_{?}, \tag{1}$$

where $\mathbf{x}_o$ is the observed data, $p(\boldsymbol{\theta})$ is the (given) prior over the model parameters $\boldsymbol{\theta}$, $p(\mathbf{x}|\boldsymbol{\theta})$ is the intractable likelihood function and $p(\boldsymbol{\theta}|\mathbf{x})$ is the desired posterior over $\boldsymbol{\theta}$. While we do not have access to the exact likelihood, we can still simulate (sample) data $\mathbf{x}$ from the model simulator: $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$. Instead of trying to obtain a numerical value of the generic posterior $p(\boldsymbol{\theta}|\mathbf{x})$ for arbitrary $\mathbf{x}$, LFI only tries to obtain an approximation of $p(\boldsymbol{\theta}|\mathbf{x}_o)$ for the given $\mathbf{x}_o$. During the inference process, we can take advantage of the sampled data: $\mathcal{D} = \{(\boldsymbol{\theta}_i, \mathbf{x}_i)\}_{i=1}^n$ where $\mathbf{x}_i \sim p(\mathbf{x}|\boldsymbol{\theta}_i)$ for selected values of $\boldsymbol{\theta}_i$.

Biological black-box sequence design. We consider biological sequence design as a black-box optimization problem:

$$\mathbf{m}^* = \arg\max_{\mathbf{m} \in \mathcal{M}} f(\mathbf{m}),$$

where $f(\cdot)$ is the oracle score function, and we would like to discover values of $\mathbf{m}$ for which $f(\mathbf{m})$ is large. In real-world situations, a query of this oracle f could represent a series of wet-lab experiments to measure specific chemical properties or specificity for a given binding site target. In general, these experiments are time- and cost-consuming. As a result, the total number of queries is limited.

In our setting, we use $\mathcal{M} = \mathcal{V}^L$ to denote the search space for sequences with fixed length L, where $\mathcal{V}$ is the vocabulary for each entry of the sequence: for DNA nucleotides $|\mathcal{V}| = 4$, and for protein amino acids $|\mathcal{V}| = 20$. For variable length setting, we have $\mathcal{M} = \cup_{L \in [L_{\text{min}}, L_{\text{max}}]} \mathcal{V}^L$, where L_{min} and L_{max} are the minimal and maximal length, respectively.

2.2 Connecting LFI and sequence design

In order to draw a connection to LFI, we require a probabilistic formulation of the black-box sequence design problem. To this end, we relax the goal of searching for a single maximum of the oracle / score function f to a posterior modeling problem, *i.e.*, finding a representative sample of the configurations of $\mathbf{m}$ sampled with probability related to some target posterior. Think of $\mathcal{E}$ as a Boolean feature of these desirable configurations, and our goal is to characterize the posterior distribution $p(\mathbf{m}|\mathcal{E})$[1] from which we obtain the desired sequences. Below, we consider two specific ways of doing this:

Example A. We explicitly define $\mathcal{E}$ as true for all the sequences whose scores are larger than a given threshold s:

$$\mathcal{E} = \{\mathbf{m} | f(\mathbf{m}) \geq s\}. \tag{2}$$

[1] We slightly overuse the notation for the event $\{\mathbf{m} \in \mathcal{E}\}$ and the sequence set $\mathcal{E}$.

Here s could be any fixed value, or a certain quantile of a particular score distribution. In this way, we have $p(\mathcal{E}|\mathbf{m}) = \mathbb{1}\{f(\mathbf{m}) \geq s\}$ where $\mathbb{1}\{\}$ is the indicator function.

Example B. In a softer version of $\mathcal{E}$, we can define its conditional probability of being true to follow a Boltzmann distribution:

$$p(\mathcal{E}|\mathbf{m}) := p(\mathbf{m} \in \mathcal{E}|\mathbf{m}) \propto \exp(f(\mathbf{m})/\tau). \tag{3}$$

where τ is a temperature parameter. We introduce the exponential because $f(\cdot)$ does not necessarily take positive values. Any monotone transformation of $f(\cdot)$ to non-negative reals could be used, so that sequences with larger oracle scores have a greater probability of making $\mathcal{E}$ true.

With this posterior objective, our goal now becomes effectively modeling and sampling from the posterior $p(\mathbf{m}|\mathcal{E})$. It is thus natural to resort to the tools of Bayesian inference for this task. In order to examine this possibility, we draw a detailed comparison between the settings of black-box sequence design problem and likelihood-free Bayesian inference in Table 1.

	Likelihood-free inference	Black-box sequence design
Element	$(\boldsymbol{\theta}, \mathbf{x})$	$(\mathbf{m}, s)$
Target	$p(\boldsymbol{\theta}\|\mathbf{x}_o)$	$p(\mathbf{m}\|\mathcal{E})$
Constraint	limited simulation: $\mathbf{x} \sim p(\mathbf{x}\|\boldsymbol{\theta})$ intractable likelihood: $p(\mathbf{x}\|\boldsymbol{\theta})$	limited query: $s \sim f(\mathbf{m})$ black-box oracle: $f(\mathbf{m})$

Table 1: Correspondence between likelihood-free inference and black-box optimization.

It can be observed that both tasks share similar elements and targets. The two settings also share similar limitations on the allowed queries, which are too time-consuming and / or cost-intensive. Notice that in sequence design, the oracle could be either exact or noisy, thus we use the more general $s \sim f(\mathbf{m})$ formulation rather than $s = f(\mathbf{m})$. We will further present several concrete examples as demonstrations of this correspondence in the following section.

Another way to understand this correspondence is to consider the following mapping T:

$$\begin{aligned} T : \Theta \times \mathcal{X} &\to \mathcal{M} \times \mathbb{R} \\ (\boldsymbol{\theta}, \mathbf{x}) &\mapsto (\mathbf{m}, s), \quad \text{s.t. } s = -\left\|\mathbf{x} - \mathbf{x}_o\right\|. \end{aligned}$$

Here we can see the score value s as a quantitative metric for how close the generated data $\mathbf{x}$ (given $\boldsymbol{\theta}$) is to the target observed data $\mathbf{x}_o$. In addition, querying the oracle in the sequence design setting can also be thought of as follows: (1) sample $\mathbf{x} \sim p(\cdot|\boldsymbol{\theta})$ and then (2) calculate $s = -\left\|\mathbf{x} - \mathbf{x}_o\right\|$ under some distance $\|\cdot\|$. In this manner, T could conceptually transform any LFI problem into a black-box optimization task. In this work, we only focus on the application of sequence design.

3 METHODOLOGY

We provide a recipe for designing new sequence design algorithms based on the correspondence in Section 2.2. The recipe induces different approaches by modeling different probabilistic component of the Bayesian inference problem. We begin with common algorithm restrictions under this setting.

Common constraint for algorithms. Due to the restriction of simulation / query in our setting, we constrain our algorithms to act in a sequential / iterative way, gradually achieving the desired posterior round by round. Every algorithm starts with an empty dataset $\mathcal{D} = \varnothing$ and an initial proposal $p_1(\cdot) = p(\cdot)$, where $p(\cdot)$ is the prior given by the task. In the r-th round of this multi-round setting, the algorithm would use the proposal $p_r(\cdot)$ of this round to sample a batch of data ($\boldsymbol{\theta}$ / $\mathbf{m}$) for simulation / query, and augment the current dataset $\mathcal{D}$ with the newly obtained batch of data. We use n to denote the batch size for each round's simulation / query. Afterwards, the algorithm updates the proposal to $p_{r+1}(\cdot)$. The outcomes for the two settings we discuss may be slightly different: an algorithm for likelihood-free inference would return the posterior, while a sequence design method would return the dataset of all the sequences it has queried, which hopefully contains desired high scored sequences. On the other hand, a sequence design method could produce as an intermediate result a generative model for sampling queries, which then completely fits with the LFI framework.

3.1 BACKWARD MODELING OF THE MECHANISM

Approximate Bayesian Computation (ABC) (Beaumont et al., 2002) is a standard method for tackling LFI problems. In Algorithm 1, we display one of the most popular variants: Sequential Monte Carlo-Approximate Bayesian Computation (SMC-ABC) (Beaumont et al., 2009). In each round, parameters $\boldsymbol{\theta}$ are sampled from the current proposal distribution $p_r(\boldsymbol{\theta})$ for simulation. A rejection step is then involved to remove the $\boldsymbol{\theta}_i$ whose simulation outcomes $\mathbf{x}_i$ cannot reproduce the observed data $\mathbf{x}_o$ with sufficient accuracy. The remaining accepted $\{\boldsymbol{\theta}_i\}_i$ are adopted to update the next round's proposal $p_{r+1}(\cdot)$ towards the target posterior, *i.e.*, by refitting q_ϕ with the modified data. We defer more details of this approach to Section A.1 in Appendix.

It would then be natural to construct an analogical sequence design algorithm using top scored entities $\{\mathbf{m}_i\}_i$ to guide the update of a certain sequence distribution, see Algorithm 2. Interestingly, this is the proposed sequence design algorithm in Gupta & Zou (2019), where the authors name this kind of updating "feedback" because training of the parametric generator $q_\phi(\mathbf{m})$ exploits feedback signals from the oracle. In this paper, we follow Brookes & Listgarten (2018) to crystallize $q_\phi(\mathbf{m})$ to be a variational autoencoder (Kingma & Welling, 2014), and use the term Feedback-Variational AutoEncoder (FB-VAE) to refer to Algorithm 2. We place Algorithm 1 & 2 side-by-side to highlight their correspondence. We also make the same arrangement for the following Algorithm 3 & 4, Algorithm 5 & 6 and Algorithm 7 & 8.

Algorithm 1 SMC-ABC

$p_1(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$;
for r in 1 to R **do**
  repeat
    sample $\boldsymbol{\theta}_i \sim p_r(\boldsymbol{\theta})$;
    simulate $\mathbf{x}_i \sim p(\mathbf{x}|\boldsymbol{\theta}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\theta}_i, \mathbf{x}_i)\}_{i=1}^n$
  sort $\mathcal{D}$ according to $-\|\mathbf{x}_i - \mathbf{x}_o\|$;
  fit $q_\phi(\boldsymbol{\theta})$ with top $\{\boldsymbol{\theta}_i\}_i$ in $\mathcal{D}$;
  $p_{r+1}(\boldsymbol{\theta}) \leftarrow q_\phi(\boldsymbol{\theta})$;
end for
return $\hat{p}(\boldsymbol{\theta}|\mathbf{x}_o) = p_{R+1}(\boldsymbol{\theta})$

Algorithm 2 FB-VAE

$p_1(\mathbf{m}) \leftarrow p(\mathbf{m})$;
for r in 1 to R **do**
  repeat
    sample $\mathbf{m}_i \sim p_r(\mathbf{m})$;
    query the oracle: $s_i \leftarrow f(\mathbf{m}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{m}_i, s_i)\}_{i=1}^n$;
  sort $\mathcal{D}$ according to s_i
  fit $q_\phi(\mathbf{m})$ with top $\{\mathbf{m}_i\}_i$ in $\mathcal{D}$;
  $p_{r+1}(\mathbf{m}) \leftarrow q_\phi(\mathbf{m})$;
end for
return $\{\mathbf{m} : (\mathbf{m}, s) \in \mathcal{D}\}$

In comparison with SMC-ABC, the Sequential Neural Posterior (SNP) method (Papamakarios & Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019) for likelihood-free inference adopts a more flexible approach, taking the power of conditional neural density estimator (*e.g.*, Papamakarios et al. (2017)) to model the general posterior $p(\boldsymbol{\theta}|\mathbf{x})$, which takes arbitrary $\boldsymbol{\theta}$ and $\mathbf{x}$ as two inputs and outputs a distribution. This neural estimator is trained via approximately minimizing the Kullback–Leibler (KL) divergence between $q_\phi(\boldsymbol{\theta}|\mathbf{x})$ and the true posterior $p(\boldsymbol{\theta}|\mathbf{x})$. We defer more training details to Section A.1 in Appendix. Under the connection viewpoint, one similar algorithm for sequence design is the Design by Adaptive Sampling (DbAS) proposed in Brookes & Listgarten (2018) which is characterized in Algorithm 4, fitting $q_\phi(\mathbf{m})$ through minimizing the KL divergence with the posterior $p(\mathbf{m}|\mathcal{E})$. Based on the difference in specific implementations, both algorithms have more than one variant, whose details are deferred to Section A.1 in Appendix.

Algorithm 3 Sequential Neural Posterior

$p_1(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$;
for r in 1 to R **do**
  repeat
    sample $\boldsymbol{\theta}_i \sim p_r(\boldsymbol{\theta})$;
    simulate $\mathbf{x}_i \sim p(\mathbf{x}|\boldsymbol{\theta}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\theta}_i, \mathbf{x}_i)\}_{i=1}^n$;
  $q_\phi \leftarrow \arg\min_q \mathbb{E}_{\mathbf{x}} \left[D_{\mathrm{KL}}(p(\boldsymbol{\theta}|\mathbf{x})\|q)\right]$;
  $p_{r+1}(\boldsymbol{\theta}) \leftarrow q_\phi(\boldsymbol{\theta}|\mathbf{x}_o)$;
end for
return $\hat{p}(\boldsymbol{\theta}|\mathbf{x}_o) = p_{R+1}(\boldsymbol{\theta})$

Algorithm 4 Design by Adaptive Sampling

$p_1(\mathbf{m}) \leftarrow p(\mathbf{m})$;
for r in 1 to R **do**
  repeat
    sample $\mathbf{m}_i \sim p_r(\mathbf{m})$;
    query the oracle: $s_i \leftarrow f(\mathbf{m}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{m}_i, s_i)\}_{i=1}^n$;
  $q_\phi \leftarrow \arg\min_q D_{\mathrm{KL}}(p(\mathbf{m}|\mathcal{E})\|q)$;
  $p_{r+1}(\mathbf{m}) \leftarrow q_\phi(\mathbf{m})$;
end for
return $\{\mathbf{m} : (\mathbf{m}, s) \in \mathcal{D}\}$

We refer to the above algorithms as "backward modeling" because the trained generative network q_ϕ (going from $\mathbf{x}/\mathcal{E}$ to $\boldsymbol{\theta}/\mathbf{m}$) is a sort of reverse model of the simulation mechanism (which goes from $\boldsymbol{\theta}/\mathbf{m}$ to $\mathbf{x}/s$).

3.2 Forward modeling of the mechanism

Whereas the above methods focus on directly modeling the target posterior with a generative model that learns a "reverse mechanism" of the simulation process, it is also possible to model the "forward mechanism", which is consistent with the simulation process. Papamakarios et al. (2019) claim that the forward modeling approach may be an easier task than its backward counterpart, as unbiased estimation of the likelihood does not depend on the choice of proposal. Consequently, in contrast to SNP, Papamakarios et al. (2019) chooses to train a neural density estimator to model the conditional likelihood distribution $q_\phi(\mathbf{x}|\boldsymbol{\theta})$ sequentially in each round. The training is achieved by maximizing the total log likelihood $\max_q \sum_i \log q_\phi(\mathbf{x}_i|\boldsymbol{\theta}_i)$ with data samples from the dataset $\mathcal{D}$ at the current (r-th) round. The downside of this forward approach is an additional computational Markov Chain Monte Carlo (MCMC) step is needed to sample from the r-th round posterior / proposal $p_r(\boldsymbol{\theta})$. The resulting approach, which is coined (Papamakarios et al., 2019) as Sequential Neural Likelihood (SNL), is summarized in Algorithm 5.

In the spirit of directly modeling the forward mechanism of sequence design, we train a regressor $\hat{f}_\phi(\mathbf{m})$ in a supervised manner to fit the oracle scorer. In order to adapt this regressor into the update procedure of the proposal of the next round, we use $\tilde{q}(\mathbf{m})$ to denote the unknown posterior $p(\mathbf{m}|\mathcal{E})$ with knowledge of $\hat{f}_\phi(\mathbf{m})$ and prior $p(\mathbf{m})$. The specific construction of $\tilde{q}(\mathbf{m})$ depends on the choice of $\mathcal{E}$. For instance, if we choose Example B in Section 2.2 to be the definition of $\mathcal{E}$, then $\tilde{q}(\mathbf{m})$ is the distribution with (unnormalized) probability $p(\mathbf{m})\cdot\exp(\hat{f}_\phi(\mathbf{m})/\tau)$. See Section A.2 in Appendix for more elaboration about this point. We then choose the update procedure of the proposal $p_{r+1}(\mathbf{m})$ to be analogical to that of SNL. We name this proposed algorithm to be Iterative Scoring (IS) to avoid confusion with likelihood-free inference algorithms. Furthermore, depending on different definition of $\mathcal{E}$, we use the name "IS-A" and "IS-B" for them in the following sections.

Algorithm 5 Sequential Neural Likelihood

$p_1(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$;
for r in 1 to R **do**
  repeat
    sample $\boldsymbol{\theta}_i \sim p_r(\boldsymbol{\theta})$;
    simulate $\mathbf{x}_i \sim p(\mathbf{x}|\boldsymbol{\theta}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\theta}_i, \mathbf{x}_i)\}_{i=1}^n$
  fit $q_\phi(\mathbf{x}|\boldsymbol{\theta})$ with $\mathcal{D}$;
  $p_{r+1}(\boldsymbol{\theta}) \propto p(\boldsymbol{\theta}) \cdot q_\phi(\mathbf{x}_o|\boldsymbol{\theta})$;
end for
return $\hat{p}(\boldsymbol{\theta}|\mathbf{x}_o) = p_{R+1}(\boldsymbol{\theta})$

Algorithm 6 Iterative Scoring

$p_1(\mathbf{m}) \leftarrow p(\mathbf{m})$;
for r in 1 to R **do**
  repeat
    sample $\mathbf{m}_i \sim p_r(\mathbf{m})$;
    query the oracle: $s_i \leftarrow f(\mathbf{m}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{m}_i, s_i)\}_{i=1}^n$;
  fit $\hat{f}_\phi(\mathbf{m})$ with $\mathcal{D}$;
  construct $\tilde{q}(\mathbf{m})$ with $\hat{f}_\phi(\cdot)$ and $p(\mathbf{m})$;
  $p_{r+1}(\mathbf{m}) \leftarrow \tilde{q}(\mathbf{m})$;
end for
return $\{\mathbf{m} : (\mathbf{m}, s) \in \mathcal{D}\}$

3.3 Modeling a probability ratio

In this subsection, we discuss yet another approach, through the estimation of a probability ratio. Gutmann & Hyvärinen (2010) proposes noise contrastive estimation as a statistical inference approach. This methodology turns a hard probability modeling problem into binary classification, which is considered easier to learn. In contrast to the aforementioned likelihood-free inference methods which rely on a form of density estimation to perform the task, Sequential Neural Ratio (SNR) (Hermans et al., 2019) takes a similar approach as noise contrastive estimation. SNR adopts a classification approach to estimate the likelihood-to-evidence ratio $r(\boldsymbol{\theta}, \mathbf{x}) = p(\boldsymbol{\theta}|\mathbf{x})/p(\boldsymbol{\theta})$. SNR is summarized in Algorithm 7. Specifically, in each round, SNR fits a binary classifier $d_\phi(\boldsymbol{\theta}, \mathbf{x}) \in [0, 1]$ in the following manner:

$$\arg\min_d \left\{ \sum\nolimits_{(\boldsymbol{\theta}_i, \mathbf{x}_i) \in \mathcal{D}} \left[-\log d(\mathbf{x}_i, \boldsymbol{\theta}_i)\right] + \sum\nolimits_{(\boldsymbol{\theta}'_i, \mathbf{x}_i) \in \mathcal{D}'} \left[-\log(1 - d(\mathbf{x}_i, \boldsymbol{\theta}'_i))\right] \right\}. \tag{4}$$

We show that with the $\mathcal{D}$ and $\mathcal{D}'$ established in Algorithm 7, we have

$$d^*(\boldsymbol{\theta}, \mathbf{x}) = \frac{p(\boldsymbol{\theta}|\mathbf{x})}{p(\boldsymbol{\theta}) + p(\boldsymbol{\theta}|\mathbf{x})}, \quad r^*(\boldsymbol{\theta}, \mathbf{x}) := \frac{d^*(\boldsymbol{\theta}, \mathbf{x})}{1 - d^*(\boldsymbol{\theta}, \mathbf{x})} = \frac{p(\boldsymbol{\theta}|\mathbf{x})}{p(\boldsymbol{\theta})}$$

where the "$*$" denotes the optimality. In fact, we have the following Proposition 1:

Proposition 1. *Let $p_0(\mathbf{a})$ and $p_1(\mathbf{a})$ be two distributions for d_a-dimension random variable $\mathbf{a}$ which takes value in the space of $\mathcal{A} = \mathbb{R}^{d_a}$, and $d(\mathbf{a}) : \mathcal{A} \to [0, 1]$ is a real-value function mapping any $\mathbf{a}$ to a positive real value number. Then the functional optimization problem*

$$\arg\max_{d:\mathcal{A}\to[0,1]} \left\{ \mathbb{E}_{\mathbf{a}\sim p_0(\mathbf{a})}[\log d(\mathbf{a})] + \mathbb{E}_{\mathbf{a}\sim p_1(\mathbf{a})}[\log(1 - d(\mathbf{a}))] \right\}.$$

will lead to the optimal solution $d^*(\mathbf{a}) = \frac{p_0(\mathbf{a})}{p_0(\mathbf{a})+p_1(\mathbf{a})}$.

See the proof in Section A.3 in Appendix. After training d, SNR can obtain the posterior density value by $p(\boldsymbol{\theta}|\mathbf{x}) = r^*(\boldsymbol{\theta}, \mathbf{x})p(\boldsymbol{\theta})$. SNR mirrors SNL in that it samples the new proposal $p_{r+1}(\boldsymbol{\theta})$ without explicitly modeling the posterior.

On the other hand, we propose a sequence design algorithm analogous to SNR and named Iterative Ratio (IR), which estimates the probability ratio $r(\mathbf{m}) = p(\mathbf{m}|\mathcal{E})/p(\mathbf{m})$ for the purpose of posterior sampling. IR first builds two datasets $\mathcal{D}$ and $\mathcal{D}'$, corresponding to two different distributions $p(\mathbf{m}|\mathcal{E})$ and $p(\mathbf{m})$. We take a similar binary classification approach, whose training objective is $\min_d \left\{ \sum_{\mathbf{m}\in\mathcal{D}}[-\log d(\mathbf{m})] + \sum_{\mathbf{m}\in\mathcal{D}'}[-\log(1 - d(\mathbf{m}))] \right\}$. The desired ratio is then obtained by $r(\mathbf{m}) := d(\mathbf{m})/(1 - d(\mathbf{m}))$. Other components of IR follow the scheme of IS, and IR is schematized in Algorithm 8. We point out that IR does not have two variants as IS does, as the definition for $\mathcal{E}$ in Example B is not usable because of the unknown normalizing constant for probability $p(\mathcal{E}|\mathbf{m})$. See Section A.3 in Appendix for more explanation.

Algorithm 7 Sequential Neural Ratio

$p_1(\boldsymbol{\theta}) \leftarrow p(\boldsymbol{\theta})$;
for r in 1 to R **do**
  repeat
    sample $\boldsymbol{\theta}_i, \boldsymbol{\theta}'_i \sim p_r(\boldsymbol{\theta})$;
    simulate $\mathbf{x}_i \sim p(\mathbf{x}|\boldsymbol{\theta}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\theta}_i, \mathbf{x}_i)\}_{i=1}^n$
  $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{(\boldsymbol{\theta}'_i, \mathbf{x}_i)\}_{i=1}^n$
  train $d_\phi(\boldsymbol{\theta}, \mathbf{x})$ classifying between $\mathcal{D}$ and $\mathcal{D}'$ with the loss in Eq. 4;
  $r_\phi(\boldsymbol{\theta}, \mathbf{x}) \leftarrow \frac{d_\phi(\boldsymbol{\theta},\mathbf{x})}{1-d_\phi(\boldsymbol{\theta},\mathbf{x})}$;
  $p_{r+1}(\boldsymbol{\theta}) \propto r_\phi(\boldsymbol{\theta}, \mathbf{x}) \cdot p(\boldsymbol{\theta})$;
end for
return $\hat{p}(\boldsymbol{\theta}|\mathbf{x}_o) = p_{R+1}(\boldsymbol{\theta})$

Algorithm 8 Iterative Ratio

$p_1(\mathbf{m}) \leftarrow p(\mathbf{m})$;
for r in 1 to R **do**
  repeat
    sample $\mathbf{m}_i \sim p_r(\mathbf{m})$;
    query the oracle: $s_i \leftarrow f(\mathbf{m}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{m}_i, s_i)\}_{i=1}^n$;
  construct $\tilde{\mathcal{D}}$ with $\mathbf{m}$ in $\mathcal{D}$ satisfying $\mathcal{E}$;
  construct $\tilde{\mathcal{D}}'$ from $p(\mathbf{m})$;
  train $d_\phi(\mathbf{m})$ classifying between $\tilde{\mathcal{D}}$ and $\tilde{\mathcal{D}}'$;
  $r_\phi(\mathbf{m}) \leftarrow \frac{d_\phi(\mathbf{m})}{1-d_\phi(\mathbf{m})}$;
  $p_{r+1}(\boldsymbol{\theta}) \propto r_\phi(\mathbf{m}) \cdot p(\mathbf{m})$;
end for
return $\{\mathbf{m} : (\mathbf{m}, s) \in \mathcal{D}\}$

3.4 Composite probabilistic methods

Building on the above analogies and framework, we move beyond the above algorithms in this subsection. The previous lines of approach – the direct posterior modeling methods in Section 3.1 and the indirect methods in Section 3.2 and 3.3 – both have their own advantages and disadvantages. The former methods may fail to get accurate inference result due to unmatched proposals, while the latter ones would need extra large amount of computation for the MCMC sampling process before obtaining accurate posterior samples, etc. Here we study composite algorithms that combine the aforementioned ingredients through the lens of our proposed unified framework. Our goal is to combine the strengths from both kinds of methods.

We first introduce Iterative Posterior Scoring (IPS) method illustrated in Algorithm 9. IPS also uses a neural network $\hat{f}_\phi(\mathbf{m})$ to model the forward mechanism as IS does, and again we use $\tilde{q}$ here

to denote the target posterior $p(\mathbf{m}|\mathcal{E})$. Instead of applying computational MCMC steps here, we train a second parametrized model q_ψ to model $\tilde{q}$ by minimizing the KL divergence between them. Notice that the optimization of q_ψ is restricted within a neural network parameterization family. As a result, in the next round, we can directly utilize $q_\psi(\mathbf{m})$ to serve as a flexible generative proposal of $p_{r+1}(\mathbf{m})$. Like IS, the IPS algorithm also has two different variants with regard to different choices of $\mathcal{E}$, we name them to be IPS-A and IPS-B. Two choices differ in the detailed construction of distribution $\tilde{q}(\mathbf{m})$.

In a similar spirit, we propose the Iterative Posterior Ratio (IPR) algorithm (see Algorithm 10). IPR is close to the IR algorithm in many aspects, but also adopts a second neural network model $q_\psi(\mathbf{m})$ like IPS. IPR works similarly to IR, in that we also construct $\tilde{q}(\mathbf{m})$, taking advantage of $r_\phi(\mathbf{m})$ and the prior $p(\mathbf{m})$ simply via $\tilde{q}(\mathbf{m}) \leftarrow r_\phi(\mathbf{m}) \cdot p(\mathbf{m})$. Note that the usage of two models in IPR is not exactly the same as in IPS: $q_\psi(\mathbf{m})$ is also achieved via minimizing KL divergence with $\tilde{q}(\mathbf{m})$, but the training of model $d_\phi(\mathbf{m})$ is closer to that in IR rather than the $\hat{f}_\phi(\mathbf{m})$ in IS. Another similarity between IPR and IR is that IPR also only has one variant, since the Example B is not applicable for this ratio modeling approach (see Section A.4 in Appendix for more details).

Algorithm 9 Iterative Posterior Scoring

$p_1(\mathbf{m}) \leftarrow p(\mathbf{m})$;
for r in 1 to R **do**
  repeat
    sample $\mathbf{m}_i \sim p_r(\mathbf{m})$;
    query the oracle: $s_i \leftarrow f(\mathbf{m}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{m}_i, s_i)\}_{i=1}^n$;
  fit $\hat{f}_\phi(\mathbf{m})$ with $\mathcal{D}$;
  construct $\tilde{q}(\mathbf{m})$ with $\hat{f}_\phi(\cdot)$ and $p(\mathbf{m})$;
  $q_\psi \leftarrow \arg\min_q D_{\mathrm{KL}}(\tilde{q}(\mathbf{m})\|q)$;
  $p_{r+1}(\mathbf{m}) \leftarrow q_\psi(\mathbf{m})$;
end for
return $\{\mathbf{m} : (\mathbf{m}, s) \in \mathcal{D}\}$

Algorithm 10 Iterative Posterior Ratio

$p_1(\mathbf{m}) \leftarrow p(\mathbf{m})$;
for r in 1 to R **do**
  repeat
    sample $\mathbf{m}_i \sim p_r(\mathbf{m})$;
    query the oracle: $s_i \leftarrow f(\mathbf{m}_i)$;
  until n samples are obtained
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{m}_i, s_i)\}_{i=1}^n$;
  construct $\tilde{\mathcal{D}}$ with $\mathbf{m}$ in $\mathcal{D}$ satisfying $\mathcal{E}$;
  construct $\tilde{\mathcal{D}}'$ from $p(\mathbf{m})$;
  train $d_\phi(\mathbf{m})$ classifying between $\tilde{\mathcal{D}}$ and $\tilde{\mathcal{D}}'$;
  $r_\phi(\mathbf{m}) \leftarrow \frac{d_\phi(\mathbf{m})}{1-d_\phi(\mathbf{m})}$;
  construct $\tilde{q}(\mathbf{m})$ with $r_\phi(\mathbf{m})$ and $p(\mathbf{m})$;
  $q_\psi \leftarrow \arg\min_q D_{\mathrm{KL}}(\tilde{q}(\mathbf{m})\|q)$;
  $p_{r+1}(\mathbf{m}) \leftarrow q_\psi(\mathbf{m})$;
end for
return $\{\mathbf{m} : (\mathbf{m}, s) \in \mathcal{D}\}$

4 EXPERIMENTS

4.1 SETUP

In this section, we systematically evaluate the proposed methods and baselines on four different *in-silico* biological sequence design benchmarks. In every round, we allow each algorithm to query the black-box oracle for a batch of n sequences $\mathbf{m}_i$ to obtain their true scores s_i, with $n = 100$ for all experiments. The total number of rounds differs across different tasks.

We experiment with our six proposed methods: Iterative Scoring (-A/B) from Section 3.2, Iterative Ratio from Section 3.3, Iterative Posterior Scoring (-A/B) and Iterative Posterior Ratio from Section 3.4. Apart from these proposed methods, we consider as baselines a battery of existing methods designed for batched black-box sequence design tasks: (1) Random, a method that randomly select proposal sequences at every round; (2) FB-VAE (Gupta & Zou, 2019) depicted in Section 3.1; (3) Evolution based (Brindle, 1980; Real et al., 2019) sequence design algorithm; and (4) DbAS (Brookes & Listgarten, 2018), described in Section 3.1.

We evaluate these sequence design algorithms by the average score of the top-10 and top-100 sequences in the resulting dataset $\mathcal{D}$ at each round. We plot the average score curves with regard to the number of rounds. We also use the area under the curve as a scalar metric for sample efficiency to compare the methods being evaluated. Specifically, since the area depends on the choice of x-axis, we simply cumulate the scores of all rounds to calculate the area. The result could be non-positive, since the score can take negative values.

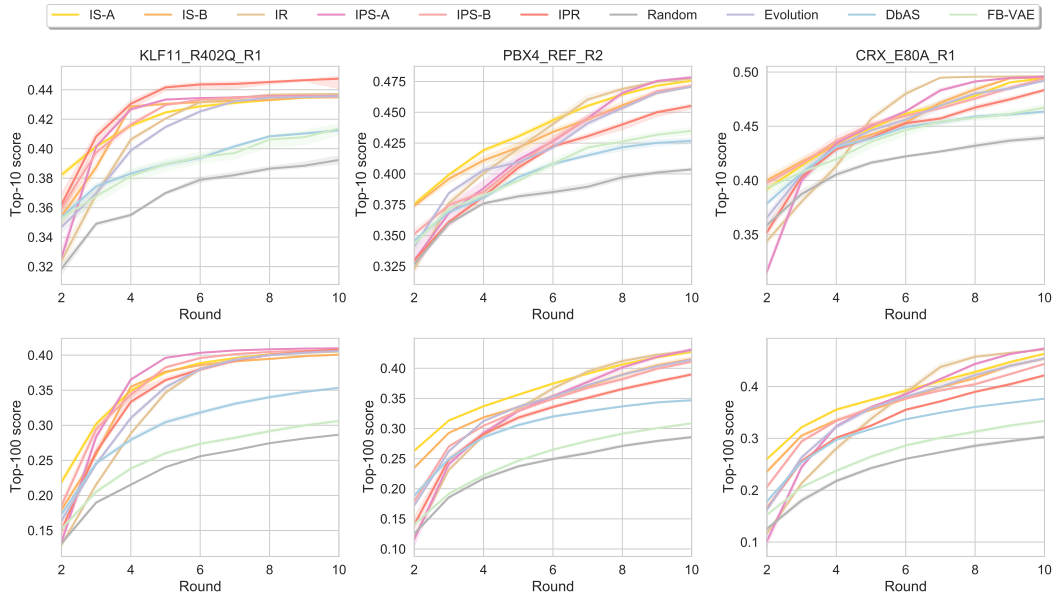

Figure 1: Top score (y-axis) curves of different methods on 3 TfBind problems (KLF11_R402Q_R1, PBX4_REF_R2 and CRX_E80A_R1) with regard to the number of rounds.

	IS-A	IS-B	IR	IPS-A	IPS-B	IPR	RANDOM	EVOLUTION	DBAS	FB-VAE
TOP-10	8.43	6.93	4.79	6.21	8.36	7.00	1.14	4.36	5.07	2.71
TOP-100	9.57	7.93	4.07	6.71	7.64	6.00	1.00	5.50	4.57	2.00

Table 2: Mean rank of evaluated algorithms with regard to the area under the top score curve for TfBind problems. The rank ranges from 1 to 10. Higher rank is better.

4.2 RESULTS

Transcription factor binding sites (TfBind). Protein sequences that bind with DNA sequences to adjust their activity are called transcription factors. In Barrera et al. (2016), the authors measure the binding properties between a battery of transcription factors and all possible length-8 DNA sequences through biological experiments. Concretely, we choose 15 transcription factors to serve as 15 different tasks. For each transcription factor, the algorithm needs to search for sequences that maximize the corresponding binding activity score. The size of the search space is $|\mathcal{V}|^L = 4^8 = 65536$. The number of total rounds is fixed to 10. For validation, we follow Angermüller et al. (2020b) and use one task (ZNF200_S265Y_R1) for hyperparameter selection. Then we test the algorithms' performance on the other 14 held-out tasks.

Figure 1 displays a comparison for all ten methods on three of the chosen binding affinity tasks. We can observe that after 10 rounds, our proposed methods perform consistently better than the backward modeling methods like DbAS and FB-VAE in terms of both Top-10 and Top-100 scores. Among all baselines, the evolution method is the strongest one, and it beats IR on some of the tasks (see complete results in Table 4 and 5 in Appendix). We also find that IS-A and IS-B increase top scores slightly faster than other methods, especially on PBX4_REF_R2 and CRX_E80A_R1. This indicates that composite methods' way of using parameterized models to replace computational procedures is not the optimal solution for small-scale tasks.

5' untranslated regions (UTR). The translation efficiency is mainly determined by the sequence of 5' UTR (Alipanahi et al., 2015). In Sample et al. (2019), the authors create a library of gene sequences with ribosome loading level as labels. They further train a convolutional neural network with this library to predict the relationship between a 5'UTR sequence and the corresponding gene expression level. We use this neural network as an oracle for this benchmark. The length of the gene sequences is fixed to 50, and thus the size of the search space is 4^{50}. For this 5'UTR benchmark, we also allow each algorithm to explore for 10 rounds. Figure 2 (left) shows that our proposed

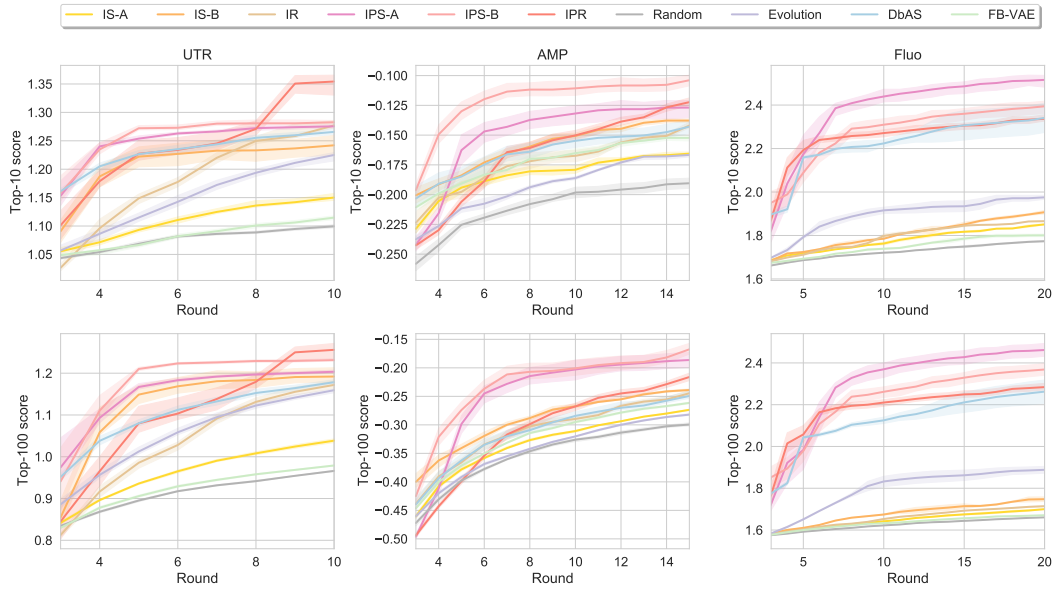

Figure 2: Top score (y-axis) curves of different methods on 3 sequence design problems (*left*: UTR, *middle*: AMP, *right*: Fluo) with regard to the number of rounds.

	IS-A	IS-B	IR	IPS-A	IPS-B	IPR	RANDOM	EVOLUTION	DBAS	FB-VAE
UTR	8.46	9.61	9.03	10.04	10.19	9.52	8.12	9.23	9.59	8.20
AMP	−5.67	−5.11	−5.37	−4.65	−4.39	−5.42	−5.96	−5.79	−5.39	−5.54
FLUO	32.76	33.33	32.95	44.51	43.13	42.42	31.64	35.44	41.40	32.52

Table 3: Comparison of the area under top-100 curves for UTR, AMP and Fluo benchmarks. Larger area means better sample efficiency.

composite methods significantly outperform other methods on the UTR task. Different from the results on the TfBind task, forward modeling methods do not achieve the best performance.

Antimicrobial peptides (AMP). Protein modeling has recently become a popular sub-area of machine learning research. We are tasked to generate AMP sequences, which are short protein sequences against multi-resistant pathogens. We train a binary classifier model to classify whether a short protein sequence belongs to AMP and defer the related details to Appendix. This is the only task we consider regarding sequence design with alterable lengths, where the length of sequences ranges from 12 to 60. Since each entry of protein sequence has $|\mathcal{V}| = 20$ different choices on amino acids, the size of search space is $\sum_{L=12}^{60} 20^L$. We set the number of total rounds to be 15 for this task. Figure 2 (middle) clearly shows that the performances of IPS-A and IPS-B dominate the AMP generation task, which demonstrates the effectiveness of our composite strategy.

Fluorescence proteins (Fluo). As another protein engineering task, we consider the optimization over fluorescent proteins, which is a commonly used test bed of modern molecular biology. This task is similar to the AMP task introduced above, but its ground-truth measurement relies on a regressor. We use a pretrained model taken from Rao et al. (2019) to act as our task oracle, which is a regressor trained to predict log-fluorescence intensity value over approximately 52,000 protein sequences of length 238 (Sarkisyan et al., 2016). More concretely, the regressor is fit on a small neighborhood of parent green fluorescent protein, and is then evaluated on a more distant protein. The training data is derived from the naturally occurring GFP in *Aequorea victoria*. The task Fluo's search space is 20^{238} and we set the number of rounds to 20. We demonstrate the Fluo results in Figure 2 (right), where we can see both composite methods and DbAS achieve much better results than other approaches. This might signify that for long sequence design tasks, backward modeling is superior to other modeling methods, which is not consistent with LFI (Papamakarios et al., 2019). We also summarize the sample efficiency results for the latter 3 benchmarks in Table 3 and 6, from which we can see that our proposed three composite methods perform remarkably promising results.

REFERENCES

Sungsoo Ahn, Junsu Kim, Hankook Lee, and Jinwoo Shin. Guiding deep molecular optimization with genetic exploration. *ArXiv*, abs/2007.04897, 2020.

Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33 (8):831–838, 2015.

Christof Angermüller, David Belanger, Andreea Gane, Zelda E. Mariet, David Dohan, Kevin Murphy, Lucy J. Colwell, and D. Sculley. Population-based black-box optimization for biological sequence design. In *ICML*, 2020a.

Christof Angermüller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy J. Colwell. Model-based reinforcement learning for biological sequence design. In *ICLR*, 2020b.

Luis A. Barrera, Anastasia Vedenko, Jesse V. Kurland, Julia M Rogers, Stephen S. Gisselbrecht, Elizabeth J Rossin, Jaie C. Woodard, Luca Mariani, Kian Hong Kock, Sachi Inukai, Trevor Siggers, Leila Shokri, Raluca Gordân, Nidhi Sahni, Chris Cotsapas, Tong Hao, S. Stephen Yi, Manolis Kellis, Mark J. Daly, Marc Vidal, David E. Hill, and Martha L. Bulyk. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351:1450 – 1454, 2016.

M. Beaumont, Wenyang Zhang, and D. Balding. Approximate bayesian computation in population genetics. *Genetics*, 162 4:2025–35, 2002.

M. Beaumont, J. Cornuet, J. Marin, and C. Robert. Adaptive approximate bayesian computation. *Biometrika*, 96:983–990, 2009.

Michael G. B. Blum. Approximate bayesian computation: A nonparametric perspective. *Journal of the American Statistical Association*, 105:1178 – 1187, 2009.

Johann Brehmer, Gilles Louppe, Juan Pavez, and Kyle Cranmer. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117:5242 – 5249, 2020.

A. Brindle. Genetic algorithms for function optimization. 1980.

David H. Brookes and J. Listgarten. Design by adaptive sampling. *ArXiv*, abs/1810.03714, 2018.

David H. Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *ICML*, 2019.

Jeffrey Chan, Valerio Perrone, Jeffrey P. Spence, Paul A. Jenkins, Sara Mathieson, and Yun S. Song. A likelihood-free inference framework for population genetic data using exchangeable neural networks. *bioRxiv*, 2018.

K. Chen and F. Arnold. Enzyme engineering for nonaqueous solvents: Random mutagenesis to enhance activity of subtilisin e in polar organic media. *Bio/Technology*, 9:1073–1077, 1991.

Yanzhi Chen and Michael U. Gutmann. Adaptive gaussian copula abc. *ArXiv*, abs/1902.10704, 2019.

Yanzhi Chen, Dinghuai Zhang, M. Gutmann, Aaron C. Courville, and Zhanxing Zhu. Neural approximate sufficient statistics for implicit models. *ArXiv*, abs/2010.10079, 2021.

P. T. de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134:19–67, 2005.

Christopher C. Drovandi, Clara Grazian, Kerrie L. Mengersen, and Christian P. Robert. Approximating the likelihood in approximate bayesian computation. *arXiv: Computation*, 2018.

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing. *arXiv preprint arXiv:2007.06225*, 2020.

Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation (with discussion). 2012.

Rafael Gómez-Bombarelli, David Kristjanson Duvenaud, José Miguel Hernández-Lobato, 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*, 4:268 – 276, 2018.

Sai Krishna Gottipati, Boris Sattarov, Sufeng Niu, Yashaswi Pathak, Haoran Wei, Shengchao Liu, Simon Blackburn, Karam Thomas, Connor Coley, Jian Tang, et al. Learning to navigate the synthetically accessible chemical space using reinforcement learning. In *International Conference on Machine Learning*, pp. 3668–3679. PMLR, 2020.

David S. Greenberg, M. Nonnenmacher, and J. Macke. Automatic posterior transformation for likelihood-free inference. *ArXiv*, abs/1905.07488, 2019.

Gabriel Lima Guimaraes, Benjamín Sánchez-Lengeling, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *ArXiv*, abs/1705.10843, 2017.

Anvita Gupta and J. Zou. Feedback gan for dna optimizes protein functions. *Nature Machine Intelligence*, 1:105–111, 2019.

Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

Michael U. Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Likelihood-free inference via classification. *Statistics and Computing*, 28:411 – 425, 2018.

Tatsunori B. Hashimoto, Steve Yadlowsky, and John C. Duchi. Derivative free optimization via repeated classification. In *AISTATS*, 2018.

J. Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free mcmc with amortized approximate likelihood ratios. *arXiv: Machine Learning*, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

Jp Hughes, S. Rees, SB Kalindjian, and KL Philpott. Principles of early drug discovery. *British Journal of Pharmacology*, 162, 2011.

Rafael Izbicki, Ann B. Lee, and Chad M. Schafer. High-dimensional density ratio estimation with extensions to approximate likelihood computation. In *AISTATS*, 2014.

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

Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

Jian Ping Li, David J. Nott, Y. Fan, and Scott Anthony Sisson. Extending approximate bayesian computation methods to high dimensions via a gaussian copula model. *Comput. Stat. Data Anal.*, 106:77–89, 2017.

Jarno Lintusaari, Michael U. Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Fundamentals and recent developments in approximate bayesian computation. *Systematic Biology*, 66: e66 – e82, 2017.

Ge Liu, Haoyang Zeng, Jonas Mueller, Brandon Carter, Ziheng Wang, Jonas Schilz, Geraldine Horny, Michael E. Birnbaum, Stefan Ewert, and David Kenneth Gifford. Antibody complementarity determining region design using high-capacity machine learning. *bioRxiv*, 2020.

Jan-Matthis Lueckmann, Pedro J. Gonçalves, G. Bassetto, Kaan Öcal, M. Nonnenmacher, and J. Macke. Flexible statistical inference for mechanistic models of neural dynamics. In *NIPS*, 2017.

Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H. Macke. Likelihood-free inference with emulator networks. *ArXiv*, abs/1805.09294, 2018.

Jean-Michel Marin, Pierre Pudlo, Christian P. Robert, and Robin J. Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22:1167–1180, 2012.

Kerrie L. Mengersen, Pierre Pudlo, and Christian P. Robert. Approximate bayesian computation via empirical likelihood. 2012.

Daniel Neil, Marwin H. S. Segler, Laura Guasch, Mohamed Ahmed, Dean Plumbley, Matthew Sellwood, and Nathan Brown. Exploring deep recurrent models with reinforcement learning for molecule design. In *ICLR*, 2018.

AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. *arXiv preprint arXiv:1909.11655*, 2019.

George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with bayesian conditional density estimation. In *NIPS*, 2016.

George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. *ArXiv*, abs/1705.07057, 2017.

George Papamakarios, D. Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *AISTATS*, 2019.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S Song. Evaluating protein transfer learning with tape. In *Advances in Neural Information Processing Systems*, 2019.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

Reuven Y. Rubinstein and Dirk P. Kroese. The cross-entropy method. In *Information Science and Statistics*, 2004.

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *ArXiv*, abs/1703.03864, 2017.

Paul Sample, Ban Wang, David W. Reid, Vladimir Presnyak, Iain J Mcfadyen, David R. Morris, and Georg Seelig. Human 5 utr design and variant effect prediction from a massively parallel translation assay. *Nature Biotechnology*, 37:803–809, 2019.

Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

Bobak Shahriari, Kevin Swersky, Ziyun Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104:148–175, 2016.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *ArXiv*, abs/2001.09382, 2020.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15: 1929–1958, 2014.

Owen Thomas, Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U. Gutmann. Likelihood-free inference by ratio estimation. *arXiv: Machine Learning*, 2016.

Minh-Ngoc Tran, David J. Nott, and Robert Kohn. Variational bayes with intractable likelihood. *Journal of Computational and Graphical Statistics*, 26:873 – 882, 2015.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3381–3387, 2008.

Jacob Witten and Zack Witten. Deep learning regression model for antimicrobial peptide design. *BioRxiv*, pp. 692681, 2019.

Simon N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466: 1102–1104, 2010.

Zachary Wu, S B Jennifer Kan, Russell D Lewis, Bruce J. Wittmann, and Frances H. Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116:8852 – 8858, 2019.

Kevin K. Yang, Zachary Wu, and Frances H. Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature Methods*, pp. 1–8, 2019.

Naruki Yoshikawa, Kei Terayama, Masato Sumita, Teruki Homma, Kenta Oono, and Koji Tsuda. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters*, 47(11):1431–1434, 2018.

Zhenpeng Zhou, Steven M. Kearnes, Li Li, Richard N. Zare, and Patrick F. Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9, 2019.