Appendix of "Degradation-Resistant Offline Optimization via Accumulative Risk Control"

Huakang Lu[a], Hong Qian[a;*], Yupeng Wu[a], Ziqi Liu[c], Ya-Lin Zhang[c], Aimin Zhou[a] and Yang Yu[b]

[a]Shanghai Institute of AI for Education and School of Computer Science and Technology, East China Normal University, Shanghai 200062, China
[b]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[c]Ant Group, Hangzhou 310023, China

The appendix consists of the proof details of Theorem 1, 2, 3 and Inequality (9) in the main paper, a detailed description of offline optimization tasks for experiments (e.g., drug discovery, material invention, and robotic design) as well as their corresponding settings, and the additional implementation details of experiments.

A Proof of Theorems in the Main Paper

Let $\|g\|_\infty = \sup\{|g(\boldsymbol{x})|: \boldsymbol{x} \in \mathcal{X}\}$. Denote the total variation (which is a distance measure) of two probability distributions $\mathcal{U}$ and $\mathcal{V}$ as $TV(\mathcal{U};\mathcal{V}) = \int_{\mathcal{X}} |u(\boldsymbol{x}) - v(\boldsymbol{x})| \mathrm{d}\boldsymbol{x}$, where $u(\boldsymbol{x})$ and $v(\boldsymbol{x})$ are the probability density functions of distributions $\mathcal{U}$ and $\mathcal{V}$, respectively.

Lemma 1. *Given a continuous random variable $\boldsymbol{x}$ and a Lebesgue measurable function g defined over $\mathcal{X}$, if $\|g\|_\infty$ exists, then the difference between expected values of $g(\boldsymbol{x})$ under two distributions $\mathcal{U}$ and $\mathcal{V}$ can be upper bounded by the total variation distance $TV(\mathcal{U};\mathcal{V})$ and $\|g\|_\infty$, i.e,*

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}}[g(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{V}}[g(\boldsymbol{x})] \le 2\|g\|_\infty TV(\mathcal{U};\mathcal{V}) \,. \tag{1}$$

Proof A.1.

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}}[g(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{V}}[g(\boldsymbol{x})] \\
&= \int_{\mathcal{X}} (u(\boldsymbol{x}) - v(\boldsymbol{x}))g(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&\le |\int_{\mathcal{X}} (u(\boldsymbol{x}) - v(\boldsymbol{x}))g(\boldsymbol{x})\mathrm{d}\boldsymbol{x}| \\
&\le \int_{\mathcal{X}} |(u(\boldsymbol{x}) - v(\boldsymbol{x}))g(\boldsymbol{x})|\mathrm{d}\boldsymbol{x} \\
&\le \|g\|_\infty \int_{\mathcal{X}} |u(\boldsymbol{x}) - v(\boldsymbol{x})|\mathrm{d}\boldsymbol{x} \\
&= 2\|g\|_\infty TV(\mathcal{U};\mathcal{V}) \,,
\end{aligned}$$

where the last equality is by $TV(\mathcal{U};\mathcal{V}) = \frac{1}{2}\int_{\mathcal{X}} |u(\boldsymbol{x}) - v(\boldsymbol{x})|\mathrm{d}\boldsymbol{x}$.

A.1 *Proof of Theorem 1*

We have a modeled distribution $h_{\boldsymbol{\phi}}(\boldsymbol{x}) = \frac{\exp(-E_{\boldsymbol{\phi}}(\boldsymbol{x}))}{Z(\boldsymbol{\phi})}$, and contrastive divergence loss $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi}) = \mathrm{KL}(p(\boldsymbol{x})\|h_{\boldsymbol{\phi}}(\boldsymbol{x})) - \mathrm{KL}(q(\boldsymbol{x})\|h_{\boldsymbol{\phi}}(\boldsymbol{x}))$, where $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ are the probability density functions of distributions $\mathcal{P}$ and $\mathcal{Q}$, respectively.

Theorem 1. *(Equivalent form of $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$) Given a low-risk distribution $\mathcal{P}$ and a high-risk distribution $\mathcal{Q}$, $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$ has an equivalent form of $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[E_{\boldsymbol{\phi}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}}[E_{\boldsymbol{\phi}}(\boldsymbol{x})]$.*

Proof A.2. *Considering minimizing $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$ as in the EBM training process, we have that*

$$\begin{aligned}
&\arg\min_{\boldsymbol{\phi}} \mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi}) \\
&= \arg\min_{\boldsymbol{\phi}}[\mathrm{KL}(p(\boldsymbol{x})\|h_{\boldsymbol{\phi}}(\boldsymbol{x})) - \mathrm{KL}(q(\boldsymbol{x})\|h_{\boldsymbol{\phi}}(\boldsymbol{x}))] \\
&= \arg\min_{\boldsymbol{\phi}}[\int_{\mathcal{X}} p(\boldsymbol{x}) \ln \frac{p(\boldsymbol{x})}{h_{\boldsymbol{\phi}}(\boldsymbol{x})}\mathrm{d}\boldsymbol{x} - \int_{\mathcal{X}} q(\boldsymbol{x}) \ln \frac{q(\boldsymbol{x})}{h_{\boldsymbol{\phi}}(\boldsymbol{x})}\mathrm{d}\boldsymbol{x}] \\
&= \arg\min_{\boldsymbol{\phi}}[\int_{\mathcal{X}} p(\boldsymbol{x}) \ln p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int_{\mathcal{X}} p(\boldsymbol{x}) \ln h_{\boldsymbol{\phi}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&\quad - (\int_{\mathcal{X}} q(\boldsymbol{x}) \ln q(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int_{\mathcal{X}} q(\boldsymbol{x}) \ln h_{\boldsymbol{\phi}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x})] \\
&= \arg\min_{\boldsymbol{\phi}}[\int_{\mathcal{X}} q(\boldsymbol{x}) \ln h_{\boldsymbol{\phi}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}) - \int_{\mathcal{X}} p(\boldsymbol{x}) \ln h_{\boldsymbol{\phi}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}] \\
&= \arg\min_{\boldsymbol{\phi}}(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}} \ln h_{\boldsymbol{\phi}}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}} \ln h_{\boldsymbol{\phi}}(\boldsymbol{x})) \,.
\end{aligned}$$

Taking the gradient of $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}} \ln h_{\boldsymbol{\phi}}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}} \ln h_{\boldsymbol{\phi}}(\boldsymbol{x})$, we have that

$$\begin{aligned}
&\frac{\partial(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}} \ln h_{\boldsymbol{\phi}}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}} \ln h_{\boldsymbol{\phi}}(\boldsymbol{x}))}{\partial\boldsymbol{\phi}} \\
&= -(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}}[\frac{\partial E_{\boldsymbol{\phi}}(\boldsymbol{x})}{\partial\boldsymbol{\phi}}] + \frac{\partial Z(\boldsymbol{\phi})}{\partial\boldsymbol{\phi}}) \\
&\quad + (\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\frac{\partial E_{\boldsymbol{\phi}}(\boldsymbol{x})}{\partial\boldsymbol{\phi}}] + \frac{\partial Z(\boldsymbol{\phi})}{\partial\boldsymbol{\phi}}) \\
&= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\frac{\partial E_{\boldsymbol{\phi}}(\boldsymbol{x})}{\partial\boldsymbol{\phi}}] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}}[\frac{\partial E_{\boldsymbol{\phi}}(\boldsymbol{x})}{\partial\boldsymbol{\phi}}] \\
&= \frac{\partial(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[E_{\boldsymbol{\phi}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}}[E_{\boldsymbol{\phi}}(\boldsymbol{x})])}{\partial\boldsymbol{\phi}} \,.
\end{aligned}$$

Since the EBM is trained by a gradient-based optimizer, $\mathcal{L}_{\mathrm{CD}}(\boldsymbol{\phi})$ is equivalent to the loss function $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[E_{\boldsymbol{\phi}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}}[E_{\boldsymbol{\phi}}(\boldsymbol{x})]$ in the energy model training.

A.2 *Proof of Theorem 2*

For any distribution π over the solution space $\mathcal{X}$, let $\widehat{S}_{\boldsymbol{\theta}}(\pi) = \mathbb{E}_{\boldsymbol{x}\sim\pi}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})]$ and $S(\pi) = \mathbb{E}_{\boldsymbol{x}\sim\pi}[f(\boldsymbol{x})]$.

[*] Corresponding Author. E-mail: hqian@cs.ecnu.edu.cn.

Theorem 2. *(Upper Bound of the Prediction Error in Langevin Sampling) For any distribution $\mathcal{Q}^k$ sampled in the Langevin dynamics procedure, if $\|f\|_\infty$ and $\|\widehat{f}_{\boldsymbol{\theta}}\|_\infty$ exist, the prediction error of $\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ on $\mathcal{Q}^k$ is upper bounded*

$$\widehat{S}_{\boldsymbol{\theta}}(\mathcal{Q}^k) - S(\mathcal{Q}^k) \leq \widehat{S}_{\boldsymbol{\theta}}(\mathcal{P}) - S(\mathcal{P}) + C_{f;f_{\boldsymbol{\theta}}} \cdot TV(\mathcal{Q}^k;\mathcal{P})\,, \quad (2)$$

where $C_{f;f_{\boldsymbol{\theta}}} = 2(\|\widehat{f}_{\boldsymbol{\theta}}\|_\infty + \|f\|_\infty)$.

Proof A.3. *Inspired by [13], we decompose $\widehat{S}_{\boldsymbol{\theta}}(\mathcal{Q}^k) - S(\mathcal{Q}^k)$ into three components (a), (b) and (c) below and bound them separately.*

$$\begin{aligned}
&\widehat{S}_{\boldsymbol{\theta}}(\mathcal{Q}^k) - S(\mathcal{Q}^k)\\
&= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}^k}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}^k}[f(\boldsymbol{x})]\\
&= \underbrace{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}^k}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})]}_{(a)}\\
&\quad \underbrace{-(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}^k}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[f(\boldsymbol{x})])}_{(b)}\\
&\quad \underbrace{+\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[f(\boldsymbol{x})]}_{(c)}\,.
\end{aligned}$$

For part (a), by Lemma 1, we have

$$\begin{aligned}
(a) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}^k}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})]\\
&\leq 2\|\widehat{f}_{\boldsymbol{\theta}}\|_\infty TV(\mathcal{Q}^k;\mathcal{P})\,.
\end{aligned}$$

For part (b), we have

$$\begin{aligned}
(b) &= -(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{Q}^k}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[f(\boldsymbol{x})])\\
&\leq 2\|f\|_\infty TV(\mathcal{P};\mathcal{Q}^k)\\
&= 2\|f\|_\infty TV(\mathcal{Q}^k;\mathcal{P})\,,
\end{aligned}$$

where the inequality is by Lemma 1 and the last equality is by the symmetry of total variance distance. For part (c), we have

$$\begin{aligned}
(c) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[f(\boldsymbol{x})]\\
&= \widehat{S}_{\boldsymbol{\theta}}(\mathcal{P}) - S(\mathcal{P})\,,
\end{aligned}$$

which can be controlled by minimizing the predictive loss $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}(\|\widehat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\|^2)$ in the surrogate model training phase. Finally, let $C_{f;f_{\boldsymbol{\theta}}} = 2(\|\widehat{f}_{\boldsymbol{\theta}}\|_\infty + \|f\|_\infty)$, the following result holds

$$\widehat{S}_{\boldsymbol{\theta}}(\mathcal{Q}^k) - S(\mathcal{Q}^k) \leq \widehat{S}_{\boldsymbol{\theta}}(\mathcal{P}) - S(\mathcal{P}) + C_{f;f_{\boldsymbol{\theta}}} \cdot TV(\mathcal{Q}^k;\mathcal{P})\,.$$

A.3 Proof of Theorem 3

Recall that $R_{\boldsymbol{\phi}}(\boldsymbol{x}) = \frac{m(E_{\widetilde{\mathcal{Q}}} - E_{\boldsymbol{\phi}}(\boldsymbol{x}))}{E_{\widetilde{\mathcal{Q}}} - E_{\widetilde{\mathcal{P}}}}$. Given a high-risk distribution $\widetilde{\mathcal{Q}}$, a low-risk distribution $\widetilde{\mathcal{P}}$, and an positive initial momentum m. Let $E_{\widetilde{\mathcal{Q}}} = \mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{Q}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')]$ and $E_{\widetilde{\mathcal{P}}} = \mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{P}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')]$. Since $\widetilde{\mathcal{Q}}$ is sampled by the Langevin dynamics and can reach the maximum value of the energy model, the upper bound in Lemma 1 can be reached. Thus, it is reasonable to assume that the difference between $\mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{Q}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')]$ and $\mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{P}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')]$ is $2\|E\|_\infty TV(\widetilde{\mathcal{Q}};\widetilde{\mathcal{P}})$.

Theorem 3. *(Upper Bound of the Risk Suppression Factor) Given a solution $\boldsymbol{x}$, consider a Gaussian distribution $\mathcal{N}(\boldsymbol{x};\sigma)$ where σ is the standard deviation, if $\|E\|_\infty$ exists, then $R_{\boldsymbol{\phi}}(\boldsymbol{x})$ is upper bounded*

$$R_{\boldsymbol{\phi}}(\boldsymbol{x}) \leq m\frac{TV(\widetilde{\mathcal{Q}};\mathcal{N}(\boldsymbol{x};\sigma))}{TV(\widetilde{\mathcal{Q}};\widetilde{\mathcal{P}})}\,. \quad (3)$$

Proof A.4.

$$\begin{aligned}
R_{\boldsymbol{\phi}}(\boldsymbol{x}) &= \frac{m(E_{\widetilde{\mathcal{Q}}} - E_{\boldsymbol{\phi}}(\boldsymbol{x}))}{E_{\widetilde{\mathcal{Q}}} - E_{\widetilde{\mathcal{P}}}}\\
&= m\frac{\mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{Q}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}'\sim\mathcal{N}(\boldsymbol{x};\sigma)}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')]}{\mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{Q}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{P}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')]}\\
&\leq m\frac{2\|E\|_\infty TV(\widetilde{\mathcal{Q}};\mathcal{N}(\boldsymbol{x};\sigma))}{2\|E\|_\infty TV(\widetilde{\mathcal{Q}};\widetilde{\mathcal{P}})}\\
&= m\frac{TV(\widetilde{\mathcal{Q}};\mathcal{N}(\boldsymbol{x};\sigma))}{TV(\widetilde{\mathcal{Q}};\widetilde{\mathcal{P}})}\,,
\end{aligned}$$

where the inequality is by the assumption and Lemma 1.

A.4 Proof of Inequality (9) in the Main Paper

Proof A.5.

$$\begin{aligned}
\int_{E_{\widetilde{\mathcal{P}}}}^{E_{\widetilde{\mathcal{Q}}}} R_{\boldsymbol{\phi}}(\boldsymbol{x})\mathrm{d}E_{\boldsymbol{\phi}}(\boldsymbol{x}) &= \frac{m\cdot(E_{\widetilde{\mathcal{Q}}} - E_{\widetilde{\mathcal{P}}})}{2}\\
&= \frac{m}{2}(\mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{Q}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}'\sim\widetilde{\mathcal{P}}}[E_{\boldsymbol{\phi}}(\boldsymbol{x}')])\\
&\leq m\cdot\|E\|_\infty\cdot TV(\widetilde{\mathcal{Q}};\widetilde{\mathcal{P}})\,,
\end{aligned}$$

where the last inequality is by Lemma 1.

B Experiment: Offline Optimization Tasks

In this section, we describe the tasks used for experiments in detail. A fundamental motivation in offline optimization is the inaccessibility of expensive evaluations for new solutions. Hence, the measure of methods' optimization performance when online is also intractable if expensive real-world experiments are involved. Towards this end, substitute oracles are proposed in Design-bench [16] to provide evaluations for metering offline optimization methods when online. Herein, the task-specified oracles are also introduced in addition to the background and data as follows.

B.1 TF Bind 8

Background. `TF Bind 8` task is derived from the transcription factor binding activity study [3]. Only SIX6_REF_R1, a specific transcription factor, remains in the dataset. This criterion is widely used as a common evaluation measure for optimization task [1, 2]. TF Bind 8 task aims to design a length-8 DNA sequence with the maximum binding activity.

Data and Evaluation. TF Bind 8 is a discrete optimization task with a dataset size of 65792. All 65792 possible DNA designs are measured with human transcription factor SIX6_REF_R1. The bottom 50% data (23898) is given for offline optimization methods training. The design output is evaluated by an exact oracle which is a lookup table that contains scores for every possible solution.

B.2 ChEMBL

Background. `ChEMBL` task is derived from a large-scale drug property database [8]. The ChEMBL task aims to design a molecule for assay CHEMBL3885882 with a maximum MCHC value. The assay CHEMBL3886882 is chosen because of the high validation rank correlation. Specifically, the random forest regression model achieves the accuracy of 0.7141 when mapping molecules to MCHC values, while other assays cannot reach 0.5.

Data and Evaluation. ChEMBL is a discrete optimization task, and each data is a length-31 SMILES string of categorical variables that take one of 591 elements. The ChEMBL dataset is collected from physical experiments for chemical property measurement. The dataset is preprocessed into discrete token sequences, and all sequences longer than 31 lengths are removed. The oracles are chosen as the traditional optimization methods. The bottom 50% data sorted by MCHC value is given for offline optimization methods training like ChEMBL task does. The evaluator, i.e., the approximate oracle, is a ResNet model.

B.3 UTR

Background. `UTR` task is derived from the study in biology [14], and the construction of related offline optimization task is inspired by [1]. The UTR task aims to design a length-50 DNA sequence with the maximum expression level.

Data and Evaluation. UTR is a discrete optimization task with a dataset size of 280000. The dataset is constructed by collecting the top 280000 length-50 discrete sequences with total read sorting [14], and the bottom 50% data (140000) is given for offline optimization methods training. [16] changes the oracle of the UTR task to deep neural networks used in [1] to obtain the sequence with better visibility for offline optimization methods. The interest of the UTR task (i.e., the expression level of DNA sequence output) is evaluated by a ResNet oracle (approximate oracle).

B.4 Superconductor

Background. `Superconductor` task aims to optimize the chemical formula for a superconducting material with maximum critical temperature.

Data and Evaluation. Superconductor is a continuous optimization task with a dataset size of 21263. The superconductor dataset is from the real-world superconductor problem formerly collected by [10]. Each data is represented as an invertible input vector depending on the serialization of the chemical formula. The random forest oracle (approximate oracle) is learned from the total dataset (21263), and 80% of all samples in the dataset (17010) is divided into offline optimization methods training.

B.5 Dkitty

Background. `Dkitty` task is originally proposed in Design-bench as an offline optimization task, and it aims to optimize the Dkitty robot morphologies to navigate the robot to the specified location with maximum efficiency.

Data and Evaluation. Dkitty task is a continuous optimization task with a dataset size of 25009. The dataset is initialized by Gaussian noise and collected by CMA-ES [11]. The exact oracle is a neural network trained by Soft Actor-Critic (SAC) [9], which can handle a wide range of morphologies. The evaluation depends on the MuJoCo simulator [15] and the pre-trained neural network for 100 time steps and 16 independent and repeated trials. The bottom 60% data is given for offline optimization methods.

B.6 Hopper

Background. `Hopper` task aims to optimize the weights of a neural network policy with the maximum Hopper-v2 Gym [4] return.

Data and Evaluation. Hopper is a continuous optimization task with a dataset size of 3200. The dataset is collected from the neural network weights optimized by the Proximal Policy Optimization (PPO) [6] in the Hopper-v2 MuJoCo environment [15]. Specifically, there are 3200 unique weights in the dataset. The dataset is obtained from 32 trials of PPO after one million steps. The Hopper-v2 Gym simulator (exact oracle) evaluates each network policy input in the Hopper-v2 MuJoCo environment and returns the feedback of 1000 steps simulation.

C Experiment: Additional Implementation Details

Implementation Details of Baseline and SOTA Methods. We describe the implementation details of baseline and SOTA methods that are used for comparison. Following the unified evaluation protocol, we apply reference implementations of BO-qEI [20], CMA-ES [11], REINFORCE [19], gradient ascent, CbAS [5], and MIN [12] in Design-bench [16]. As for DDEA-SE [18], considering that it employs Radial Basis Function (RBF) networks as the surrogate model for the low-dimensional offline optimization tasks in the original work, we use deep neural networks with greater learning capability instead to deal with the more complex tasks in this paper. Finally, we adopt the algorithms and hyperparameter settings of NEMO [7], COMs [17], and IOM [13] reported in the original papers and their corresponding source codes. Exceptionally, for SOTA methods, the hyperparameters are tuned according to settings on similar tasks if they are not provided in a certain task.

References

[1] Christof Angermüller, David Belanger, Andreea Gane, Zelda Mariet, David Dohan, Kevin Murphy, Lucy J. Colwell, and D. Sculley, ‘Population-based black-box optimization for biological sequence design’, in *Proceedings of the 37th International Conference on Machine Learning*, pp. 324–334, Virtual, (2020).

[2] Christof Angermüller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy J. Colwell, ‘Model-based reinforcement learning for biological sequence design’, in *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, (2020).

[3] Luis A Barrera, Anastasia Vedenko, Jesse V Kurland, Julia M Rogers, Stephen S Gisselbrecht, Elizabeth J Rossin, Jaie Woodard, Luca Mariani, Kian Hong Kock, Sachi Inukai, et al., ‘Survey of variation in human transcription factors reveals prevalent dna binding changes’, *Science*, **351**(6280), 1450–1454, (2016).

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, ‘Openai gym’, *CoRR*, **abs/1606.01540**, (2016).

[5] David H. Brookes, Hahnbeom Park, and Jennifer Listgarten, ‘Conditioning by adaptive sampling for robust design’, in *Proceedings of the 36th International Conference on Machine Learning*, pp. 773–782, Long Beach, California, (2019).

[6] Weimin Chen, Kelvin Kian Loong Wong, Sifan Long, and Zhili Sun, ‘Relative entropy of correct proximal policy optimization algorithms with modified penalty factor in complex environment’, *Entropy*, **24**(4), 440, (2022).

[7] Justin Fu and Sergey Levine, ‘Offline model-based optimization via normalized maximum likelihood estimation’, in *Proceedings of the 9th International Conference on Learning Representations*, Virtual, (2021).

[8] Anna Gaulton, Louisa J. Bellis, A. Patrícia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, and John P. Overington, ‘ChEMBL: A large-scale bioactivity database for drug discovery’, *Nucleic Acids Research*, **40**, 1100–1107, (2012).

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, ‘Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor’, in *Proceedings of the 35th International*

Conference on Machine Learning, pp. 1856–1865, Stockholm, Sweden, (2018).

[10] Kam Hamidieh, ‘A data-driven statistical model for predicting the critical temperature of a superconductor’, *Computational Materials Science*, **154**, 346–354, (2018).

[11] Nikolaus Hansen, ‘The CMA evolution strategy: A comparing review’, in *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms*, volume 192, 75–102, Springer, (2006).

[12] Aviral Kumar and Sergey Levine, ‘Model inversion networks for model-based optimization’, in *Advances in Neural Information Processing Systems 33*, pp. 5126–5137, Virtual, (2020).

[13] Han Qi, Yi Su, Aviral Kumar, and Sergey Levine, ‘Data-driven model-based optimization via invariant representation learning’, in *Advances in Neural Information Processing Systems 35*, New Orleans, Louisiana, (2022).

[14] Paul J Sample, Ban Wang, David W Reid, Vlad 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**(7), 803–809, (2019).

[15] Emanuel Todorov, Tom Erez, and Yuval Tassa, ‘Mujoco: A physics engine for model-based control’, in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, Algarve, Portugal, (2012).

[16] Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine, ‘Design-bench: Benchmarks for data-driven offline model-based optimization’, in *Proceedings of the 39th International Conference on Machine Learning*, pp. 17–23, Baltimore, Maryland, (2022).

[17] Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine, ‘Conservative objective models for effective offline model-based optimization’, in *Proceedings of the 38th International Conference on Machine Learning*, pp. 10358–10368, Virtual, (2021).

[18] Handing Wang, Yaochu Jin, Chaoli Sun, and John Doherty, ‘Offline data-driven evolutionary optimization using selective surrogate ensembles’, *IEEE Transactions on Evolutionary Computation*, **23**(2), 203–216, (2019).

[19] Ronald J. Williams, ‘Simple statistical gradient-following algorithms for connectionist reinforcement learning’, *Machine Learning*, **8**, 229–256, (1992).

[20] James T. Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth, ‘The reparameterization trick for acquisition functions’, *CoRR*, **abs/1712.00424**, 75–102, (2017).