From Function to Distribution Modeling: A PAC-Generative Approach to Offline Optimization

Qiang Zhang[*]
Dept of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77843
zhangqiang@tamu.edu

Ruida Zhou[*]
Dept of Electrical and Computer Engineering
UCLA
Los Angeles, CA 90095
ruida@g.ucla.edu

Yang Shen
Dept of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77843
yshen@tamu.edu

Tie Liu
Dept of Electrical and Computer Engineering
Texas A&M University
College Station, TX 77843
tieliu@tamu.edu

Abstract

This paper considers the problem of offline optimization, where the objective function is unknown except for a collection of "offline" data examples. While recent years have seen a flurry of work on applying various machine learning techniques to the offline optimization problem, the majority of these works focused on learning a surrogate of the unknown objective function and then applying existing optimization algorithms. While the idea of modeling the unknown objective function is intuitive and appealing, from the learning point of view it also makes it very difficult to tune the objective of the learner according to the objective of optimization. Instead of learning and then optimizing the unknown objective function, in this paper we take on a less intuitive but more direct view that optimization can be thought of as a process of sampling from a generative model. To learn an effective generative model from the offline data examples, we consider the standard technique of "re-weighting", and our main technical contribution is a probably approximately correct (PAC) lower bound on the natural optimization objective, which allows us to jointly learn a weight function and a score-based generative model from a surrogate loss function. The robustly competitive performance of the proposed approach is demonstrated via empirical studies using the standard offline optimization benchmarks.

1 Introduction

Offline optimization refers to the problem of optimizing an *unknown* real-valued objective function f based only on a collection of "offline" data examples $(\boldsymbol{x}_i, f(\boldsymbol{x}_i)), \forall i \in [m] := \{1, 2, \ldots, m\}$, where each $\boldsymbol{x}_i$ is an independent sample drawn from an *unknown* distribution p_{data}. Aside from these examples, no additional information on the objective function f is available prior to or during the optimization process, and hence the name "offline optimization". This rather restrictive setting is particularly relevant to the optimization scenarios where: i) the objective function is very complex and no structural information is available; and ii) querying the objective function is very expensive.

[*]Q. Zhang and R. Zhou should both be considered first authors.

D3S3: Data-driven and Differentiable Simulations, Surrogates, and Solvers @ NeurIPS 2024.

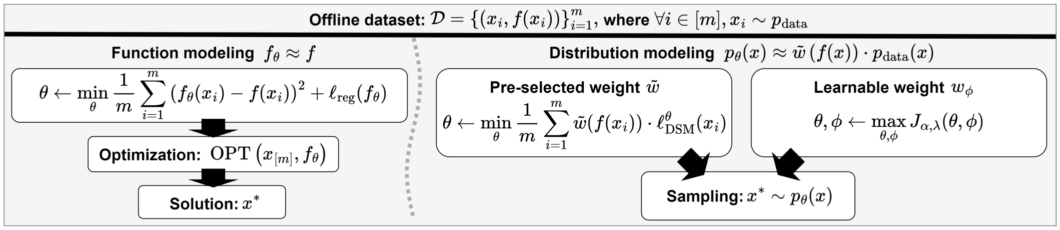


Figure 1: Function vs. distribution modeling for offline optimization.

Obviously, offline optimization is a more challenging setting than standard optimization [6], where *full* structural information on the objective function is available; or black-box optimization [5], where even though no *structural* information on the objective function is available, the objective function can be queried upon during the optimization process. Therefore, instead of aiming at the global optima, for offline optimization we are usually satisfied with finding a few candidates, among which there are *significantly* better [2] solutions than the existing offline observations. Under this lesser objective, a direct application of offline optimization is *experimental design*, for which the candidates are to be efficiently found without querying objective functions through experiments that are often slow or expensive. Applications in the literature include the design of proteins [32], chemical molecules [21], DNA sequences [29], aircraft [25], robots [39], and hardware accelerators [37].

Related work. Traditionally, offline optimization has been mainly approached through the *Bayesian* view, i.e., by endowing the unknown objective function f a *prior* distribution. This has led to a large body of work under the name *Bayesian optimization*; see Fu and Levine [19] and the references therein for the recent progress in this direction. Motivated by the rapid progress in machine learning, recent years have also seen a flurry of work on offline optimization from a *frequentist's* view [11, 22, 35, 45], i.e., by modeling the objective function f as a *deterministic but unknown* function. However, most of these works have been focusing on learning a surrogate of the unknown objective function and then applying existing optimization algorithms (see "Function Modeling" in Figure 1). Prime examples include Trabucco et al. [45], Brookes et al. [11], Gupta and Zou [22], Chen et al. [14].

While the idea of modeling the unknown objective function is intuitive and appealing, from the learning point of view it also makes it very difficult to tune the objective of the learner according to the objective of optimization [45, 11, 22]. As a result, it is very difficult to gauge whether these previous approaches actually come with any theoretical guarantees.

Main contribution. Figure 1 illustrates the key differences between function modeling and distribution modeling for offline optimization. In sharp contrast to function modeling, our approach does *not* explicitly learn a surrogate of the unknown objective function, but directly learns a generative model [3] from the offline examples using a *theoretically-grounded* surrogate of the natural optimization objective: $J_{\text{opt}}(\theta) = \mathbb{E}_{\mathbf{x} \sim p_\theta}[f(\mathbf{x})]$. The main contribution can be summarized as follows:

> We derive a PAC bound on the natural optimization objective, allowing the generative model and weight function to be jointly learned from a *surrogate* loss function that *aligns* the learner's and optimization objectives.

2 The Method

In this paper we take on a less intuitive but more *direct* view of optimization and consider it as a process of *sampling* from a *generative model*. There are two natural advantages to this view. First, through sampling *exploration* is now intrinsic in the optimization process. Second, this view allows us to shift our focus from modeling the objective function to modeling a *target distribution*. Unlike learning a surrogate on the objective function, as we shall see, the objective of learning a generative model can be naturally aligned with the objective of optimization, thus bringing *theoretical guarantees* on the optimization performance.

More specifically, let p_θ be a *generative model* from which sampling can produce, with high probability, samples whose objective values are significantly better than the offline observations. Note that

[2]Throughout the paper, better or superior solutions refer to those with either larger or smaller objective values, depending on whether the goal of optimization is maximizing or minimizing the objective function.

[3]We emphasize here that the idea of leveraging generative models for offline optimization is not entirely new, e.g. Brookes et al. [11], Krishnamoorthy et al. [34, 33].

unlike the traditional generative models, whose purpose to generate samples that are "similar" to the training examples, the goal of our generative model p_θ is to generate samples with *superior* objective values than the offline observations. Relative to the data-generating distribution p_{data}, these targeted samples with superior objective values are the "outliers". Therefore, from the learning perspective, our main challenge here is to learn a generative model that generates *outliers* rather than the norm.

2.1 Learning a Generative Model with a Pre-selected Normalized Weight Function

To facilitate the learning of a desired generative model, in this paper we shall consider the standard technique of "re-weighting" [12]. Roughly speaking, we shall consider a *weight* function that assigns higher weights to the domain points with better objective values and then train a generative model using the *weighted* offline examples. This allows us to tune the generative model towards generating samples with better objective values.

Formally, let $q_{\text{target}}(\boldsymbol{x}) := \tilde{w}(f(\boldsymbol{x})) \cdot p_{\text{data}}(\boldsymbol{x})$ be a *hypothetical* target distribution, where $\tilde{w}$ is a normalized, non-negative *weight* function such that $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\tilde{w}(f(\mathbf{x}))] = 1$, and p_{data} is the (unknown) data-generating distribution from which the offline observations $\boldsymbol{x}_{[m]} := (\boldsymbol{x}_i : i \in [m])$ were drawn. In our approach, the hypothetical target distribution q_{target} plays *dual* roles: On one hand, it serves as the *hypothetical* learning target of the generative model p_θ; on the other hand, it is also connected to the unknown data-generating distribution p_{data} via the normalized weight function $\tilde{w}$ and hence allows a generative model p_θ to be learned from the offline data examples. Operationally, we would like to train a generative model p_θ such that $p_\theta \approx q_{\text{target}}$. But what would be a suitable choice for the normalized weight function $\tilde{w}$?

In this section, we shall focus on training a *score-based* model, which is mainly motivated by the following connection between the *score function* of the hypothetical target distribution q_{target} and the *gradient* of the unknown objective function f:

$$\boldsymbol{s}_{\text{target}}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log q_{\text{target}}(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} \log [\tilde{w}(f(\boldsymbol{x})) p_{\text{data}}(\boldsymbol{x})] = \boldsymbol{s}_{\text{data}}(\boldsymbol{x}) + \frac{\tilde{w}'(f(\boldsymbol{x}))}{\tilde{w}(f(\boldsymbol{x}))} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}),$$

where $\boldsymbol{s}_{\text{target}}$ and $\boldsymbol{s}_{\text{data}}$ are the score functions of q_{target} and p_{data}, respectively. If $\tilde{w}$ is monotone *increasing*, the derivative $\tilde{w}'(f(\boldsymbol{x})) > 0$ for all $\boldsymbol{x} \in \mathcal{X}$. In this case, the score function $\boldsymbol{s}_{\text{target}}$ is aligned with the *gradient* of the objective function f, so sampling along the direction of $\boldsymbol{s}_{\text{target}}$ will naturally produce samples with high objective values.

In practice, the normalized weight function can be either *pre-selected* or *learned* from the offline data examples. In the former case, as we have seen, the generative model p_θ can be learned via a loss function that is simply the *weighted* version of the loss function for training a *standard* generative model. In the latter case, however, *a priori*, it is unclear how to construct a loss function that would allow us to *jointly* learn a generative model p_θ and a normalized weight function $\tilde{w}$.

2.2 Jointly Learning a Generative Model and a Normalized Weight Function

To address the above challenge, in this section we shall start with the following *natural* optimization objective $J_{\text{opt}}(\theta)$ for identifying a desired generative model p_θ. The above natural optimization objective, however, *cannot* be evaluated for any θ, because the objective function f is *unknown*. Instead of trying to learn a surrogate on f and then use it to guide the training of the generative model, here we consider the more learning-theoretic approach of constructing a *probably approximately correct (PAC)* [42] bound on J_{opt}. Unlike J_{opt}, which depends only on θ, the PAC bound depends on both θ and the normalized weight function $\tilde{w}$. As we shall see, not only it captures *both* the utility and learnability considerations for selecting $\tilde{w}$, it will also *naturally* suggest a surrogate loss function, from which both a generative model p_θ and a normalized weight function $\tilde{w}$ can be jointly learned from the offline data examples.

To construct an appropriate loss function, let us assume without loss of generality that our goal of optimization is to *maximize* an unknown objective function f. Our proposed loss function is based on the following PAC *lower* bound on the natural optimization objective $J_{\text{opt}}(\theta)$:

$$\begin{aligned} J_{\text{opt}}(\theta) \geq \underbrace{\hat{J}_{\boldsymbol{x}_{[m]}}(\tilde{w})}_{\text{empirical utility}} - \underbrace{c_0 K \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})}}_{\text{empirical weighted DSM loss}} - \underbrace{c_0 K \sqrt{2\Delta} \sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})}}_{\text{empirical variance}} - \\ c_1 K W_2(\bar{q}_{\text{target}}, \mathcal{N}) - c_0 K \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - O\left(1/\sqrt[8]{m}\right). \end{aligned} \quad (1)$$

Table 1: Experimental results on the benchmark datasets [↑].

	Supercond.	TFBind8	GFP	UTR	Fluores.	Ave. Improv.	#Best
$\mathcal{D}_{\text{best}}$	0.399	0.439	0.789	0.593	0.485		
Grad	0.483±0.021	**0.985±0.007**	0.053±0.002	0.657±0.039	**0.747±0.209**	0.234	2/5
COMs	0.481±0.017	0.918±0.027	0.864±0.000	**0.683±0.009**	**0.740±0.064**	0.414	2/5
CbAS	0.491±0.028	0.868±0.076	0.864±0.000	0.659±0.009	0.574±0.020	0.320	0/5
$\psi = 20$	0.423±0.044	0.903±0.050	**0.865±0.000**	**0.693±0.006**	**0.803±0.057**	0.407	3/5
$\alpha = 0.25$	**0.537±0.045**	0.941±0.034	**0.865±0.000**	**0.693±0.013**	**0.809±0.078**	**0.485**	**4/5**

Incorporating proper changes to the PAC lower bound (1) leads to the following objective for jointly learning a (un-normalized) weight function w_ϕ and a score-function model $\boldsymbol{s}_t^\theta$:

$$J_{\alpha,\lambda}(\theta,\phi) =$$

$$\frac{1}{m}\sum_{i=1}^{m}\frac{w_\phi(f(\boldsymbol{x}_i))f(\boldsymbol{x}_i)}{\hat{Z}_\phi} - \lambda\sqrt{\frac{1}{m}\sum_{i=1}^{m}\frac{w_\phi(f(\boldsymbol{x}_i))\ell_{\text{DSM}}^{\theta}(\boldsymbol{x}_i)}{\hat{Z}_\phi}} - \alpha\sqrt[4]{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{w_\phi(f(\boldsymbol{x}_i))}{\hat{Z}_\phi} - 1\right)^2}. \quad (2)$$

Details regarding the PAC lower bound (1), the surrogate loss function (2), as well as the training and sampling algorithms are elaborated in A.4, A.5, and A.6, respectively.

3 Experimental Results

We assess the performance of the proposed learning algorithm using the four standard tasks (Superconductor, TF Bind 8, GFP, and UTR) from the Design-Bench benchmark [46]. In addition, we have included the “Fluorescence" task from Fannjiang et al. [18] for a comprehensive evaluation.

Evaluation. We generated a total of $N = 128$ designs for each task and subsequently computed the mean and standard deviation of the 100th percentile of the normalized ground truth over eight independent trials.

Results. The results are listed in Table 1, where $\mathcal{D}_{\text{best}}$ denotes the normalized maximum objective value among the initial samples; “Grad" and “COMs" refer to [45]; “CbAS" refers to [11]; “$\psi = 20$" refers to our proposed approach with a pre-selected weight function $w(y) = \exp(\psi y)$ for $\psi = 20$; and “$\alpha = 0.25$" refers to our proposed approach with a learnable weight function and hyper-parameters $\alpha = 0.25$, $\lambda = 0.1$. Results that fall within one standard deviation of the best performance are highlighted in bold.

Note that across all tasks, our proposed approaches demonstrate not only notable improvement over the best initial samples, but also *consistently competitive* performances against the other three prominent offline optimization algorithms. Quantitatively, our proposed approach with a learnable weight function achieves the *highest* average improvement over all five tasks, where the improvement over a specific task is defined as $(y - \mathcal{D}_{\text{best}})/\mathcal{D}_{\text{best}}$. We believe that this superior consistency is rooted in our *modeling* perspective and the *theoretically-grounded* design of the learning algorithm.

4 Conclusion

In this paper, we introduced a novel generative approach to offline optimization by shifting the focus from traditional function modeling to distribution modeling. This approach leverages the concept of sampling from a generative model rather than optimizing a surrogate of the unknown objective function. We proposed a PAC-bound framework that enables the joint learning of a generative model and a weight function, ensuring that the learning process is aligned with the optimization goals. Our empirical results, validated on standard offline optimization benchmarks, demonstrate the robustness and competitive performance of the proposed method. Future research could explore further refinements of the PAC-bound framework and its applications to more diverse optimization problems.

References

[1] L. Ambrosio, N. Gigli, and G. Savaré. *The Optimal Transportation Problem*, pages 133–149. Springer Science & Business Media, 2005.

[2] L. Ambrosio, E. Brué, and D. Semola. *Lecture 3: The Kantorovich–Rubinstein Duality*, pages 23–34. Springer International Publishing, 2021.

[3] B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[5] C. Audet and W. Hare. *Introduction: Tools and Challenges in Derivative-Free and Blackbox Optimization*, pages 3–14. Springer International Publishing, 2017.

[6] A. Beck. *First-Order Methods in Optimization*. SIAM, 2017.

[7] C. Beckham and C. Pal. Conservative objective models are a special kind of contrastive divergence-based energy model. *arXiv preprint arXiv:2304.03866*, 2023.

[8] C. Beckham, A. Piche, D. Vazquez, and C. Pal. Exploring validation metrics for offline model-based optimisation. *arXiv preprint arXiv:2304.03866*, 2023.

[9] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[10] V. I. Bogachev, N. V. Krylov, M. Röckner, and S. V. Shaposhnikov. *Fokker–Planck–Kolmogorov Equations*, volume 207. American Mathematical Society, 2022.

[11] D. Brookes, H. Park, and J. Listgarten. Conditioning by adaptive sampling for robust design. In *International Conference on Machine Learning*, pages 773–782. PMLR, 2019.

[12] J. Byrd and Z. Lipton. What is the effect of importance weighting in deep learning? In *International conference on machine learning*, pages 872–881. PMLR, 2019.

[13] C. Chen, Y. Zhang, J. Fu, X. S. Liu, and M. Coates. Bidirectional learning for offline infinite-width model-based optimization. *Advances in Neural Information Processing Systems*, 35: 29454–29467, 2022.

[14] C. Chen, C. Beckham, Z. Liu, X. Liu, and C. Pal. Parallel-mentoring for offline model-based optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=tJwyg9Zg9G`.

[15] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[16] R. Durrett. *Probability: theory and examples*, volume 49, pages 24–25. Cambridge university press, 2019.

[17] C. Fannjiang and J. Listgarten. Autofocused oracles for model-based design. *Advances in Neural Information Processing Systems*, 33:12945–12956, 2020.

[18] C. Fannjiang, S. Bates, A. N. Angelopoulos, J. Listgarten, and M. I. Jordan. Conformal prediction under feedback covariate shift for biomolecular design. *Proceedings of the National Academy of Sciences*, 119(43):e2204569119, 2022.

[19] J. Fu and S. Levine. Offline model-based optimization via normalized maximum likelihood estimation. In *International Conference on Learning Representations*, 2021.

[20] A. L. Gibbs and F. E. Su. On choosing and bounding probability metrics. *International statistical review*, 70(3):419–435, 2002.

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

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

[23] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[24] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[25] W. Hoburg and P. Abbeel. Geometric programming for aircraft design optimization. *AIAA Journal*, 52(11):2414–2426, 2014.

[26] A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

[27] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[28] I. Karatzas and S. Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media, 2012.

[29] N. Killoran, L. J. Lee, A. Delong, D. Duvenaud, and B. J. Frey. Generating and designing DNA with deep generative models. *arXiv preprint arXiv:1712.06148*, 2017.

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[32] S. Kolli, A. X. Lu, X. Geng, A. Kumar, and S. Levine. Data-driven optimization for protein design: Workflows, algorithms and metrics. In *ICLR2022 Machine Learning for Drug Discovery*, 2022.

[33] S. Krishnamoorthy, S. M. Mashkaria, and A. Grover. Diffusion models for black-box optimization, 2023. URL `https://arxiv.org/abs/2306.07180`.

[34] S. Krishnamoorthy, S. M. Mashkaria, and A. Grover. Generative pretraining for black-box optimization, 2023. URL `https://arxiv.org/abs/2206.10786`.

[35] A. Kumar and S. Levine. Model inversion networks for model-based optimization. *Advances in Neural Information Processing Systems*, 33:5126–5137, 2020.

[36] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[37] A. Kumar, A. Yazdanbakhsh, M. Hashemi, K. Swersky, and S. Levine. Data-driven offline optimization for architecting hardware accelerators. In *International Conference on Learning Representations*, 2022.

[38] D. Kwon, Y. Fan, and K. Lee. Score-based generative modeling secretly minimizes the wasserstein distance. *Advances in Neural Information Processing Systems*, 35:20205–20217, 2022.

[39] T. Liao, G. Wang, B. Yang, R. Lee, K. Pister, S. Levine, and R. Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 International Conference on Robotics and Automation*, pages 2488–2494, 2019.

[40] F. Mazé and F. Ahmed. Topodiff: A performance and constraint-guided diffusion model for topology optimization. *arXiv preprint arXiv:2208.09591*, 2022.

[41] H. Qi, Y. Su, A. Kumar, and S. Levine. Data-driven offline decision-making via invariant representation learning. *Advances in Neural Information Processing Systems*, 35:13226–13237, 2022.

[42] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[43] Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[44] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[45] B. Trabucco, A. Kumar, X. Geng, and S. Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021.

[46] B. Trabucco, X. Geng, A. Kumar, and S. Levine. Design-bench: Benchmarks for data-driven offline model-based optimization. In *International Conference on Machine Learning*, pages 21658–21676. PMLR, 2022.

[47] M. Vidyasagar. *Learning and generalisation: with applications to neural networks*. Springer Science & Business Media, 2013.

[48] P. Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[49] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[50] H. Yuan, K. Huang, C. Ni, M. Chen, and M. Wang. Reward-directed conditional diffusion: Provable distribution estimation and reward improvement. *arXiv preprint arXiv:2307.07055*, 2023.

Appendices

Contents

List of Symbols

The next list describes several symbols that are used within the entire body of the paper.

α, λ	Hyper-parameters (learnable case)
ℓ_{DSM}	Denoising score matching loss function
$\hat{\mathfrak{R}}$	Rademacher complexity
$\mathcal{D}_{\text{best}}$	Maximum objective value within the offline dataset
$\mathcal{N}$	Normal distribution
p_{data}	Data-generating distribution
ψ	Hyper-parameter (pre-selected case)
q_{target}	Target distribution (hypothetical)
$\tilde{w}$	Normalized weight function
$\boldsymbol{x}$	Domain point, Design or Feature vector
$\boldsymbol{x}_{[m]}$	A set of offline domain points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$
$\boldsymbol{x}_i$	Offline domain point
f	Unknown objective function
$f(\boldsymbol{x}_i)$	Offline objective value
f_θ	Parameterized surrogate objective function
J_{opt}	Natural optimization objective
p_θ	Parameterized generative model
w_ϕ	Parameterized weight function (unnormalized)
W_p	p-Wasserstein distance
y	Objective value, Label or Score

A Technical Details

A.1 The denoising diffusion probabilistic model (DDPM)

For score-based generative models, we are particularly interested in the *denoising diffusion probabilistic model (DDPM)* [44, 24] due to its stability and performance on high-dimensional datasets. Below we first recall a few essential results on the DDPM.

Consider a *forward* process of continuously injecting white Gaussian noise into a signal $\mathbf{x}_t$:

$$d\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t dt + \sqrt{\beta(t)}d\mathbf{w}_t, \quad t \in [0, 1], \tag{3}$$

where $\beta : [0, 1] \to \mathbb{R}_{++}$ is a positive *noise scheduler*, $\mathbf{w}_t$ is a standard *Wiener* process [28], and time in this process is assumed to flow in the *forward* direction from $t = 0$ to $t = 1$. Denote by q_t the marginal distribution of $\mathbf{x}_t$ from the forward process (3). The DDPM is mainly motivated by the fact that the marginal distributions q_t, $t \in [0, 1]$, can be *recovered* through the following *reverse* process [3]:

$$d\mathbf{x}_t = -\beta(t)\left(\frac{1}{2}\mathbf{x}_t + \boldsymbol{s}_t^\theta(\mathbf{x}_t)\right)dt + \sqrt{\beta(t)}d\bar{\mathbf{w}}_t, \tag{4}$$

where $\boldsymbol{s}_t^\theta$ is a model of the score function of q_t and $\bar{\mathbf{w}}_t$ is (again) a standard *Wiener* process but with time flowing *backward* from $t = 1$ to $t = 0$. More specifically, let p_1 be the initial distribution of $\mathbf{x}_1$ from the reverse process (4) and let p_t^θ be the marginal distribution of $\mathbf{x}_t$, $t \in [0, 1)$ from the reverse process (4). If we let $p_1 = q_1$ and $\boldsymbol{s}_t^\theta$ be the *exact* score function of q_t for all $t \in [0, 1]$, we have $p_t^\theta = q_t$ for *all* $t \in [0, 1)$ [10]. In particular, the initial distribution of the forward process q_0 can be recovered at the end of the reverse process via the score functions of q_t, $t \in [0, 1]$. Thus, to learn the initial distribution of the forward process q_0, it suffices to learn a model $\boldsymbol{s}_t^\theta$ that approximates the score functions of q_t for all $t \in [0, 1]$.

There are several methods [26, 48, 43] that allow a model $\boldsymbol{s}_t^\theta$ to be learned from a training dataset drawn from q_0. Here we focus on the *denoising score matching (DSM)* method due to its scalability to large datasets. For the DSM method, a model $\boldsymbol{s}_t^\theta$ is learned by minimizing the following DSM loss:

$$\mathcal{L}_{\text{DSM}}(\theta; q_0) := \mathbb{E}_{\mathbf{x}\sim q_0}\left[\ell_{\text{DSM}}^\theta(\mathbf{x})\right], \tag{5}$$

where

$$\ell_{\text{DSM}}^\theta(\mathbf{x}) := \int_0^1 \lambda(t)\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\boldsymbol{I})}\left[\left\|\boldsymbol{s}_t^\theta(\mathbf{x}_t) + \frac{\mathbf{z}}{\sigma(t)}\right\|^2\right]dt \tag{6}$$

is the *point-wise* DSM loss, $\mathbf{x}_t = \sqrt{1-\sigma(t)^2}\mathbf{x} + \sigma(t)\mathbf{z}$, $\lambda : [0, 1] \to \mathbb{R}_{++}$ can be any positive function, and $\sigma(t) = \sqrt{1 - \exp\left[-\int_0^t \beta(s)ds\right]}$.

While the previous DSM loss can be easily estimated from a dataset drawn from q_0 and hence is very *conductive* to learning, *a priori* it is unclear how it would connect to any *generative* loss between q_0 and p_0^θ. Interestingly, it was shown in Theorem 2 and Corollary 3 of Kwon et al. [38] that under some (relatively) mild conditions[4] on β, q_0, and $\boldsymbol{s}_\theta$, by choosing $\lambda(t) = \beta(t)$ we have

$$W_2(q_0, p_0^\theta) \leq c_0\sqrt{\mathbb{E}_{\mathbf{x}\sim q_0}\left[\ell_{\text{DSM}}^\theta(\mathbf{x})\right]} + c_1 W_2(q_1, p_1), \tag{7}$$

where $W_2(q, p)$ is the 2-Wasserstein distance [1] between the probability distributions q and p, and c_0 and c_1 are constants that only depend on the choice of the noise scheduler β and some prior knowledge on p_0 and $\boldsymbol{s}_t^\theta$ but is independent of the model parameter θ. In Kwon et al. [38], this result was coined as "score-based generative modeling *secretly* minimizes the Wasserstein distance".

[4]The readers are referred to Section 3.1 of Kwon et al. [38] for the assumptions under which the inequality (7) holds.

A.2 Weighted learning

To learn a generative model $p_\theta \approx q_{\rm target}$, we shall let $q_0 = q_{\rm target}$ and p_1 be the standard multivariate Gaussian distribution $\mathcal{N}$. If we denote q_1 and p_0^θ by $\bar{q}_{\text{target}}$ and p_θ respectively, by (7) we have

$$W_2(q_{\rm target}, p_\theta) \leq c_0 \sqrt{\mathbb{E}_{\mathbf{x}\sim q_{\rm target}}\left[\ell^\theta_{\text{DSM}}(\mathbf{x})\right]} + c_1 W_2(\bar{q}_{\text{target}}, \mathcal{N}). \tag{8}$$

We mention here that the output distribution of the forward process $\bar{q}_{\text{target}}$ is potentially dependent on the choice of $\tilde{w}$, even though this dependency is not explicit from the notation. In practice, however, $\bar{q}_{\text{target}}$ can be made very close to the standard multivariate Gaussian distribution $\mathcal{N}$ with an appropriate choice of the noise scheduler β. Therefore, the Wasserstein distance $W_2(\bar{q}_{\text{target}}, \mathcal{N})$ is very small and is usually disregarded from the learning process.

To estimate the DSM loss $\mathbb{E}_{\mathbf{x}\sim q_{\rm target}}\left[\ell^\theta_{\text{DSM}}(\mathbf{x})\right]$ from the offline data examples, note that by the definition of $q_{\rm target}$, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim q_{\rm target}}\left[\ell^\theta_{\text{DSM}}(\mathbf{x})\right] &= \mathbb{E}_{\mathbf{x}\sim p_{\rm data}}\left[\tilde{w}(f(\mathbf{x}))\ell^\theta_{\text{DSM}}(\mathbf{x})\right] &(9)\\ &\approx \frac{1}{m}\sum_{i=1}^{m} \tilde{w}(f(\boldsymbol{x}_i))\ell^\theta_{\text{DSM}}(\boldsymbol{x}_i), &(10)\end{aligned}$$

where (9) is also known as "re-weighting" or *importance sampling* [12]. By (8), minimizing the empirical weighted DSM loss (10) can help to identify a generative model $p_\theta \approx q_{\rm target}$ for a pre-selected normalized weight function $\tilde{w}$.

Finally, we note that scaling the normalized weight function $\tilde{w}$ does *not* change the optimal solution that minimizes the empirical weighted DSM loss (10). Therefore, when using (10) to identify a generative model p_θ, one can use an *un-normalized* weight function instead of a normalized one.

A.3 Trade-off between utility and learnability

Intuitively, there are two considerations for selecting a normalized weight function $\tilde{w}$. On one hand, from the *utility* point of view, we would like to choose $\tilde{w}$ such that the hypothetical target distribution $q_{\rm target}$ focuses most of its densities on the domain points with *superior* objective values. This can be achieved, for example, by choosing $\tilde{w}$ to be heavily *skewed* towards superior objective values. On the other hand, from the *learning* viewpoint, the generative model p_θ is learned from the offline observations, which were generated from the unknown data-generating distribution $p_{\rm data}$. If $\tilde{w}$ is chosen to be heavily skewed, the hypothetical target distribution $q_{\rm target}$ then becomes very *different* from the data-generating distribution $p_{\rm data}$. In this case, learning the generative model p_θ from the offline data examples may be subject to very high *sample* complexity.

A.4 The main theorem

Our proposed loss function is based on the following PAC *lower* bound on the natural optimization objective:

$$J_{\text{opt}}(\theta) = \mathbb{E}_{\mathbf{x}\sim p_\theta}[f(\mathbf{x})]. \tag{11}$$

Theorem A.1. *Assume that: i) the unknown objective function f is K-Lipschitz and satisfies $|f(\boldsymbol{x})| \leq F$ for all $\boldsymbol{x} \in \mathcal{X}$; ii) the generative model p_θ is a DDPM; iii) the point-wise DSM loss ℓ^θ_{DSM} satisfies $0 \leq \ell^\theta_{DSM}(\boldsymbol{x}) \leq \Delta$ for all $\boldsymbol{x} \in \mathcal{X}$ and all $\theta \in \Theta$; and iv) the conditions from Section 3.1 of Kwon et al. [38] on the noise scheduler β, the data-generating distribution $p_{\rm data}$, the normalized weight function $\tilde{w}$, and the score-function model $\boldsymbol{s}^\theta_t$ are satisfied. Let $\tilde{\mathcal{W}}$ be the collection of all normalized weight functions $\tilde{w}$ that are L-Lipschitz and satisfy $0 \leq \tilde{w}(y) \leq B$ for any $y \in [-F, F]$. With probability $\geq 1-\delta$, we have for* any *$\tilde{w} \in \tilde{\mathcal{W}}$ and* any *$\theta \in \Theta$,*

$$\begin{aligned}J_{opt}(\theta) \geq \underbrace{\hat{J}_{\boldsymbol{x}_{[m]}}(\tilde{w})}_{\text{empirical utility}} - \underbrace{c_0 K \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})}}_{\text{empirical weighted DSM loss}} - \underbrace{c_0 K \sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})}}_{\text{empirical variance}} - \\ c_1 K W_2(\bar{q}_{target}, \mathcal{N}) - c_0 K \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - O\left(1/\sqrt[8]{m}\right),\end{aligned} \tag{12}$$

where

$$\hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m} \tilde{w}(f(\boldsymbol{x}_i))f(\boldsymbol{x}_i) \tag{13}$$

is the empirical *utility of* $\tilde{w}$*,*

$$\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) = \frac{1}{m}\sum_{i=1}^{m} \tilde{w}(f(\boldsymbol{x}_i))\ell^{\theta}_{DSM}(\boldsymbol{x}_i) \tag{14}$$

is the empirical *weighted DSM loss of* $\boldsymbol{s}^{\theta}_t$*,*

$$\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m} (\tilde{w}(f(\boldsymbol{x}_i)) - 1)^2 \tag{15}$$

is the empirical *variance of* $\tilde{w}$*,*

$$\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}}\left[\sup_{\theta\in\Theta} \frac{1}{m}\sum_{i=1}^{m} \sigma_i \ell^{\theta}_{DSM}(\boldsymbol{x}_i)\right] \tag{16}$$

is the empirical *Rademacher complexity with respect to the parameter family* Θ*, and the last term* $O\left(1/\sqrt[8]{m}\right)$ *is independent of the model parameter* θ *and* $\tilde{w}$*.*

The proof of the above theorem is long and technical and hence is deferred to next section to enhance the flow of the paper. We mention here that the proof utilizes several key technical results including the *duality* theorem of Kantorovich-Rubenstein [2] for the 1-Wasserstein distance [4], the fact that "score-based generative modeling *secretly* minimizes the Wasserstein distance" [38], the Rademacher bound for *bounded* functions [42], the Wasserstein contraction property [9], and the covering bound for *Lipschitz* functions [47].

Note that to maximize the PAC lower bound (1), we need to simultaneously maximize the *utility* of $\tilde{w}$ and minimize the *weighted DSM loss* of $\boldsymbol{s}^{\theta}_t$ and the *variance* of $\tilde{w}$. Therefore, the PAC lower bound (1) captures both the utility and learnability considerations for selecting a normalized weight function $\tilde{w}$.

A.5 From PAC lower bound to surrogate loss function

To jointly learn a generate model p_θ and a normalized function $\tilde{w}$, first note that the last two terms of the PAC lower bound (1) are independent of θ and $\tilde{w}$ and hence can be ignored from the learning objective. The forth term is due to the "initial" sampling error of the reverse process. As discussed previously in Section A.2, while this term is potentially dependent on the normalized weight function $\tilde{w}$, in practice it can be made very small by choosing an appropriate noise scheduler β and hence will be ignored from our learning objective.

To make the first three terms *learnable*, we consider the following two modifications to the bound.

First, the coefficients $c_0 K$ and $c_0 K\sqrt{2\Delta}$ in the second and the third term require some prior knowledge on the unknown data-generating distribution p_{data} and the unknown objective function f. In practice, we replace them by two *hyper-parameters* λ and α, respectively. We mention here that the hyper-parameter α plays a particular important role in the learning objective, as it controls the utility-learnability tradeoff for selecting a normalized weight function $\tilde{w}$.

Second, the weight function $\tilde{w}$ needs to be *normalized* with respect to the unknown data-generating distribution p_{data} and the unknown objective function f. In practice, we let $\tilde{w}(\cdot) = \frac{w_\phi(\cdot)}{Z_\phi}$, where w_ϕ is an *un-normalized* weight function parameterized by a second parameter ϕ, and $Z_\phi = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[w_\phi(f(\mathbf{x}))]$ is the normalizing constant. While the exact calculation of Z_ϕ again requires the knowledge of p_{data} and f, in practice it can be easily estimated from the offline data examples as $\hat{Z}_\phi = \frac{1}{m}\sum_{i=1}^{m} w_\phi(f(\boldsymbol{x}_i))$.

A.6 Algorithms

Algorithm 1 TRAINING

1: **Input**: Offline examples $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$; hyper-parameters α, λ; learning-rate parameters η_1, η_2.
2: **General step**:
3: $\phi_0 \leftarrow \arg\max_{\phi\in\Phi} \left\{ \frac{1}{m}\sum_{i=1}^m \frac{w_\phi(f(\boldsymbol{x}_i))f(\boldsymbol{x}_i)}{\hat{Z}_\phi} - \alpha \sqrt[4]{\frac{1}{m}\sum_{i=1}^m \left(\frac{w_\phi(f(\boldsymbol{x}_i))}{\hat{Z}_\phi} - 1\right)^2} \right\}$ ▷ via GD
4: $\theta_0 \leftarrow \arg\min_{\theta\in\Theta} \left\{ \frac{1}{m}\sum_{i=1}^m w_{\phi_0}(f(\boldsymbol{x}_i)) \cdot \ell^{\theta}_{\text{DSM}}(\boldsymbol{x}_i) \right\}$ ▷ via SGD
5: **for** $k = 0$ **to** $K-1$ **do**
6: $\quad\phi_{k+1} \leftarrow \phi_k + \eta_1 \cdot \nabla_\phi J_{\alpha,\lambda}(\theta_k, \phi_k)$ ▷ via GD
7: $\quad\theta_{k+1} \leftarrow \theta_k - \eta_2 \cdot \nabla_\theta \left\{ \frac{1}{m}\sum_{i=1}^m w_{\phi_{k+1}}(f(\boldsymbol{x}_i)) \cdot \ell^{\theta_k}_{\text{DSM}}(\boldsymbol{x}_i) \right\}$ ▷ via SGD
8: **end for**
9: **Output**: Model parameters $(\phi^*, \theta^*) = (\phi_K, \theta_K)$.

Algorithm 2 SAMPLING/OPTIMIZATION

1: **Input**: Score function model $\boldsymbol{s}_t^{\theta^*}(\boldsymbol{x})$, number of samples N, number of steps T, noise scheduler parameters $(\beta_{\min}, \beta_{\max})$, and $\tilde{\beta}(t) = \frac{1}{T}\left[\beta_{\min} + \frac{t}{T}(\beta_{\max} - \beta_{\min})\right]$.
2: **General step**:
3: Draw N samples $\boldsymbol{x}_T^{(1)}, \boldsymbol{x}_T^{(2)}, \ldots, \boldsymbol{x}_T^{(N)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \boldsymbol{I})$
4: **for** $n = 1$ **to** N **do**
5: $\quad$**for** $t = T$ **to** 1 **do**
6: $\quad\quad\boldsymbol{x}_{t-1}^{(n)} \leftarrow \left(2 - \sqrt{1 - \tilde{\beta}(t)}\right) \cdot \boldsymbol{x}_t^{(n)} + \frac{1}{2}\tilde{\beta}(t) \cdot \boldsymbol{s}_{t/T}^{\theta^*}(\boldsymbol{x}_t^{(n)})$
7: $\quad$**end for**
8: **end for**
9: **Output**: Optimized samples $\boldsymbol{x}_0^{(1)}, \boldsymbol{x}_0^{(2)}, \ldots, \boldsymbol{x}_0^{(N)}$.

B Proof of the Main Theorem

In this section, we introduce a few technical results, which will lead to the proof of the main theorem.

B.1 Wasserstein distance

Let μ and ν be two probability distributions on $\mathbb{R}^d$. A *coupling* γ between μ and ν is a joint distribution on $\mathbb{R}^d \times \mathbb{R}^d$ whose marginals are μ and ν. The p-Wasserstein distance between μ and ν (with respect to the Euclidean norm) is given by [1]:

$$W_p(\mu, \nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \mathbb{E}_{(\mathbf{x},\tilde{\mathbf{x}})\sim\gamma}\left[\|\mathbf{x} - \tilde{\mathbf{x}}\|^p\right] \right)^{1/p}, \tag{17}$$

where $\Gamma(\mu, \nu)$ is the set of all couplings between μ and ν, and $\|\cdot\|$ denotes the standard Euclidean norm.

The 1-Wasserstein distance, also known as the *earth mover's distance*, has an important equivalent representation that follows from the *duality* theorem of Kantorovich-Rubenstein [2]:

$$W_1(\mu, \nu) = \frac{1}{K} \sup_{\|\tilde{f}\|_{\text{Lip}} \le K} \left\{ \mathbb{E}_{\mathbf{x}\sim\mu}[\tilde{f}(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\nu}[\tilde{f}(\mathbf{x})] \right\}, \tag{18}$$

where $\|\cdot\|_{\text{Lip}}$ denotes the *Lipschitz* norm. In our construction, this *dual* representation of the 1-Wasserstein distance serves as the bridge between the *objective-specific* generative loss and the *generic* generative loss.

By the standard Jensen's inequality [16], we also have

$$W_1(\mu, \nu) \leq W_2(\mu, \nu) \tag{19}$$

for any two distributions μ and ν. As we shall see, this simple relationship between the 1-Wasserstein and 2-Wasserstein distances can help to further connect the *objective-specific* generative loss to the *DSM* loss for training a *score-based* generative model.

B.2 Generalization bound for weighted learning

Let $\ell_\theta : \mathcal{X} \to \mathbb{R}$ be a *bounded* loss function parameterized by $\theta \in \Theta$ such that $0 \leq \ell_\theta(\boldsymbol{x}) \leq \Delta$ for all $\boldsymbol{x} \in \mathcal{X}$ and all $\theta \in \Theta$. Consider the problem of estimating the expected *weighted* loss $\mathcal{L}_p(\theta, \tilde{w}) = \mathbb{E}_{\mathbf{x}\sim p}[\tilde{w}(\mathbf{x})\ell_\theta(\mathbf{x})]$, where $\tilde{w} : \mathcal{X} \to \mathbb{R}_+$ is a normalized, *bounded* weight function such that $\mathbb{E}_{\mathbf{x}\sim p}[\tilde{w}(\mathbf{x})] = 1$ and $0 \leq \tilde{w}(\boldsymbol{x}) \leq B$ for all $\boldsymbol{x} \in \mathcal{X}$. We have the following PAC upper bound, with respect to the parameter family Θ, on the expected weighted loss $\mathcal{L}_p(\theta, \tilde{w})$ for any given $\tilde{w}$.

Lemma B.1. *For any given $\tilde{w}$, with probability $\geq 1 - \delta$ we have for* any $\theta \in \Theta$

$$\mathcal{L}_p(\theta, \tilde{w}) \leq \hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) + 2\Delta\sqrt{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} + 2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) + 3\sqrt{\frac{2B\Delta\log(2/\delta)}{m}}, \tag{20}$$

where $\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) = \frac{1}{m}\sum_{i=1}^m \tilde{w}(\boldsymbol{x}_i)\ell_\theta(\boldsymbol{x}_i)$ is the empirical *weighted loss over the training dataset $\boldsymbol{x}_{[m]}$, $\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^m (\tilde{w}(\boldsymbol{x}_i) - 1)^2$ is the* empirical *variance of $\tilde{w}$ over $\boldsymbol{x}_{[m]}$, and $\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}}\left[\sup_{\theta\in\Theta} \frac{1}{m}\sum_{i=1}^m \sigma_i\ell_\theta(\boldsymbol{x}_i)\right]$ is the* empirical *Rademacher complexity with respect to the parameter family Θ over $\boldsymbol{x}_{[m]}$.*

Proof. By assumption, we have $0 \leq \tilde{w}(\boldsymbol{x}) \leq B$ for any $\boldsymbol{x} \in \mathcal{X}$ and $0 \leq \ell_\theta(\boldsymbol{x}) \leq \Delta$ for any $\boldsymbol{x} \in \mathcal{X}$ and any $\theta \in \Theta$. It follows immediately that the *weighed* loss function $\tilde{w}(\boldsymbol{x})\ell_\theta(\boldsymbol{x})$ satisfies $0 \leq \tilde{w}(\boldsymbol{x})\ell_\theta(\boldsymbol{x}) \leq B\Delta$ for any $\boldsymbol{x} \in \mathcal{X}$ and any $\theta \in \Theta$. Applying the standard Rademacher bound to the *weighted* loss function class $(\tilde{w}(\boldsymbol{x})\ell_\theta(\boldsymbol{x}) : \theta \in \Theta)$, with probability $\geq 1 - \delta$ we have for any $\theta \in \Theta$

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\tilde{w}(\mathbf{x})\ell_\theta(\mathbf{x})\right] \leq \hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) + 2\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta) + 3\sqrt{\frac{2B\Delta\log(2/\delta)}{m}}, \tag{21}$$

where $\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) = \frac{1}{m}\sum_{i=1}^m \tilde{w}(\boldsymbol{x}_i)\ell_\theta(\boldsymbol{x}_i)$ is the empirical weighted loss over $\boldsymbol{x}_{[m]}$, and $\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}}\left[\sup_{\theta\in\Theta} \frac{1}{m}\sum_{i=1}^m \sigma_i\tilde{w}(\boldsymbol{x})\ell_\theta(\boldsymbol{x}_i)\right]$ is the empirical *weighted* Rademacher complexity [42] with respect to the parameter family Θ over $\boldsymbol{x}_{[m]}$. The empirical weighted Rademacher complexity $\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta)$ can be further bounded from above as:

$$\begin{aligned}
\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta) &= \mathbb{E}_{\boldsymbol{\sigma}_{[m]}}\left[\sup_{\theta\in\Theta}\frac{1}{m}\sum_{i=1}^m \sigma_i\left(w(\boldsymbol{x}_i) - 1 + 1\right)\ell_\theta(\boldsymbol{x}_i)\right] \\
&\leq \mathbb{E}_{\sigma_{[m]}}\left[\sup_{\theta\in\Theta}\frac{1}{m}\sum_{i=1}^m \sigma_i(\tilde{w}(\boldsymbol{x}_i) - 1)\ell_\theta(\boldsymbol{x}_i)\right] + \mathbb{E}_{\boldsymbol{\sigma}_{[m]}}\left[\sup_{\theta\in\Theta}\frac{1}{m}\sum_{i=1}^m \sigma_i\ell_\theta(\boldsymbol{x}_i)\right]
\end{aligned} \tag{22}$$

Further note that

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^m \sigma_i(\tilde{w}(\boldsymbol{x}_i) - 1)\ell_\theta(\boldsymbol{x}_i) &\leq \sqrt{\frac{\sum_{i=1}^m(\tilde{w}(\boldsymbol{x}_i) - 1)^2}{m}}\sqrt{\frac{\sum_{i=1}^m\left(\sigma_i\ell_\theta(\boldsymbol{x}_i)\right)^2}{m}} \\
&= \sqrt{\frac{\sum_{i=1}^m(\tilde{w}(\boldsymbol{x}_i) - 1)^2}{m}}\sqrt{\frac{\sum_{i=1}^m \ell_\theta^2(\boldsymbol{x}_i)}{m}} \\
&\leq \Delta\sqrt{\frac{\sum_{i=1}^m(\tilde{w}(\boldsymbol{x}_i) - 1)^2}{m}}
\end{aligned}$$

for any $\theta \in \Theta$ and any realization of $\boldsymbol{\sigma}_{[m]}$, where the first inequality follows from the standard Cauchy-Schwarz inequality, the second equality follows from the fact that the square of a Rademacher variable takes a constant value of 1, and the last inequality follows from the assumption that $0 \leq \ell_\theta(\boldsymbol{x}) \leq \Delta$ for any $\boldsymbol{x} \in \mathcal{X}$ and any $\theta \in \Theta$. It follows immediately that

$$\mathbb{E}_{\sigma_{[m]}}\left[\sup_{\theta\in\Theta}\frac{1}{m}\sum_{i=1}^{m}\sigma_i(\tilde{w}(\boldsymbol{x}_i)-1)\ell_\theta(\boldsymbol{x}_i)\right] \leq \Delta\sqrt{\frac{\sum_{i=1}^{m}(\tilde{w}(\boldsymbol{x}_i)-1)^2}{m}}. \tag{23}$$

Substituting (23) into (22) gives

$$\hat{\mathfrak{R}}^{\tilde{w}}_{\boldsymbol{x}_{[m]}}(\Theta) \leq \Delta\sqrt{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} + \hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta), \tag{24}$$

where $\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m}\left(\tilde{w}(\boldsymbol{x}_i)-1\right)^2$ is the empirical variance of $\tilde{w}$ over $\boldsymbol{x}_{[m]}$, and $\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}}\left[\sup_{\theta\in\Theta}\frac{1}{m}\sum_{i=1}^{m}\sigma_i\ell_\theta(\boldsymbol{x}_i)\right]$ is the empirical (un-weighted) Rademacher complexity with respect to the parameter family Θ over $\boldsymbol{x}_{[m]}$. Substituting (24) into (21) completes the proof of (20) and hence Lemma B.1. $\square$

The main insight from the above lemma is that the *generalization error* (with respect to the parameter θ) between the expected weighted loss $\mathcal{L}_p(\theta, \tilde{w})$ and the empirical weighted loss $\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})$ can be controlled by controlling the *complexity* of the model class Θ and the *variance* of the normalized weight function $\tilde{w}$.

B.3 A distribution-dependent surrogate on the natural optimization objective

Proposition B.2. *Assume that the unknown objective function f is K-Lipschitz and the generative model p_θ is a DDPM. Under the assumptions from Section 3.1 of Kwon et al. [38] on the noise scheduler β, the data-generating distribution p_{data}, the normalized weight function $\tilde{w}$, and the score-function model $\boldsymbol{s}^\theta_t$, we have*

$$J_{opt}(\theta) \geq \underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[\tilde{w}(f(\mathbf{x}))f(\mathbf{x})\right]}_{\text{expected utility}} - \underbrace{c_0K\sqrt{\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[\tilde{w}(f(\mathbf{x}))\ell^\theta_{DSM}(\mathbf{x})\right]}}_{\text{expected weighted DSM loss}} - c_1KW_2(\bar{q}_{target}, \mathcal{N}), \tag{25}$$

where ℓ^θ_{DSM} is the point-wise DSM loss of $\boldsymbol{s}^\theta_t$ as defined in (6), $\bar{q}_{target}$ is the output distribution of the forward process (3), Φ is the standard Gaussian distribution, and c_0 and c_1 are constants that are independent of the model parameter θ and $\tilde{w}$.

Proof. We start by writing $J_{\mathrm{opt}}(\theta)$ as:

$$J_{\mathrm{opt}}(\theta) = \mathbb{E}_{\mathbf{x}\sim q_{\mathrm{target}}}[f(\mathbf{x})] - \left\{\mathbb{E}_{\mathbf{x}\sim q_{\mathrm{target}}}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_\theta}[f(\mathbf{x})]\right\}. \tag{26}$$

By the definition of q_{target}:

$$q_{\mathrm{target}}(\boldsymbol{x}) := \tilde{w}(f(\boldsymbol{x}))\cdot p_{\mathrm{data}}(\boldsymbol{x}), \tag{27}$$

we have

$$\mathbb{E}_{\mathbf{x}\sim q_{\mathrm{target}}}[f(\mathbf{x})] = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[\tilde{w}(f(\mathbf{x}))f(\mathbf{x})\right]. \tag{28}$$

Furthermore,

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim q_{\mathrm{target}}}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_\theta}[f(\mathbf{x})] &\leq \sup_{\|\tilde{f}\|_{\mathrm{Lip}}\leq K}\left\{\mathbb{E}_{\mathbf{x}\sim q_{\mathrm{target}}}[\tilde{f}(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_\theta}[\tilde{f}(\mathbf{x})]\right\} &(29)\\ &= K\cdot W_1(q_{\mathrm{target}}, p_\theta), &(30)\end{aligned}$$

where the first inequality follows directly from the assumption that f is K-Lipschitz, and the second equality follows from the dual representation (30) of the 1-Wasserstein distance.

Under the assumption that p_θ is a DDPM, we can further bound the 1-Wasserstein distance $W_1(q_{\mathrm{target}}, p_\theta)$ as:

$$W_1(q_{\mathrm{target}}, p_\theta) \leq W_2(q_{\mathrm{target}}, p_\theta) \leq c_o\sqrt{\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[\tilde{w}(f(\mathbf{x}))\ell^\theta_{\mathrm{DSM}}(\mathbf{x})\right]} + c_1W_2(\bar{q}_{\mathrm{target}}, \mathcal{N}), \tag{31}$$

where the first inequality follows from (19), and the second inequality follows from (8) and the definition of q_{target} in (27).

Substituting (28), (30), and (31) into (26) completes the proof of (25). $\square$

Next, we shall convert the distribution-dependent surrogate on the right-hand side of (25) into a *PAC* lower bound using the standard *complexity* theory.

B.4 Proof of Theorem A.1

For the proof, we shall write q_{target} and $\bar{q}_{\text{target}}$ as $q^{\tilde{w}}_{\text{target}}$ and $\bar{q}^{\tilde{w}}_{\text{target}}$ respectively, to emphasize their dependencies on the normalized weight function $\tilde{w}$. Let us first recall from Proposition B.2 that the natural optimization objective $J_{\text{opt}}(\theta)$ can be bounded from below as:

$$J_{\text{opt}}(\theta) \geq \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\tilde{w}(f(\mathbf{x}))f(\mathbf{x})\right] - c_0 K\sqrt{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\tilde{w}(f(\mathbf{x}))\ell^{\theta}_{\text{DSM}}(\mathbf{x})\right]} - c_1 K W_2(\bar{q}^{\tilde{w}}_{\text{target}}, \mathcal{N}). \tag{32}$$

To turn the right-hand side into a *PAC* lower bound on $J_{\text{opt}}(\theta)$, let us first *fix* a normalized weight function $\tilde{w} \in \tilde{\mathcal{W}}$.

Given $\tilde{w}$, let us first apply the standard Hoeffding's inequality to obtain a *concentration* lower bound on the expected utility $\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\tilde{w}(f(\mathbf{x}))f(\mathbf{x})]$. More specifically, by assumption we have $|f(\boldsymbol{x})| \leq F$ for any $\boldsymbol{x} \in \mathcal{X}$ and $0 \leq \tilde{w}(y) \leq B$ for any $y \in [-F, F]$. It follows that the *weighed* objective function $\tilde{w}(f(\boldsymbol{x}))f(\boldsymbol{x})$ satisfies $|\tilde{w}(f(\boldsymbol{x}))f(\boldsymbol{x})| \leq BF$ for any $\boldsymbol{x} \in \mathcal{X}$. By Hoeffding's inequality, with probability $\geq 1 - \delta'/2$ we have

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\tilde{w}(f(\mathbf{x}))f(\mathbf{x})\right] \geq \hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) - \sqrt{\frac{BF\log(2/\delta')}{m}}, \tag{33}$$

where $\hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m}\tilde{w}(f(\boldsymbol{x}_i))f(\boldsymbol{x}_i)$ is the empirical utility of $\tilde{w}$. Next by Lemma (B.1), with probability $\geq 1 - \delta'/2$ we have for *any* $\theta \in \Theta$

$$\begin{aligned}&\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\tilde{w}(f(\mathbf{x}))\ell^{\theta}_{\text{DSM}}(\mathbf{x})\right]\\ &\leq \hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) + 2\Delta\sqrt{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} + 2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) + 3\sqrt{\frac{2B\Delta\log(4/\delta')}{m}}\end{aligned}$$

and hence

$$\begin{aligned}&\sqrt{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\tilde{w}(f(\mathbf{x}))\ell^{\theta}_{DSM}(\mathbf{x})\right]}\\ &\leq \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) + 2\Delta\sqrt{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} + 2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) + 3\sqrt{\frac{2B\Delta\log(4/\delta')}{m}}}\\ &\leq \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})} + \sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} + \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} + \sqrt[4]{\frac{18B\Delta\log(4/\delta')}{m}},\end{aligned} \tag{34}$$

where $\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w}) = \frac{1}{m}\sum_{i=1}^{m}\tilde{w}(\boldsymbol{x}_i)\ell^{\theta}_{\text{DSM}}(\boldsymbol{x}_i)$ is the empirical weighted DSM loss of $\boldsymbol{s}^{\theta}_t$ over $\boldsymbol{x}_{[m]}$, $\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w}) = \frac{1}{m}\sum_{i=1}^{m}(\tilde{w}(\boldsymbol{x}_i) - 1)^2$ is the empirical variance of $\tilde{w}$ over $\boldsymbol{x}_{[m]}$, and $\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta) = \mathbb{E}_{\boldsymbol{\sigma}_{[m]}}\left[\sup_{\theta\in\Theta}\frac{1}{m}\sum_{i=1}^{m}\sigma_i\ell^{\theta}_{\text{DSM}}(\boldsymbol{x}_i)\right]$ is the empirical Rademacher complexity with respect to the parameter family Θ over $\boldsymbol{x}_{[m]}$. Substituting (33) and (34) into (32), with probability $\geq 1 - \delta'$ we have for *any* $\theta \in \Theta$

$$\begin{aligned}J_{\text{opt}}(\theta) \geq\ &\hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) - c_0 K\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})} - c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} - c_1 K W_2(\bar{q}^{\tilde{w}}_{\text{target}}, \mathcal{N}) -\\ &c_0 K\sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - c_0 K\sqrt[4]{\frac{18B\Delta\log(4/\delta')}{m}} - \sqrt{\frac{BF\log(2/\delta')}{m}}.\end{aligned} \tag{35}$$

To remove the *conditioning* on $\tilde{w}$, let $\tilde{\mathcal{W}}_\epsilon$ be an ϵ-*cover* [47] of $\tilde{\mathcal{W}}$ under the L_∞ norm. By (35), for any *given* $\tilde{v} \in \tilde{\mathcal{W}}_\epsilon$, with probability $\geq 1 - \delta'$ we have for *any* $\theta \in \Theta$

$$\begin{aligned}J_{\text{opt}}(\theta) \geq\ &\hat{J}_{\boldsymbol{x}[m]}(\tilde{v}) - c_0 K\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{v})} - c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{v})} - c_1 K W_2(\bar{q}^{\tilde{v}}_{\text{target}}, \mathcal{N}) -\\ &c_0 K\sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - c_0 K\sqrt[4]{\frac{18B\Delta\log(4/\delta')}{m}} - \sqrt{\frac{BF\log(2/\delta')}{m}}.\end{aligned}$$

By the *union* bound, with probability $\geq 1 - |\tilde{\mathcal{W}}_\epsilon|\delta'$ we have for *any* $\theta \in \Theta$ and *any* $\tilde{v} \in \tilde{\mathcal{W}}_\epsilon$

$$\begin{aligned}J_{\text{opt}}(\theta) \geq\ &\hat{J}_{\boldsymbol{x}[m]}(\tilde{v}) - c_0 K\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{v})} - c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{v})} - c_1 K W_2(\bar{q}^{\tilde{v}}_{\text{target}}, \mathcal{N}) -\\ &c_0 K\sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - c_0 K\sqrt[4]{\frac{18B\Delta\log(4/\delta')}{m}} - \sqrt{\frac{BF\log(2/\delta')}{m}}.\end{aligned}$$

Choosing $\delta' = \delta/|\tilde{\mathcal{W}}_\epsilon|$, with probability $\geq 1-\delta$ we have for *any* $\theta \in \Theta$ and *any* $\tilde{v} \in \tilde{\mathcal{W}}_\epsilon$

$$J_{\text{opt}}(\theta) \geq \hat{J}_{\boldsymbol{x}[m]}(\tilde{v}) - c_0 K \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta,\tilde{v})} - c_0 K\sqrt{2\Delta}\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{v})} - c_1 K W_2(\bar{q}^{\tilde{v}}_{\text{target}}, \mathcal{N}) - \\ c_0 K \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - c_0 K \sqrt[4]{\frac{18B\Delta \log(4|\tilde{\mathcal{W}}_\epsilon|/\delta)}{m}} - \sqrt{\frac{BF\log(2|\tilde{\mathcal{W}}_\epsilon|/\delta)}{m}}. \tag{36}$$

By the definition of ϵ-cover, for any $\tilde{w} \in \tilde{\mathcal{W}}$, there exists an $\tilde{v} \in \tilde{\mathcal{W}}_\epsilon$ such that $|\tilde{w}(f(\boldsymbol{x})) - \tilde{v}(f(\boldsymbol{x}))| \leq \epsilon$ for any $\boldsymbol{x} \in \mathcal{X}$. Note that this immediately implies that:

$$\begin{aligned}\hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) - \hat{J}_{\boldsymbol{x}[m]}(\tilde{v}) &= \frac{1}{m}\sum_{i=1}^m (\tilde{w}(f(\boldsymbol{x}_i)) - \tilde{v}(f(\boldsymbol{x}_i)))\, f(\boldsymbol{x}_i) \\ &\leq \frac{1}{m}\sum_{i=1}^m |\tilde{w}(f(\boldsymbol{x}_i)) - \tilde{v}(f(\boldsymbol{x}_i))|\, |f(\boldsymbol{x}_i)| \leq F\epsilon, \end{aligned} \tag{37}$$

where the last inequality follows from the assumption that $|f(\boldsymbol{x})| \leq F$ for any $\boldsymbol{x} \in \mathcal{X}$;

$$\begin{aligned}\sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta,\tilde{v})} - \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta,\tilde{w})} &= \sqrt{\frac{1}{m}\sum_{i=1}^m \tilde{v}(f(\boldsymbol{x}_i))\ell^\theta_{\text{DSM}}(\boldsymbol{x}_i)} - \sqrt{\frac{1}{m}\sum_{i=1}^m \tilde{w}(f(\boldsymbol{x}_i))\ell^\theta_{\text{DSM}}(\boldsymbol{x}_i)} \\ &\leq \sqrt{\frac{1}{m}\sum_{i=1}^m |\tilde{v}(f(\boldsymbol{x}_i)) - \tilde{w}(f(\boldsymbol{x}_i))|\, \ell^\theta_{\text{DSM}}(\boldsymbol{x}_i)} \leq \sqrt{\Delta\epsilon}, \end{aligned} \tag{38}$$

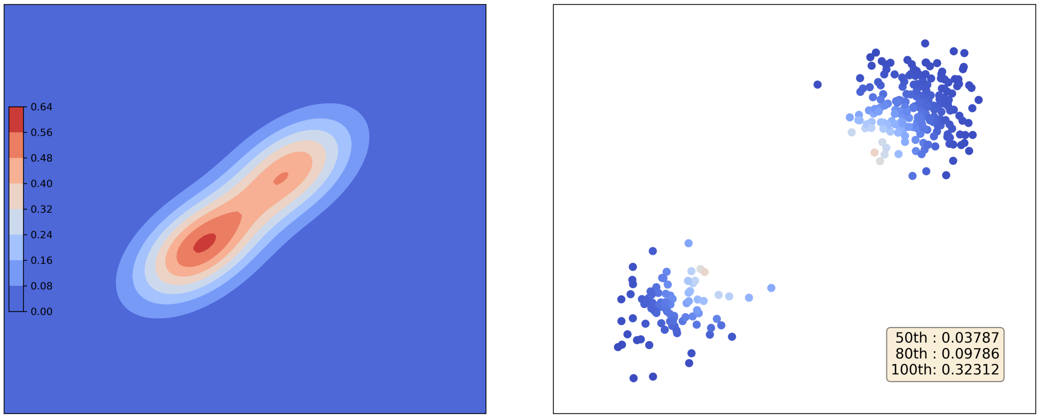
where the last inequality follows from the assumption that $0 \leq \ell^\theta_{\text{DSM}}(\boldsymbol{x}) \leq \Delta$ for any $\boldsymbol{x} \in \mathcal{X}$;

$$\begin{aligned}&\sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{v})} - \sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} \\ &= \sqrt[4]{\frac{1}{m}\sum_{i=1}^m (\tilde{v}(f(\boldsymbol{x}_i)) - 1)^2} - \sqrt[4]{\frac{1}{m}\sum_{i=1}^m (\tilde{w}(f(\boldsymbol{x}_i)) - 1)^2} \\ &= \sqrt[4]{\frac{1}{m}\sum_{i=1}^m (\tilde{v}(f(\boldsymbol{x}_i)) - \tilde{w}(f(\boldsymbol{x}_i)) + \tilde{w}(f(\boldsymbol{x}_i)) - 1)^2} - \sqrt[4]{\frac{1}{m}\sum_{i=1}^m (\tilde{w}(f(\boldsymbol{x}_i)) - 1)^2} \\ &\leq \sqrt[4]{\frac{1}{m}\sum_{i=1}^m (\tilde{v}(f(\boldsymbol{x}_i)) - \tilde{w}(f(\boldsymbol{x}_i)))^2} + \sqrt[4]{\frac{2}{m}\sum_{i=1}^m |\tilde{v}(f(\boldsymbol{x}_i)) - \tilde{w}(f(\boldsymbol{x}_i))|\, |\tilde{w}(f(\boldsymbol{x}_i)) - 1|} \\ &\leq \sqrt{\epsilon} + \sqrt[4]{2(B+1)\epsilon}, \end{aligned} \tag{39}$$

where the last inequality follows from the assumption that $0 \leq \tilde{w}(f(\boldsymbol{x})) \leq B$ for any $\boldsymbol{x} \in \mathcal{X}$; and

$$\begin{aligned}W_2(\bar{q}^{\tilde{v}}_{\text{target}}, \mathcal{N}) - W_2(\bar{q}^{\tilde{w}}_{\text{target}}, \mathcal{N}) &\leq W_2(\bar{q}^{\tilde{v}}_{\text{target}}, \bar{q}^{\tilde{w}}_{\text{target}}) \\ &\leq c_2 W_2(q^{\tilde{v}}_{\text{target}}, q^{\tilde{w}}_{\text{target}}) \\ &\leq c_2 d_2(\mathcal{X}) d_{\text{TV}}(q^{\tilde{v}}_{\text{target}}, q^{\tilde{w}}_{\text{target}}) \\ &= \frac{1}{2} c_2 d_2(\mathcal{X}) \int_{\mathcal{X}} |\tilde{v}(f(\boldsymbol{x})) - \tilde{w}(f(\boldsymbol{x}))| p_{\text{data}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ &\leq \frac{1}{2} c_2 d_2(\mathcal{X})\epsilon, \end{aligned} \tag{40}$$

where c_2 is the *Wasserstein contraction constant* [9] of the forward process (3), $d_2(\mathcal{X}) := \max_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}'\|_2$ is the *diameter* of $\mathcal{X}$ with respect to the ℓ_2 norm, and $d_{\text{TV}}(q^{\tilde{v}}_{\text{target}}, q^{\tilde{w}}_{\text{target}})$ denotes the *total variation distance* between $q^{\tilde{v}}_{\text{target}}$ and $q^{\tilde{w}}_{\text{target}}$. Here, the first inequality follows from the fact that the 2-Wasserstein distance is a metric and hence follows the triangle inequality, the second inequality follows from the *Wasserstein contraction* property [9] of the forward process,



(a) Objective function (b) Offline examples

Figure 2: A toy example: Initial setting.

and the third inequality follows from the *total-variation* bound [20] on the 2-Wasserstein distance. Substituting (37)–(40) into (36), with probability $\geq 1-\delta$ we have for *any* $\theta \in \Theta$ and *any* $\tilde{w} \in \tilde{\mathcal{W}}$

$$\begin{aligned} J_{\text{opt}}(\theta) \quad \geq \quad & \hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) - c_0 K \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})} - c_0 K \sqrt{2\Delta} \sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} - c_1 K W_2(\bar{q}^{\tilde{w}}_{\text{target}}, \mathcal{N}) - \\ & c_0 K \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - c_0 K \sqrt[4]{\frac{18 B \Delta \log(4|\tilde{\mathcal{W}}_\epsilon|/\delta)}{m}} - \sqrt{\frac{BF \log(2|\tilde{\mathcal{W}}_\epsilon|/\delta)}{m}} - \\ & \left(F + \frac{1}{2} c_2 d_2(\mathcal{X})\right)\epsilon - \left(\sqrt{\Delta} + 1\right)\sqrt{\epsilon} - \sqrt[4]{2(B+1)\epsilon}. \end{aligned} \tag{41}$$

By assumption, any normalized weight function from $\tilde{\mathcal{W}}$ is L-Lipschitz and bounded by B. Therefore, the covering number $|\tilde{\mathcal{W}}_\epsilon|$ is of the order $O(\exp(1/\epsilon))$. Let $\epsilon = m^{-\gamma}$ for some $\gamma \in (0,1)$, and we have from (41)

$$\begin{aligned} J_{\text{opt}}(\theta) \quad \geq \quad & \hat{J}_{\boldsymbol{x}[m]}(\tilde{w}) - c_0 K \sqrt{\hat{\mathcal{L}}_{\boldsymbol{x}_{[m]}}(\theta, \tilde{w})} - c_0 K \sqrt{2\Delta} \sqrt[4]{\hat{V}_{\boldsymbol{x}_{[m]}}(\tilde{w})} - c_1 K W_2(\bar{q}^{\tilde{w}}_{\text{target}}, \mathcal{N}) - \\ & c_0 K \sqrt{2\hat{\mathfrak{R}}_{\boldsymbol{x}_{[m]}}(\Theta)} - O(m^{-(1-\gamma)/4}) - O(m^{-\gamma/4}). \end{aligned} \tag{42}$$

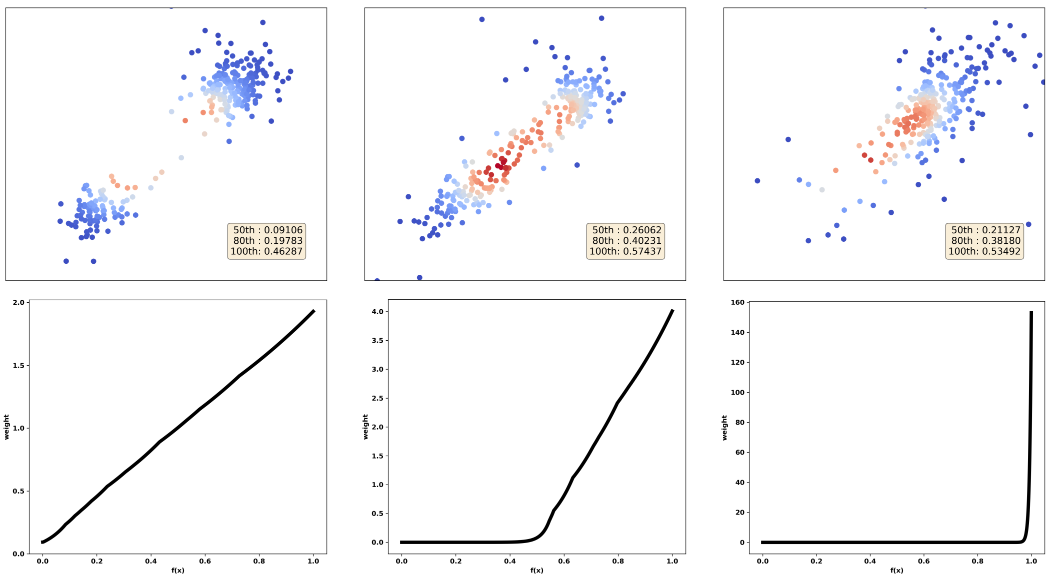
Choosing $\gamma = 1/2$ in (42) completes the proof of (1) and hence Theorem A.1.

C Additional Experiments and Details

C.1 A toy example

We first experimentally validate the proposed learning algorithm using a toy example in $\mathbb{R}^2$. In this example, the unknown objective function f is a mixture of two Gaussian density functions: $f(\boldsymbol{x}) = 2\sqrt{3}\pi\,[0.45 \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + 0.55 \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma})]$, where $\boldsymbol{\mu}_1 = [1.5, 1.5]^t$, $\boldsymbol{\mu}_2 = [-1.5, -1.5]^t$, $\boldsymbol{\Sigma} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$, and the unknown data-generating distribution p_{data} is a mixture of two Gaussian distributions: $p_{\text{data}}(\boldsymbol{x}) = 0.3 \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_3, \boldsymbol{I}) + 0.7 \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_4, \boldsymbol{I})$, where $\boldsymbol{\mu}_3 = [-4, -4]^t$, $\boldsymbol{\mu}_4 = [4, 4]^t$, and $\boldsymbol{I}$ is the 2×2 identity matrix. The weight function w_ϕ and the score-function model $\boldsymbol{s}_t^\theta$ are jointly learned by maximizing the proposed objective (2).

The filled contour plot of the objective function f is shown in Figure 2a, with warmer colors representing higher objective values. Figure 2b shows 300 samples drawn from the data-generating distribution p_{data}, with the color of each sample rendered according to its ground-truth objective value. These 300 samples and their corresponding objective values are the offline examples from which the weight function w_ϕ and the score-function model $\boldsymbol{s}_\theta$ are trained. Figure 3 shows the optimized



(a) $\alpha = 1$ (b) $\alpha = 0.3$ (c) $\alpha = 0$

Figure 3: A toy example: Top: Optimized samples; bottom: Learned weight function.

samples and the learned weight function w_ϕ for several different values of the hyper-parameter α while fixing the hyper-parameter $\lambda = 0.1$.

The legitimacy of the proposed approach is demonstrated by the following observations. i) Even though the weight function is *not* constrained to be monotonic *a priori*, as shown in the bottom row of Figure 3, the learned weight functions are *monotone increasing* and hence put higher weights to samples with higher objective values. ii) When $\alpha = 1$, the learned weight function is relatively “flat" across its input domain. As a result, the learned generative model is very close to the data-generating distribution, and the optimized samples are very “similar" to the initial samples. As we decrease the value of α from 1 to 0.3, the learned weight function becomes much more skewed towards the higher input values. As a result, some of the optimized samples have been nudged along the direction of the gradient of the objective function and hence have much higher objective values than the initial samples. When we further decrease the value of α to 0, the learned weight function becomes extremely skewed. In this case, the hypothetical target distribution is *not* learnable. As a result, instead of the gradient direction, the optimized samples have been nudged along *all* directions. Therefore, the hyper-parameter α can effectively control the *utility–learnability tradeoff* for selecting a weight function w_ϕ. iii) Compared to the data-generating distribution, with an appropriate choice of the hyper-parameters α and λ, the learned generative model is substantially more *capable* of generating samples with higher objective values, as demonstrated by the differences of the 50th, 80th, and 100th percentiles between the samples drawn from these two distributions. More results on this toy example can be found in Appendix C.4.1.

C.2 Benchmark datasets

We conducted experiments on five standard offline optimization tasks:

- **Superconductor**, which aims to design a superconductor with 86 components to maximize the critical temperature;
- **TF (Transcription Factor) Bind 8**, which aims to find a DNA sequence of 8 base pairs to maximize its binding affinity to a particular transcription factor;
- **GFP (Green Fluorescent Protein)**, which aims to find a protein sequence of 238 amino acids to maximize the fluorescence;
- **UTR (Untranslated Region)**, which aims to find a human 5’ UTR DNA sequence of 50 base pairs to maximize the expression level of its corresponding gene;

Table 2: The benchmark datasets

	Supercond.	TFBind8	GFP	UTR	Fluores.
Type	continuous	discrete	discrete	discrete	discrete
Dimension	86	8	237	50	13
Category	N/A	4	20	4	2
# Train/Total	17014/21263	32898/65792	5000/56086	140k/280k	4096/8192
Min/Max	0.0/185.0	0.0/1.0	1.283/4.123	0.0/12.0	0.155/1.692
$\mathcal{D}_{\text{best}}$	74.0/0.4	0.439/0.439	3.525/0.789	7.123/0.594	0.900/0.485

- **Fluorescence**, which aims to identify a protein with high brightness. At each position, the selection of an amino acid is limited to those found in the sequences of the two parent fluorescent proteins. These parent proteins differ at precisely 13 positions in their sequences while being identical at all other positions.

For all previous tasks except for the Fluorescence, we utilized the Design-Bench package [46] to generate the training data, pre-process the data (including the conversion of categorical features to numerical values), and evaluate new designs. For the Fluorescence task, we collected raw data from Fannjiang et al. [18]. The objective value in this case is represented by the combined brightness. From a total of $2^{13} = 8192$ samples, we selected the worst 4096 examples as our training dataset. While the features in the Fluorescence dataset are binary, we simply treated them as continuous inputs to our algorithm.

The key attributes of the aforementioned benchmark datasets can be found in Table 2, which include:

- **Type**: The type of features represented in the dataset, which can be either continuous or discrete;
- **Dimension**: The feature dimension of the dataset;
- **Category**: The number of categories for each feature (only applicable to the discrete datasets);
- **# Train/Total**: The number of samples in the training and entire datasets. The entire dataset includes both the training dataset and additional data examples, which are used to help evaluate the new designs;
- **Min/Max**: The minimum and maximum objective values within the entire dataset;
- $\mathcal{D}_{\textbf{best}}$: The un-normalized and normalized maximum objective values within the training dataset.

C.3 Implementation details

Normalization. As we adopted DDPM as our generative model, we normalized each feature to the interval $[-1, 1]$. For the objective values, we mapped the original values in the training dataset to the range of $[0, 1]$. This step ensures consistency in the learning of the (un-normalized) weight function w_ϕ. For the GFP task, we employed a variational auto-encoder [31] to embed the high-dimensional features into a lower-dimensional space before normalizing them into the interval $[-1, 1]$.

Networks. In our implementation, we used neural networks to model both the (un-normalized) weight function w_ϕ and the score function $\boldsymbol{s}_t^\theta$. The weight function is a simple *scalar* function. In our implementation, we simply used a 4-layer multi-layer perceptron (MLP) with ReLU activation functions. In addition, we applied an *exponential* function to the output of the MLP to enforce the *non-negativity* of the weight function. The architecture for the score function model consists of a time-embedding layer and five blocks of "Dense-BatchNorm-ELU". Before each block, we injected time-embedding information by concatenating it with the input to the block.

Training. The noise scheduler for the DDPM was chosen as $\beta(t) = \beta_{\text{min}} + (\beta_{\text{max}} - \beta_{\text{min}})t$ for $t \in [0, 1]$, where $\beta_{\text{min}} = 0.1$ and $\beta_{\text{max}} = 20$. The detailed training procedure is described in Algorithm 1. The training scheme involves first identifying a suitable *initialization* of ϕ and θ and then followed by an *alternating* maximization over ϕ and θ. More specifically, to obtain a suitable initialization of ϕ and θ, we first note that the model θ *only* shows up in the second term of our learning objective (2). Maximizing the other two terms over ϕ gives us an initial estimate ϕ_0 (see Line 3 of Algorithm 1). In our implementation, this maximization was performed via full-batch gradient descent (GD), for which we used the Adam optimizer [30] with a constant leaning rate 10^{-3}. Once an initial estimate ϕ_0 has been obtained, we can obtain an initial estimate θ_0 by minimizing the second term over θ while setting $\phi = \phi_0$ (see Line 4 of Algorithm 1). To minimize the weighted denoising score matching loss, we considered a time range of $t \in [10^{-3}, 1]$ and used the Adam optimizer with a variable learning rate via stochastic gradient descent (SGD). The learning rate was gradually decreased from 10^{-3} to 10^{-4} during training. The alternating maximization of the learning objective (2) over the parameters ϕ and θ is described in Line 5–8 of Algorithm 1. Again the Adam optimizer was used, and the learning rates were set as $\eta_1 = \eta_2 = 10^{-4}$.

Sampling/Optimization. The sampling/optimization procedure is described in Algorithm 2. This procedure is identical to the probability-flow sampler in Song et al. [44].

C.4 Additional experimental results

C.4.1 Toy example

Additional choices of the hyper-parameter α. Previously in Section C.1, we described a toy example in $\mathbb{R}^2$ and used it to validate our proposed approach. In particular, in Figure 3 we reported the optimized samples and the learned weight function w_{ϕ^*} for several choices of the hyper-parameter α. Here in Figures 4 and 5 we report the optimized samples and the learned weight function w_{ϕ^*} for some additional choice of the hyper-parameter α. Note that when $\alpha = \infty$, the learned weight function w_{ϕ^*} is completely flat across its domain, and thus the hypothetical target distribution q_{target} is identical to the data-generating distribution p_{data}. It should become very clear from these reported results that the hyper-parameter α can effectively control the utility-learnability tradeoff for selecting a weight function.

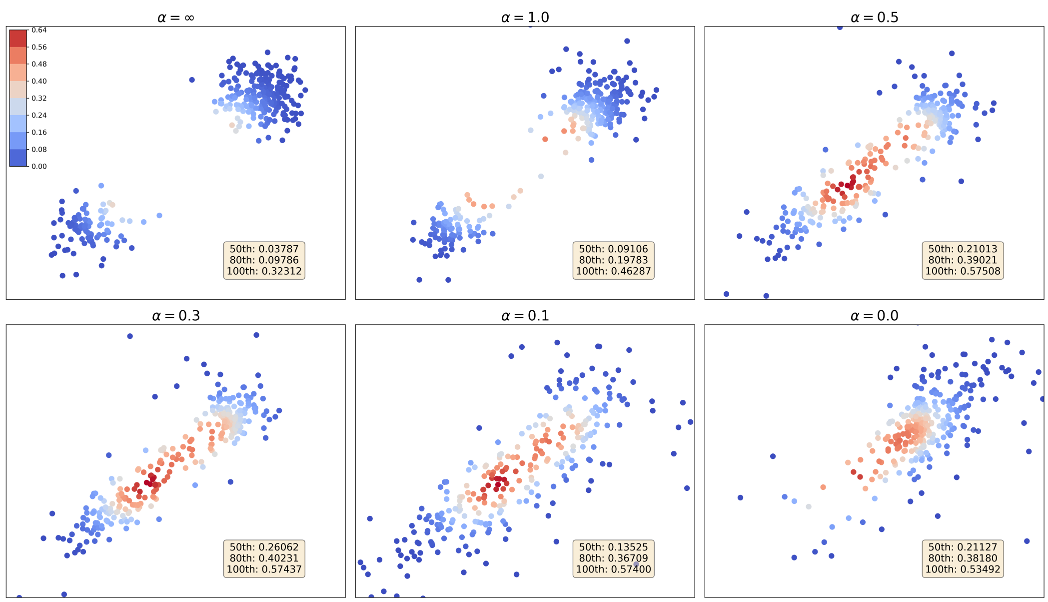


Figure 4: Optimized samples (with learnable weight function) for different choices of the hyper-parameter α.

Pre-selected weight function. Instead of considering a *learnable* weight function w_ϕ, we may also consider using a *pre-selected* weight function to train the generative model p_θ. Motivated by the learned weight functions w_{ϕ^*} from Figure 5, here we consider the simple *exponential* function $w_\psi(y) = \exp(\psi y)$, where ψ is a hyper-parameter. Note that when $\psi = 0$, the weight function w_ψ is completely flat across its domain, and as we increase the value of ψ, w_ψ becomes increasingly skewed towards the higher values in its domain. The optimized samples and the corresponding pre-selected weight functions are reported in Figures 6 and 7. Note here that we have purposely

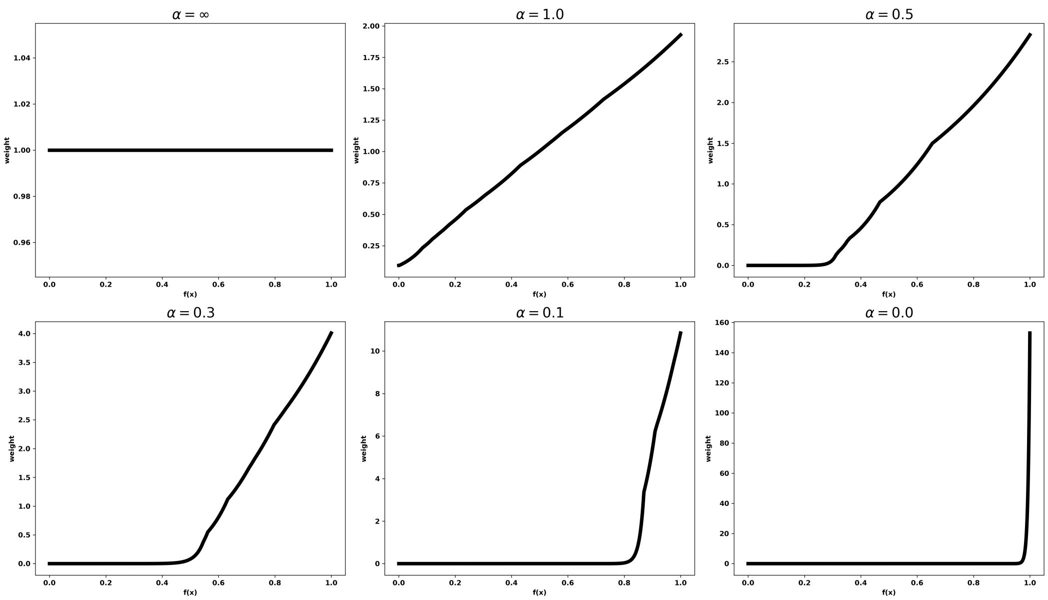


Figure 5: Learned Weight function w_{ϕ^*} for different choices of the hyper-parameter α.

chosen the values of the hyper-parameter ψ such that the pre-selected weight functions w_ψ in Figure 7 *mimic* the learned weight function w_{ϕ^*} in Figure 5. As a result, the optimized samples from Figures 6 have similar statistical profiles as those from Figures 4. Next, we shall use the benchmark datasets to illustrate that a learnable weight function can significantly outperform a pre-selected weight function in terms of generating samples with a consistent and superior statistical profile.

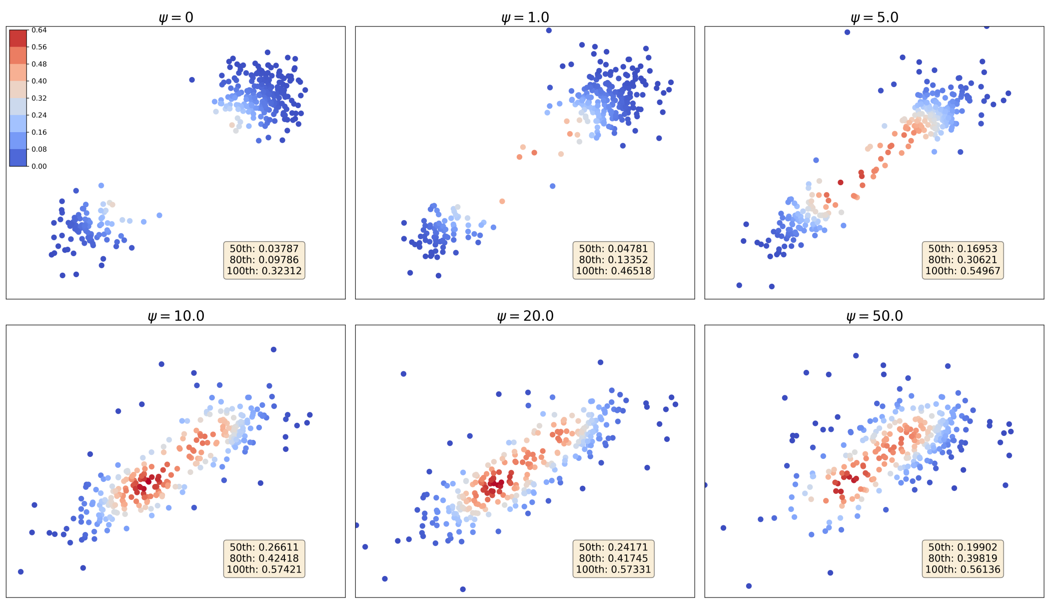


Figure 6: Optimized samples with pre-selected weight function for different choices of the hyper-parameter ψ.

C.4.2 Benchmark datasets

Here we report additional results on the benchmark datasets using both the learnable weight function w_ϕ and the pre-selected exponential weight function w_ψ. In our experiments, we fixed the value of the

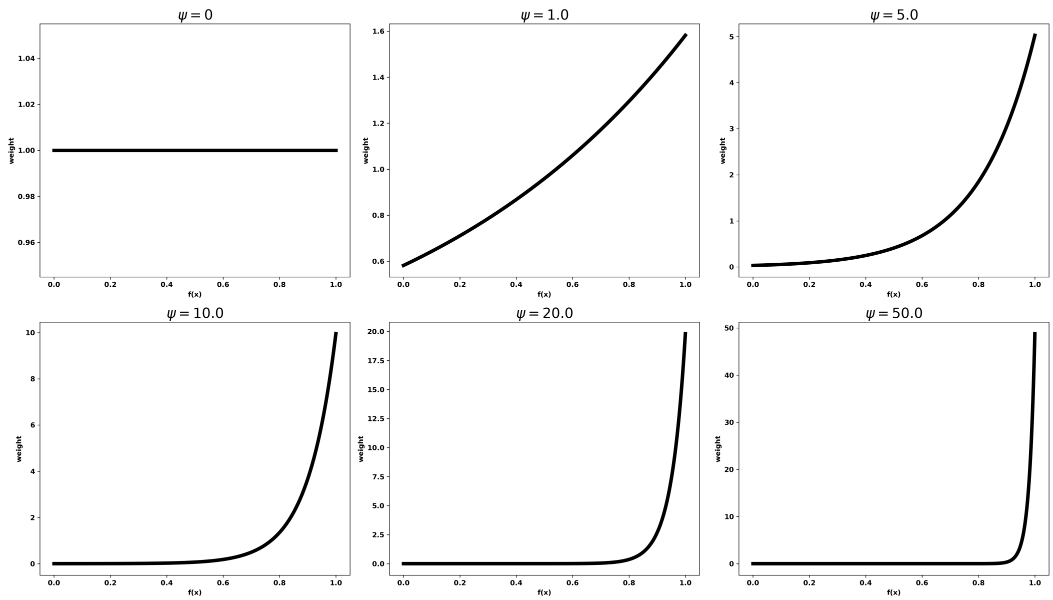


Figure 7: Pre-selected weight function w_ψ for different choices of the hyper-parameter ψ.

hyper-parameter $\lambda = 0.1$ and considered several different values for the hyper-parameter α (learnable weight function) and ψ (pre-selected weight function). The mean and standard deviation of the *best* generated samples for each benchmark dataset are reported in Table 3. The average improvements for different choices of the hyper-parameter α (learnable weight function) and ψ (pre-selected weight function) are also reported in Table 3. It is clear that the use of a learnable weight function with $\alpha = 0.25$ significantly outperform any pre-selected weight function considered here in terms of the average improvement. The learned weight functions w_{ϕ^*} that correspond to $\alpha = 0.25$ for each of the benchmark datasets are reported in Figure 8.

On the top row of Figure 8, the dashed lines represent pre-selected weight functions, while the solid line represents the learned weight function for each of the benchmark datasets. It is evident that the learned weight functions differ significantly from the pre-selected exponential weight functions. The label histograms of for each of the benchmark datasets are shown on the bottom row of Figure 8. Noticeably, labels of Superconductor and Fluorescence datasets are heavily skewed towards small objective values. Coincidentally, their learned weight functions tend to flatten when the objective values are large (roughly $0.8 - 1.0$). Given that the majority of data examples in these datasets have small objective values, assigning higher weights to large objective values would result in a smaller "effective sample size" in weighted learning. Therefore, it makes sense that these two datasets adopted a less steep slope in the learned weight function to maintain a reasonable effective sample size. This observation underscores that our learning objective (2) can adaptively choose an appropriate weight function to balance the utility-learnability tradeoff. Thus, to fully harness the potential of the proposed generative approach, it is crucial to make the weight function learnable as well.

In all the experiments involving a learnable weight function, we consistently set λ to 0.1. We also conducted additional experiments to investigate the impact of the hyper-parameter λ on the results. The findings, as depicted in Figure 9, reveal that λ proves to be a relatively insensitive hyper-parameter within our method. Across all datasets, variations in λ ranging from 0.01 to 1.0 only result in very minor differences in terms of the optimization performance.

D Further Discussions

To further elucidate the main contributions of this paper, we put the proposed *PAC-generative* approach to offline optimization in the context of several related works.

Modeling target distribution *vs*. modeling objective function. As mentioned previously in Section 1, the “standard" approach to offline optimization is to first learn a surrogate of the unknown

Table 3: Mean and standard deviation of the best generated samples for different choices of the hyper-parameter α (learnable weight function) and ψ (pre-selected weight function).

	Supercond.	TFBind8	GFP	UTR	Fluores.	Ave. Improv.
$\mathcal{D}_{\text{best}}$	0.399	0.439	0.789	0.593	0.485	
$\psi = 0.5$	0.493±0.033	0.892±0.054	0.865±0.000	0.669±0.009	0.881±0.036	0.462
$\psi = 1.0$	0.478±0.027	0.914±0.044	0.865±0.000	0.678±0.015	0.882±0.022	0.467
$\psi = 5.0$	0.496±0.026	0.917±0.022	0.865±0.000	0.689±0.009	0.859±0.052	0.472
$\psi = 20.0$	0.423±0.044	0.903±0.050	0.865±0.000	0.693±0.006	0.803±0.057	0.407
$\alpha = 0.15$	0.425±0.073	0.925±0.043	0.864±0.000	0.693±0.010	0.696±0.060	0.374
$\alpha = 0.2$	0.468±0.029	0.913±0.051	0.864±0.000	0.698±0.016	0.739±0.054	0.410
$\alpha = 0.25$	0.537±0.045	0.941±0.034	0.865±0.000	0.693±0.013	0.809±0.078	0.485
$\alpha = 0.3$	0.502±0.054	0.924±0.029	0.865±0.000	0.697±0.008	0.798±0.045	0.456

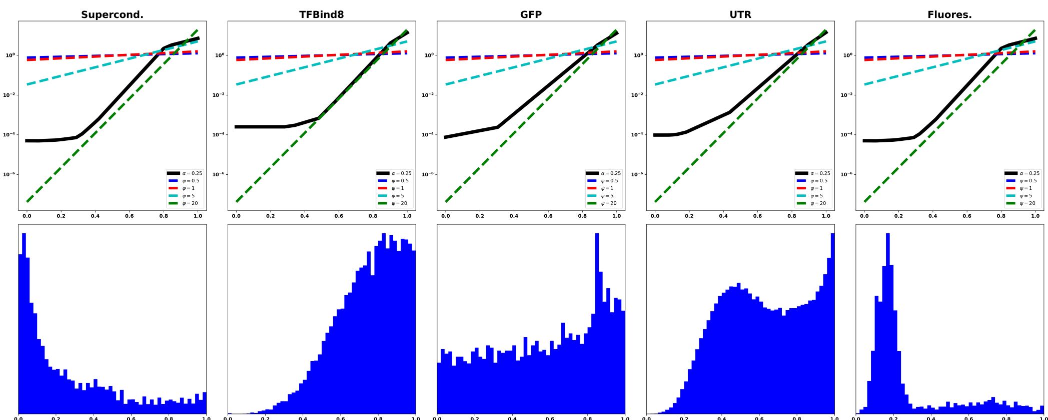


Figure 8: Top: Learned weight functions w_{ϕ^*} (solid, black line) for the benchmark datasets; bottom: histograms of offline training labels.

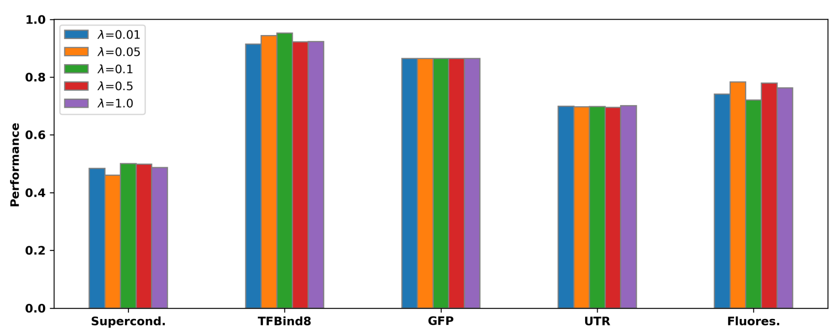


Figure 9: Investigation of λ when $\alpha = 0.2$.

objective function and then apply existing optimization algorithms. The main challenge for modeling the objective function is the so-called *distributional shift*. That is, when the optimization algorithm explores regions *away* from the offline observations, the learned surrogate tends to become less accurate. It is thus crucial to understand how far the optimization algorithm can explore away from the offline observations and how to maintain the accuracy of the learned surrogate throughout the exploration process. Notable efforts in the literature include Qi et al. [41] and Trabucco et al. [45], which considered regularized surrogate models in favor of invariance and conservatism; Fannjiang and Listgarten [17] and Chen et al. [13], which considered surrogate models learned via importance sampling and contrastive learning; and Fannjiang et al. [18], which used conformal prediction to quantify the uncertainty of the learned surrogate.

Despite these efforts, however, it remains unclear how to align the objective of learning a surrogate of the unknown objective function with the objective of optimization. This is evidenced by the very recent work Beckham and Pal [7], which discussed how one may interpret the conservative approach proposed in Trabucco et al. [45], and Beckham et al. [8], which suggested that an alternative evaluation metric is potentially better than simply choosing the best candidates using the learned surrogate. In contrast, the PAC-generative approach proposed in this paper is based on modeling a target distribution (as opposed to the objective function). As we have shown, under this generative view, it is possible to tune the objective of the learner according to a natural optimization objective.

Weighted learning *vs*. conditional/guided generation. Recent years have seen remarkable success in conditional/guided image generation [15, 23]. Conditional/guided generation can be easily adapted to offline optimization. Specifically, one can simultaneously learn a *standard* score-based generative model and a surrogate of the objective function and then use the *gradient* of the learned surrogate to guide the generation of the optimized samples [40]. Alternatively, one may also model the target distribution q_{target} as the *conditional* distribution p_{data} given $f(\mathbf{x}) \geq y_0$ for some *threshold* y_0 and train a generative model that approximates this conditional distribution [11, 22]. However, learning the conditional distribution may also require a surrogate of the objective function. In contrast, in our approach we model the target distribution q_{target} using a weight function. As discussed in Section 2.1, in our *weighted-learning* model, the score of q_{target} is *intrinsically* aligned with the gradient of the objective function. Hence we directly train a generative model from the offline data examples to learn the score of q_{target}, and there is *no* need to learn a surrogate of the objective function separately.

Offline optimization *vs*. offline reinforcement learning (RL). While the focus of this paper is offline optimization, recent years have also seen a substantial amount of interest in *offline RL* [36, 49, 27, 50]. Even though these two problems face some similar challenges, in our evaluation offline RL is the considerably more challenging setting. It is thus of interest to see whether the proposed PAC-generative approach can lead to any success in offline RL as well.

D3S3@NeurIPS Paper Checklist (Optional)

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We derived a PAC-style lower bound on optimization objective and learned generative models based on the surrogate loss function.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We didn't have a separate 'Limitations' section, while we had a thorough discussion in Appendix D to compare with other methods.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Detailed proofs are presented in Appendix B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We elaborated all details in the main paper and Appendix C. We can also provide source code if needed.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code includes a README file with instructions to run the code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please check Appendix C.3 and source code if necessary.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We ran our experiments for 8 trials and considered the statistical numbers as final criteria.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Because our experiments do not require heavy computing resources. A normal PC with GPU(s) can run this code efficiently.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We anonymized our code in order to obey the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This work only focused on methodology.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly mentioned or cited assets owned by others.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our code includes proper documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.