A MORE ABOUT METHODOLOGY

A.1 BACKWARD MODELING OF THE MECHANISM

About SMC-ABC. In Algorithm 1, when we pick the top-m $\boldsymbol{\theta}$ at the r-th round, the picked parameters actually follow such a distribution

$$p(\boldsymbol{\theta}|\boldsymbol{x}_o) \propto \sum_{l=1}^{r} p_l(\boldsymbol{\theta}) \cdot p(\|\boldsymbol{x} - \boldsymbol{x}_o\| < \epsilon \mid \boldsymbol{\theta}) = \left(\sum_{l=1}^{r} p_l(\boldsymbol{\theta})\right) \cdot p(\|\boldsymbol{x} - \boldsymbol{x}_o\| < \epsilon \mid \boldsymbol{\theta}) \tag{5}$$

where the ϵ here is implicitly defined by how "top" the selection process is, namely the ratio $\frac{m}{nr}$. As a result, we point out that in Algorithm 1, another more complicated form of the last step $p_{r+1}(\boldsymbol{\theta}) \leftarrow q_\phi(\boldsymbol{\theta})$ is $p_{r+1}(\boldsymbol{\theta}) \propto q_\phi(\boldsymbol{\theta})p(\boldsymbol{\theta})/\sum_l^r p_l(\boldsymbol{\theta})$, where an additional renormalizing term is involved. We ignore this term and used the simpler alternative in order to keep our main text clean. We refer interested readers to Beaumont et al. (2009) for more details.

About Sequential Neural Posterior. Define $p(\mathbf{x}) = \int p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$ and $\tilde{p}(\mathbf{x}) = \int p(\mathbf{x}|\boldsymbol{\theta})\tilde{p}(\boldsymbol{\theta})d\boldsymbol{\theta}$ for any arbitrary proposal distribution $\tilde{p}(\boldsymbol{\theta})$ which is not necessary to be the prior $p(\boldsymbol{\theta})$. What's more, we define $p(\boldsymbol{\theta}|\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathbf{x})$ and $\tilde{p}(\boldsymbol{\theta}|\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta})\tilde{p}(\boldsymbol{\theta})/\tilde{p}(\mathbf{x})$ to be the true posterior and the proposal posterior.

Starting from the goal of approximating the true posterior,

$$\begin{aligned}
\arg\min_q \mathbb{E}_{p(\mathbf{x})}[D_{\mathrm{KL}}(p(\boldsymbol{\theta}|\mathbf{x})\|q(\boldsymbol{\theta}|\mathbf{x}))] &= \arg\max_q \int p(\mathbf{x})d\mathbf{x} \int p(\boldsymbol{\theta}|\mathbf{x}) \log q(\boldsymbol{\theta}|\mathbf{x})d\boldsymbol{\theta} \\
&= \arg\max_q \int p(\boldsymbol{\theta}, \mathbf{x}) \log q(\boldsymbol{\theta}|\mathbf{x})d\boldsymbol{\theta}\mathbf{x} \\
&= \arg\max_q \mathbb{E}_{p(\boldsymbol{\theta},\mathbf{x})}[\log q(\boldsymbol{\theta}|\mathbf{x})].
\end{aligned}$$

It seems that we can directly train the parameterized neural density estimator q_ϕ in a data driven manner via $\max_\phi \sum_i \log q_\phi(\boldsymbol{\theta}_i|\mathbf{x}_i)$ where i is the index for data sample. When the number of training samples as well as the parameterization family of ϕ are large enough, the obtained q_ϕ would be close enough to the true posterior. This would require the data samples to follow $(\boldsymbol{\theta}_i, \mathbf{x}_i) \sim p(\boldsymbol{\theta}, \mathbf{x}) = p(\boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\theta})$. However, practically one uses a proposal $\tilde{p}(\boldsymbol{\theta})$ to first generate some $\{\boldsymbol{\theta}_i\}_i$ and then generate $\{\mathbf{x}_i\}_i$ by simulation. When the proposal distribution $\tilde{p}(\boldsymbol{\theta})$ is not exactly the prior distribution $p(\boldsymbol{\theta})$, the resulting $q_\phi(\cdot|\cdot)$ would be:

$$\tilde{p}(\boldsymbol{\theta}|\mathbf{x}) = p(\boldsymbol{\theta}|\mathbf{x})\frac{\tilde{p}(\boldsymbol{\theta})p(\mathbf{x})}{p(\boldsymbol{\theta})\tilde{p}(\mathbf{x})} \propto p(\boldsymbol{\theta}|\mathbf{x})\frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}, \tag{6}$$

which is a biased estimation and is not what we want.

Three variants of SNP take different approaches to try to fix this bias. SNP-A (Papamakarios & Murray, 2016) first fits the biased proposal posterior $\tilde{p}(\boldsymbol{\theta}|\mathbf{x})$ in the aforementioned way and utilize the relation in Eq. 6 to solve for an unbiased estimation. This approach is restricted to mixture of Gaussian distribution family and thus has limited expressiveness. SNP-B (Lueckmann et al., 2017) uses importance sampling to address this issue via $\max_\phi \mathbb{E}_{(\mathbf{x},\boldsymbol{\theta})\sim p(\mathbf{x}|\boldsymbol{\theta})\tilde{p}(\boldsymbol{\theta})}\left[\frac{p(\boldsymbol{\theta})}{\tilde{p}(\boldsymbol{\theta})} \log q_\phi(\boldsymbol{\theta} \mid \mathbf{x})\right]$. One downside of this approach is the high variance involved by the importance weights $p(\boldsymbol{\theta})/\tilde{p}(\boldsymbol{\theta})$. SNP-C (Greenberg et al., 2019) proposes to use reparameterize the proposal posterior by setting $\tilde{q}_\phi(\boldsymbol{\theta}|\mathbf{x}) = q_\phi(\boldsymbol{\theta}|\mathbf{x})\frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}\frac{1}{Z_\phi(\mathbf{x})}$ where $Z_\phi(\mathbf{x}) = \int q_\phi(\boldsymbol{\theta}|\mathbf{x})\frac{\tilde{p}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}d\boldsymbol{\theta}$ is the corresponding normalizing factor for $\mathbf{x}$. SNP-C then maximizes $\mathbb{E}_{(\mathbf{x},\boldsymbol{\theta})\sim p(\mathbf{x}|\boldsymbol{\theta})\tilde{p}(\boldsymbol{\theta})}[\log \tilde{q}_\phi(\boldsymbol{\theta} \mid \mathbf{x})]$.

About Design by Adaptive Sampling. We aim to approximate the posterior via minimizing the KL divergence:

$$\begin{aligned}\arg\min_q D_{\mathrm{KL}}(p(\mathbf{m}|\mathcal{E})\|q(\mathbf{m})) &= \arg\max_q \int p(\mathbf{m}|\mathcal{E}) \log q(\mathbf{m}) d\mathbf{m} \\ &= \arg\max_q \int p(\mathcal{E}|\mathbf{m}) p(\mathbf{m}) \log q(\mathbf{m}) d\mathbf{m} \\ &= \arg\max_q \mathbb{E}_{\tilde{q}(\mathbf{m})} \left[\frac{p(\mathbf{m})}{\tilde{q}(\mathbf{m})} p(\mathcal{E}|\mathbf{m}) \log q(\mathbf{m}) \right],\end{aligned}$$

where $\tilde{q}(\mathbf{m})$ could be any distribution of $\mathbf{m}$. Brookes et al. (2019) takes this formulation. Brookes & Listgarten (2018) only differs in the place that it ignores the denominator term. According to Angermüller et al. (2020b), we choose the latter variant as one of our baselines because it is more stable in practice. We refer interested readers to Brookes & Listgarten (2018) for more details.

Notice that we are not doing exactly the same things for LFI and black-box sequence design. Since LFI models flexible posterior $p(\boldsymbol{\theta}|\mathbf{x})$ which is a distribution for arbitrary $\mathbf{x}$, we can also choose to model $p(\mathbf{m}|s)$ for arbitrary s. Nevertheless, in the neural network modeling, conditioning by a scalar value is not an effective approach as the effect of low dimensional scalar value conditioning may be covered by other high dimensional input. Therefore, we choose to directly model the target posterior $p(\mathbf{m}|\mathcal{E})$ with a single neural network.

A.2 Forward modeling of the mechanism

We still use $\tilde{p}(\boldsymbol{\theta})$ to denote an arbitrary proposal distribution and $\tilde{p}(\boldsymbol{\theta}, \mathbf{x}) := p(\mathbf{x}|\boldsymbol{\theta})\tilde{p}(\boldsymbol{\theta})$. Then we have

$$\begin{aligned}\arg\min_q \mathbb{E}_{\tilde{p}(\boldsymbol{\theta})} \left[D_{\mathrm{KL}} \left(p(\mathbf{x}|\boldsymbol{\theta}) \| q(\mathbf{x}|\boldsymbol{\theta}) \right) \right] &= \arg\max_q \int \tilde{p}(\boldsymbol{\theta}) d\boldsymbol{\theta} \int p(\mathbf{x}|\boldsymbol{\theta}) \log q(\mathbf{x}|\boldsymbol{\theta}) d\mathbf{x} \\ &= \arg\max_q \mathbb{E}_{\tilde{p}(\boldsymbol{\theta}, \mathbf{x})} \left[\log q(\mathbf{x}|\boldsymbol{\theta}) \right].\end{aligned}$$

We point out that with much enough data and large enough expressiveness of the neural density estimator parameterization family, no matter what proposal $\tilde{p}(\boldsymbol{\theta})$ is used to provide training samples $\{(\boldsymbol{\theta}_i, \mathbf{x}_i)\}_i \sim \tilde{p}(\boldsymbol{\theta}, \mathbf{x})$, we have the resulting $q_{\hat{\phi}}(\boldsymbol{\theta}|\mathbf{x})$ equals true likelihood $p(\mathbf{x}|\boldsymbol{\theta})$ in the support of the proposal. What SNL gives is an unbiased estimation and thus does not have the same problem as SNP.

Now we elaborate the construction of $\tilde{q}(\mathbf{m})$ in Iterative Scoring algorithm. Here we use the notation $\tilde{q}(\mathbf{m})$ to denote our *approximation* of the posterior $p(\mathbf{m}|\mathcal{E})$. Notice that we want $\tilde{q}(\mathbf{m}) \propto p(\mathbf{m}) \cdot p(\mathcal{E}|\mathbf{m})$. If we choose Example A to serve as the definition of event $\mathcal{E}$, then the samples of $\tilde{q}(\mathbf{m})$ can be obtained in this way: (1) sample $\mathbf{m}$ from prior $p(\mathbf{m})$ and (2) accept this sample if $\hat{f}_\phi(\mathbf{m})$ is larger than threshold s, or otherwise reject it. Alternatively, if we choose Example B, we have $\tilde{q}(\mathbf{m}) \propto p(\mathbf{m}) \cdot \exp(\hat{f}_\phi(\mathbf{m})/\tau)$. Similar to SNL, we do MCMC sampling from this unnormalized probability function.

A.3 Modeling a probability ratio

About Sequential Neural Ratio. Dataset $\mathcal{D}$ is generated in the way that (1) first sample $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$ and (2) simulate $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$. Consequently, $\mathcal{D}$ follows the distribution $p(\boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\theta}) = p(\boldsymbol{\theta}, \mathbf{x})$. On the other hand, the other dataset $\mathcal{D}'$ generates $\boldsymbol{\theta}$ and $\mathbf{x}$ in parallel and independent manner. Notice here $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$ and $\mathbf{x}$ follows the marginal distribution: $\mathbf{x} \sim p(\mathbf{x}) = \int p(\boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\theta})d\boldsymbol{\theta}$.

Proof of Proposition 1.

Proof. We define a functional $\mathcal{F}$ to be the optimization objective:

$$\mathcal{F}[d] = \mathbb{E}_{\mathbf{a}\sim p_0(\mathbf{a})}[\log d(\mathbf{a})] + \mathbb{E}_{\mathbf{a}\sim p_1(\mathbf{a})}[\log(1 - d(\mathbf{a}))]$$

We calculate its functional derivative. For arbitrary function u and infinite small ϵ

$$\begin{aligned}\mathcal{F}[d+\epsilon u]-\mathcal{F}[d] &= \mathbb{E}_{p_0}[\log(1+\epsilon\frac{u}{d})]+\mathbb{E}_{p_1}[\log(1+\epsilon\frac{-u}{1-d})]\\ &= \epsilon\int u\cdot\left(\frac{p_0}{d}+\frac{-p_1}{1-d}\right)+\mathcal{O}(\epsilon)\\ \Rightarrow \lim_{\epsilon\to 0}\frac{\mathcal{F}[d+\epsilon u]-\mathcal{F}[d]}{\epsilon} &= \int u\cdot\left(\frac{p_0}{d}+\frac{-p_1}{1-d}\right)=\int u\cdot\delta\mathcal{F}.\end{aligned}$$

We set the functional derivative to zero:

$$\begin{aligned}\delta\mathcal{F}=0\Rightarrow\frac{p_0}{p_1}&=\frac{d}{1-d}\\ \Rightarrow d(\mathbf{a})&=\frac{p_0(\mathbf{a})}{p_0(\mathbf{a})+p_1(\mathbf{a})}.\end{aligned}$$

This optimal function d^* apparently takes value in $[0,1]$. □

About Iterative Ratio. Notice that in Algorithm 8 we construct two datasets: $\tilde{\mathcal{D}}$ and $\tilde{\mathcal{D}}'$. To generate $\tilde{\mathcal{D}}$, we need to be able to pick some sequence samples $\mathbf{m}$ from $\mathcal{D}$ and make the selected ones follow the posterior $p(\mathbf{m}|\mathcal{E})$. This procedure will depend on our choice of event $\mathcal{E}$. For Example A this is easy, since we just need to filter out the sequences whose oracle value is smaller than the threshold. However, for Example B, it is hard to do similar things, since given score value from $\mathcal{D}$ we only know the unnormalized value of posterior probability, and cannot determine which sequence should be filtered out. The construction of $\tilde{\mathcal{D}}'$ which follows prior distribution $p(\mathbf{m})$ is trivial.

A.4 Composite probabilistic methods

For IPS, the optimization with regard to the second parameterized model $q_\psi(\mathbf{m})$ is

$$\begin{aligned}q_\psi &= \arg\min_q D_{\mathrm{KL}}(\tilde{q}(\mathbf{m})\|q)=\arg\max_q\int\tilde{q}(\mathbf{m})\log q(\mathbf{m})d\mathbf{m}\\ &=\arg\max_q\int p(\mathbf{m}|\mathcal{E})\log q(\mathbf{m})d\mathbf{m}=\arg\max_q\int p(\mathcal{E}|\mathbf{m})p(\mathbf{m})\log q(\mathbf{m})d\mathbf{m}.\end{aligned}$$

The exact value of $p(\mathcal{E}|\mathbf{m})$ depends on different choices of configuration feature $\mathcal{E}$ in Section 2.2. On the other hand, IPR, like IR, also only has one variant, which is with Example A:

$$q_\psi=\arg\min_q D_{\mathrm{KL}}(\tilde{q}(\mathbf{m})\|q)=\arg\max_q\int r_\phi(\mathbf{m})p(\mathbf{m})\log q(\mathbf{m})d\mathbf{m},$$

which is a tractable optimization problem. Both IPR and IR are not fit for Example B since it cannot provide an exact probability value and thus cannot be adopted to construct $\tilde{\mathcal{D}}$.

B More about Experiments

Random method uses no neural network model. FB-VAE uses a VAE model. The encoder of the VAE first linearly transform one-hot input into a hidden feature which is 64 dimension, and then separately linearly transform to a 64-dimension mean output and 64-dimension variance output. The decoder contains a 64×64 linear layer and a linear layer that maps the hidden feature to categorical output. All other methods utilize bi-directional long short-term memory model (BiLSTM) (Hochreiter & Schmidhuber, 1997) with a linear embedding layer. Both the embedding dimension and the hidden size of LSTM is set to 32. For composite methods that use two models, we use one-layer LSTM for each of them. For the other algorithms that only use one LSTM, we set its number of

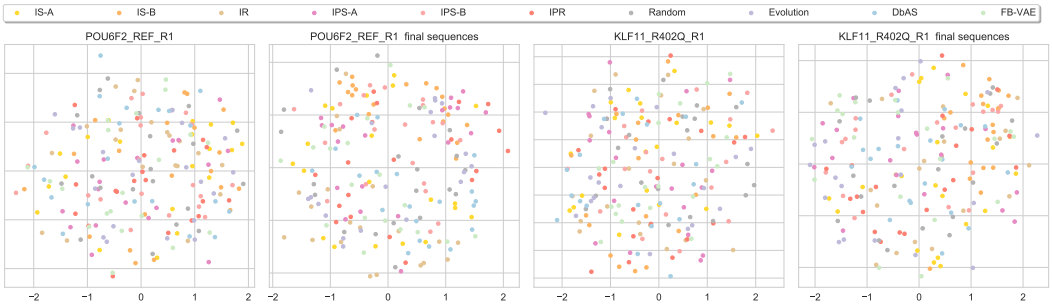

Figure 3: Diversity visualization results for two TfBind tasks.

	IS-A	IS-B	IR	IPS-A	IPS-B	IPR	RANDOM	EVO.	DBAS	FB-VAE
POU6F2_REF_R1	9	7	3	5	10	8	1	4	6	2
KLF11_R402Q_R1	8	6	5	7	9	10	1	4	2	3
EGR2_R359W_R1	4	5	8	9	7	6	1	2	10	3
HOXD13_S316C_R1	8	10	3	4	7	9	1	5	6	2
HOXB7_K191R_R1	10	8	6	4	9	3	1	5	7	2
PBX4_REF_R2	10	9	8	7	5	4	1	6	2	3
GFI1B_A204T_R1	8	7	3	4	9	10	1	6	5	2
FOXC1_REF_R1	10	8	3	5	7	9	1	4	6	2
KLF1_REF_R1	6	4	7	9	8	10	3	1	2	5
SIX6_REF_R1	8	7	3	10	9	6	1	5	2	4
ARX_L343Q_R2	10	3	4	7	9	5	1	6	8	2
CRX_E80A_R1	9	10	7	5	8	2	1	6	3	4
ESX1_K193R_R1	9	6	3	5	10	8	1	4	7	2
VSX1_G160D_R1	9	7	4	6	10	8	1	3	5	2
AVERAGE	8.43	6.93	4.79	6.21	8.36	7.00	1.14	4.36	5.07	2.71

Table 4: Top-10 score ranking for the TfBind instances that we adopt. "Evo." stands for the evolution algorithm.

layers to be two. No Dropout (Srivastava et al., 2014) is used in LSTM models. In this way, the number of parameters of the VAE is slightly larger than that of the two layer BiLSTM, and all methods (except Random) share similar model parameter size.

All the experiments are repeated with fifty random seeds and report the mean value (and also standard deviation in the figure plots). We set $p(\mathbf{m})$ to be uniform prior for all tasks for simplicity, which uniformly samples from the dictionary $\mathcal{V}$ for each entry of the sequence. For length alterable task, we first uniformly sample the length between minimum length and maximum length and then sample each entry.

We explain details about evolution based method mentioned in the main text, which can be seen as a substantial example of directed evolution (Chen & Arnold, 1991). Like other model based methods, Evolution also trains an LSTM regressor to predict the score of a sequence, which is further used to assist in the reproduce procedure. The Evolution algorithm maintains a generation list through the whole exploration process. In each round, the method mutates and reproduces the sequences to enlarge the generation list, and then utilizes the learned regressor to select top sequences for the next generation.

For validation, we follow (Angermüller et al., 2020b) and sweep each algorithm for fifty trials and pick the best configuration. We tune learning rate and whether to re-initialize the optimizer for each new round for all methods. We tune threshold for DbAS, FB-VAE and the methods that is with Example A. For the other choice of $\mathcal{E}$, we tune the temperature. For evolution, we tune the number of offsprings for each sequence in generation list, the probability of substitution, insertion and deletion. For TfBind we use ZNF200_S265Y_R1 for validation. For UTR, AMP and Fluo, since we only have one oracle instance for each benchmark, we do not use a hold-out validation method.

	IS-A	IS-B	IR	IPS-A	IPS-B	IPR	RANDOM	EVO.	DBAS	FB-VAE
POU6F2_REF_R1	10	8	3	5	9	6	1	7	4	2
KLF11_R402Q_R1	10	7	4	8	9	6	1	5	3	2
EGR2_R359W_R1	7	8	4	10	5	9	1	3	6	2
HOXD13_S316C_R1	10	9	3	4	7	6	1	5	8	2
HOXB7_K191R_R1	10	9	3	6	8	4	1	5	7	2
PBX4_REF_R2	10	9	7	5	6	4	1	8	3	2
GFI1B_A204T_R1	10	7	4	5	9	6	1	8	3	2
FOXC1_REF_R1	10	7	4	8	6	9	1	5	3	2
KLF1_REF_R1	8	6	5	9	7	10	1	4	3	2
SIX6_REF_R1	9	7	5	10	8	4	1	6	3	2
ARX_L343Q_R2	10	7	4	8	9	3	1	6	5	2
CRX_E80A_R1	10	9	5	7	8	4	1	6	3	2
ESX1_K193R_R1	10	9	3	4	8	6	1	5	7	2
VSX1_G160D_R1	10	9	3	5	8	7	1	4	6	2
AVERAGE	9.57	7.93	4.07	6.71	7.64	6.00	1.00	5.50	4.57	2.00

Table 5: Top-100 score ranking for the TfBind instances that we adopt. "Evo." stands for the evolution algorithm.

	IS-A	IS-B	IR	IPS-A	IPS-B	IPR	RANDOM	EVOLUTION	DBAS	FB-VAE
UTR	10.87	11.71	11.43	12.06	12.15	11.94	10.60	11.20	11.89	10.65
AMP	−2.98	−2.67	−2.84	−2.54	−2.16	−2.74	−3.36	−3.09	−2.73	−2.80
FLUO	35.31	35.82	35.59	46.19	44.28	43.88	34.33	37.40	43.45	34.78

Table 6: Comparison of the area under top-10 curves for UTR, AMP and Fluo benchmarks. Larger area means better sample efficiency.

For TfBind benchmark, we use the following transcription factor instances and treat them as different black-box optimization tasks: ZNF200_S265Y_R1, POU6F2_REF_R1_8, KLF11_R402Q_R1, EGR2_R359W_R1, HOXD13_S316C_R1, HOXB7_K191R_R1, PBX4_REF_R2, GFI1B_A204T_R1, FOXC1_REF_R1, KLF1_REF_R1, SIX6_REF_R1, ARX_L343Q_R2, CRX_E80A_R1, ESX1_K193R_R1 and VSX1_G160D_R1. We do not do post-processing such as score normalization whitening for the data for simplicity. We first calculate the area under curve to summarize the performance in a scalar output, and put the ranking result for each algorithm in Table 4 and Table 5, which provide more details for Table 2. To further investigate the diversity of different algorithms, we choose two TfBind instances (POU6F2_REF_R1 and KLF11_R402Q_R1) and visualize the resulting sequences with T-SNE (van der Maaten & Hinton, 2008) in Figure 3. We provide two visualization views for both task instances: (1) we uniformly sample 20 sequences from the whole $n \cdot R$ sequences for each algorithm and visualize them; (2) for each algorithm, we visualize 20 sequences uniformly sampled from the last batch (*i.e.*, at the last round). This is notated with "final sequences" in the figure. We do not visualize all the sequences for simplicity. We use Hamming distance in the computation of T-SNE. From Figure 3, we can see that there is no obvious difference for the evaluated methods. This indicates that our proposed methods can achieve better performance while maintaining on-par diversity level with the baselines. This is not exactly consistent to the findings of Angermüller et al. (2020a), which claims some algorithms such as DbAS achieve very limited diversity. We do not use the "optima fraction" metric in Angermüller et al. (2020a;b), since this metric may not deal with multimode oracle landscape well and needs extra unstable computation such as clustering. Besides, this metric cannot generalize to other benchmarks.

We elaborate the construction of our AMP oracle. We use the AMP dataset from (Witten & Witten, 2019) which contains 6,760 AMP sequences. A multilayer perceptron classifier is trained to predict if a protein sequence can prohibit the growth of a particular pathogen in that AMP dataset. This classifier operates on the features extracted by ProtAlbert (Elnaggar et al., 2020) model. Following the setup in (Angermüller et al., 2020b), we treat the predicted logits as the ground-truth measurement. Moreover, we demonstrate the area under Top-10 curves for UTR, AMP and Fluo benchmarks in Table 6, which is a good complement for Table 3 but is missing due to limited space in the main text.

C Related Works and Discussion

Likelihood-free inference. We have already introduced the main classes of likelihood-free inference algorithms in the main text: (1) Approximate Bayesian Computation (ABC) method (Beaumont et al., 2009; Blum, 2009; Marin et al., 2012; Lintusaari et al., 2017) in Section 3.1; (2) Posterior modeling method that is also stated in Section 3.1, including classical ones (Tran et al., 2015; Li et al., 2017; Chen & Gutmann, 2019) and modern SNP methods (Papamakarios & Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019); (3) Likelihood modeling method described in Section 3.2, also containing various classical algorithms (Wood, 2010; Mengersen et al., 2012; Drovandi et al., 2018) and modern SNL variants (Lueckmann et al., 2018; Papamakarios et al., 2019); and (4) Probability ratio modeling methods mentioned in Section 3.3 diverge in estimating likelihood ratio (Gutmann & Hyvärinen, 2010; Gutmann et al., 2018; Brehmer et al., 2020) or likelihood-to-evidence ratio (Thomas et al., 2016; Izbicki et al., 2014), where the latter paradigm is a good fit for LFI problem (Hermans et al., 2019). Besides, there are also works about how to construct low-dimensional summary statistics for LFI (Fearnhead & Prangle, 2012; Chan et al., 2018; Chen et al., 2021).

Machine learning based drug design. Generative modeling and discriminative modeling are two basic ways of thinking in machine learning. In literature for sequence design, generative modeling is also known as *cross entropy method*. This is a famous kind of design method that is close to our "backward modeling of the mechanism" approach. Cross entropy methods seek to solve an expectation maximization problem (*i.e.*, $\max_p \mathbb{E}_{p(\mathbf{m})}[f(\mathbf{m})]$) where the sequences follow a distribution p. On the other hand, we think of this as a way for modeling the posterior $p(\mathbf{m}|\mathcal{E})$ and develop corresponding analysis under the probabilistic framework, which is like a more accurate version of cross entropy method. Many related methods (including the ones stated in Section 3.1) train the distribution by likelihood maximization for sequences with large scores, or use some sort of reweighting to achieve similar effects (Rubinstein & Kroese, 2004; de Boer et al., 2005; Neil et al., 2018; Gupta & Zou, 2019; Brookes et al., 2019).

Discriminative modeling usually goes in a "model-based optimization" way (terminology from Angermüller et al. (2020a)), *i.e.*, use a discriminative model $\hat{f}(\mathbf{m})$ to fit the real oracle $f(\mathbf{m})$ and act as a surrogate for it. The surrogate model can replace the true oracle $f(\mathbf{m})$ which involves costly biological experiments. This corresponds to our "forward modeling of the mechanism" in Section 3.2. Bayesian optimization (Shahriari et al., 2016) is a classical example, which utilizes $\hat{f}$ (typically a Gaussian process model) to define an acquisition function to guide the exploration and exploitation. Many modern biochemical methods also belong to this category (Gómez-Bombarelli et al., 2018; Hashimoto et al., 2018; Yang et al., 2019; Wu et al., 2019; Sample et al., 2019; Liu et al., 2020).

Other categories of drug design methods include evolution algorithms (Brindle, 1980; Wierstra et al., 2008; Salimans et al., 2017; Yoshikawa et al., 2018; Jensen, 2019; Real et al., 2019; Ahn et al., 2020) that search over the target space with genetic operators like insert, mutation, and crossover, and reinforcement learning (Guimaraes et al., 2017; Neil et al., 2018; Zhou et al., 2019; Shi et al., 2020; Angermüller et al., 2020b) which see the formation of a drug as a Markov decision process and train the policy to learn highly-rewarding drugs. We do not find other work that is similar to our probability ratio modeling approach (Section 3.3) from the literature, which we take as a novel contribution.