Tuning-Free Alignment of Diffusion Models with Direct Noise Optimization

Zhiwei Tang [1 *] **Jiangweizhi Peng** [2] **Jiasheng Tang** [3 4] **Mingyi Hong** [2] **Fan Wang** [3] **Tsung-Hui Chang** [1 5]

Abstract

In this work, we focus on the alignment problem of diffusion models with a continuous reward function, which represents specific objectives for downstream tasks, such as improving human preference. The central goal of the alignment problem is to adjust the distribution learned by diffusion models such that the generated samples maximize the target reward function. We propose a novel alignment approach, named Direct Noise Optimization (DNO), that optimizes the injected noise during the sampling process of diffusion models. By design, DNO is *tuning-free* and *prompt-agnostic*, as the alignment occurs in an online fashion during generation. We rigorously study the theoretical properties of DNO and also empirically identify that naive implementation of DNO occasionally suffers from the *out-of-distribution reward hacking* problem, where optimized samples have high rewards but are no longer in the support of the pretrained distribution. To remedy this issue, we leverage classical high-dimensional statistics theory and propose to augment the DNO loss with certain probability regularization. We conduct a novel experiment to verify that DNO is an effective tuning-free approach for aligning diffusion models, and our proposed regularization can indeed prevent the out-of-distribution reward hacking problem.

1. Introduction

Diffusion models have become the state-of-the-art model for generating high-quality images, with exceptional resolution, fidelity, and diversity (Ho et al., 2020; Dhariwal and Nichol, 2021; Song et al., 2020b). These models are also remarkably easy to train and can be effectively extended to conditional generation (Ho and Salimans, 2022). Diffusion models work by learning to reverse the process of diffusing the data distribution $p(x)$ into noise, which can be described by a stochastic differential equation (SDE) (Song et al., 2020b; Karras et al., 2022): $dx_t = f(t)x_t dt + g(t)dw_t$, where dw_t is the standard Wiener process, and $f(t)$ and $g(t)$ are the drift and diffusion coefficients, respectively. The reverse process relies on the score function $\epsilon(x_t, t) \overset{\text{def.}}{=} \nabla_x \log p_t(x)$ where p_t denotes the p.d.f of noisy data x_t, and its closed-form can be expressed either as an ODE or as an SDE: (Song et al., 2020b):

$$\text{ODE: } dx_t = \left(f(t)x_t - \frac{1}{2} g^2(t)\epsilon(x_t, t) \right) dt, \tag{1}$$

$$\text{SDE: } dx_t = \left(f(t)x_t - g^2(t)\epsilon(x_t, t) \right) dt + g(t)dw_t. \tag{2}$$

With the capability to evaluate $\epsilon(x_t, t)$, it is possible to generate samples from noise by numerically solving either the ODE (1) or the SDE (2). The training process, therefore, involves learning a parameterized surrogate $\epsilon_\theta(x_t, t)$ to approximate $\epsilon(x_t, t)$, following a denoising score matching framework in (Song et al., 2020b; Karras et al., 2022).

Despite the effectiveness of diffusion models in modeling continuous distributions, a prevalent issue of diffusion model is that they are trained on large-scale, unlabeled, and mixed-quality data to capture the data distribution. However, when deploying these generative models for specific tasks, it is no longer suitable to sample from the original learned distribution directly, because this distribution has *not been aligned* with the task-specific objective. For instance, in image generation, users aim to generate images that are aesthetically pleasing rather than random ones. Hence, it is imperative to tailor the generative models to align with the task-specific objectives before utilization. The topic of aligning generative models has gained significant attention following the success of aligning LLMs with Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022; Liu et al., 2023a; OpenAI, 2022; Bai et al., 2022). Recently, the alignment problem has also drawn considerable interest in the context of diffusion models, as evidenced by a series of studies such as (Deng et al., 2024; Uehara

[*]This work was done when Zhiwei Tang was intern at DAMO Academy. [1]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China [2]University of Minnesota, USA [3]DAMO Academy, Alibaba Group [4]Hupan Lab, Zhejiang Province [5]Shenzhen Research Institute of Big Data, Shenzhen, China. Correspondence to: Zhiwei Tang <zhiweitang1@link.cuhk.edu.cn>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

et al., 2024; Yuan et al., 2024; Song et al., 2023a; Dong et al., 2023; Sun et al., 2023; Zhang et al., 2023; Prabhudesai et al., 2023; Black et al., 2023; Fan et al., 2023).

Alignment Problem for Diffusion Models. Given a diffusion model characterized by parameters θ and its associated distribution $p_\theta(x)$, as well as a reward function $r(x)$ that can assign real-valued scores to generated samples, the central goal of the alignment problem is to adjust the distribution $p_\theta(x)$ such that the generated samples maximize the reward from $r(x)$. Such reward models $r(x)$ can be constructed by training a neural network on human feedback data (Kirstain et al., 2023) or by utilizing custom-designed metrics. In this work, we consider the reward models to be continuous and possibly non-differentiable. In Appendix A, we provide a comprehensive review of some well-established methods for aligning diffusion models.

Drawbacks of Tuning-based Alignment Methods. Major existing alignment methods such as (Black et al., 2023; Wallace et al., 2023a; Clark et al., 2023) are tuning-based, meaning they necessitate adjustments to the network models θ. There are two main disadvantages associated with tuning-based methods. First, they require fine-tuning for each new reward model, a process that can consume considerable resources, especially when faced with extensive choices for reward models. Second, the fine-tuning process typically relies on a limited set of input prompts, which challenges the model to generalize to new and unseen prompts.

1.1. Our Contributions

In this work, we concentrate on tuning-free alignment of diffusion models, as we believe that flexibility and low computing requirements matter more than generation time for a broad range of real-world applications. We focus on an under-explored route for achieving tuning-free alignment of diffusion models—**Direct Noise Optimization (DNO)**. Specifically, we make the following contributions:

- We conduct a self-contained and comprehensive study for DNO, and demonstrate that noise optimization can be theoretically interpreted as sampling from an improved distribution.
- We identify out-of-distribution reward-hacking as a critical issue in DNO. To address this problem, we introduce a probability-regularized noise optimization method designed to ensure the generated samples remain within the support of pretrained distribution.

2. Direct Noise Optimization for Aligning Diffusion Models

Solving the ODE (1) or the SDE (2) typically involves discretizing the time steps into T steps. By starting with the initial noise $x_T \sim \mathcal{N}(0, I)$, the solution process for the ODE (1) can be viewed as a mapping that transforms the initial noise x_T into less noisy data through $x_{t-1} = \text{ODE\_solver}(x_t)$ for $t = T, ..., 1$, eventually resulting in the generated sample x_0. Similarly, solving the SDE (2) can be seen as a mapping that gradually converts both the initial noise x_T and the entire discretized Wiener process $\{w_t\}$ into the generated sample x_0, e.g., through $x_{t-1} = \text{SDE\_solver}(x_t) + w_t$.

Diffusion Sampling as a Noise-to-Sample Mapping. From this viewpoint, the diffusion sampling process can be conceptualized as an end-to-end mapping $M_\theta(z)$, which translates noise vectors z, sampled from the standard Gaussian distribution, into generated samples. Here, the noise vectors z serve as a unified abstraction for both the x_T in the ODE-based sampling process and the $(x_T, \{w_t\})$ in the SDE-based sampling process.

Given the noise-to-sample mapping M_θ described above, Direct Noise Optimization (DNO) can be mathematically formulated as follows:

$$\max_z r(M_\theta(z)). \tag{3}$$

By solving this optimization problem, we can obtain the optimized noise vectors, which are then used to generate refined samples. When the reward function $r(\cdot)$ is differentiable, gradient-based optimization methods can be applied to solve the problem efficiently. As one can see, this method is tuning-free, i.e., without the need to adjust the network parameter θ. We claim that optimizing the noise vectors for diffusion models indeed leads to an improved distribution in terms of the reward scores. To rigorously describe this improvement on distribution, we define an operator function g to represent a single gradient step, i.e., $g(z) \overset{\text{def}}{=} z + \ell \nabla_z r \circ M_\theta(z)$, where ℓ denotes the step size for gradient ascent. Additionally, we define the operator g_t, which denotes applying the gradient ascent step for t steps, i.e., $g_t(z) = g(g_{t-1}(z))$ with g_0 being the identity mapping. With these notations, we can now express the distribution after t gradient steps as $p_t(x)$, which is the distribution of $M_\theta(g_t(z))$ with $z \sim \mathcal{N}(0, I)$. In the following theorem, we demonstrate that there is a rigorous improvement after every single gradient step, i.e., the distribution $p_{t+1}(x)$ is provably better than $p_t(x)$ in terms of expected reward.

Theorem 2.1. *Assuming that $r \circ M_\theta$ is L-smooth, namely, $\|\nabla r \circ M_\theta(z) - \nabla r \circ M_\theta(z')\| \le L\|z - z'\|$ for any $z \neq z'$, it holds true that*

$$\mathbb{E}_{x\sim p_{t+1}(x)} r(x) \ge \mathbb{E}_{x\sim p_t(x)} r(x) + \left(\ell - \frac{\ell^2 L}{2}\right) \mathbb{E}_{z_0\sim N(0,I)} \left\|\nabla_z r \circ M_\theta(z)_{|z=g_t(z_0)}\right\|_2^2. \tag{4}$$

With this theorem, a natural question arises: When does the distribution stop improving? Namely, when does the second term in (4) become zero. To answer this question, we provide a detailed discussion on this point and the proof for Theorem 2.1 in Appendix E.

3. Out-Of-Distribution Reward-Hacking in Noise Optimization

It has been observed that when aligning generative models with reward functions, they can be subject to reward-hacking, i.e., the optimized samples yield high rewards but do not possess the desirable properties. Generally, there are two different types of reward-hacking. The first is because the reward model is undertrained, which results in optimized samples scoring high rewards but remaining barely distinguishable from the samples of the pretrained distribution (Miao et al., 2024; Chen et al., 2024a). The second type relates to the generative model used – the optimized samples no longer fall within the support of the pretrained distribution after optimization. We denote this second type of reward-hacking as **Out-Of-Distribution (OOD) Reward-Hacking**. In this work, we focus on this second type and reveal that OOD reward-hacking is a common issue in DNO.

Cause of OOD Samples. In this work, one of our key contributions is to identify one critical cause of OOD reward-hacking in noise optimization. That is, the optimized noise vectors stray towards the low-probability regions of the high-dimensional standard Gaussian distribution; in other words, there is an extremely low chance of such noise vectors being sampled from the Gaussian distribution. As diffusion models are originally trained with Gaussian noise, when the noise vectors originate from these low-probability areas—such as vectors comprised entirely of zeros—the neural network within the diffusion models may incur significant approximation errors for these particular inputs. This error, in turn, prompts the generation of out-of-distribution samples. In the subsequent section, we introduce an innovative method to measure the extent to which noise is part of the low-probability region by leveraging the classical concentration inequalities for high-dimensional Gaussian distributions.

3.1. Quantifying Low-Probability Region via Concentration Inequalities

In high-dimensional statistics, concentration inequalities are usually employed to describe these distinctive properties and delineate the low-probability regions of high-dimensional distributions. In the following lemma, we present two classical inequalities for the standard Gaussian distribution.

Lemma 3.1 ((Wainwright, 2019)). *Consider that $z_1, ..., z_m$ follow a k-dimensional standard Gaussian distribution. We have the following concentration inequalities for the mean and covariance:*

$$\Pr\left[\left\|\frac{1}{m}\sum_{i=1}^{m} z_i\right\| > M\right] < p_1(M), \tag{5}$$

$$\Pr\left[\left\|\frac{1}{m}\sum_{i=1}^{m} z_i z_i^\top - I_k\right\| > M\right] < p_2(M), \tag{6}$$

where $p_1(M) \overset{\text{def.}}{=} \max\left\{2e^{-\frac{mM^2}{2k}}, 1\right\}$ *and* $p_2(M) \overset{\text{def.}}{=} \max\left\{2e^{-\frac{m\left(\max\left\{\sqrt{1+M}-1-\sqrt{k/m}, 0\right\}\right)^2}{2}}, 1\right\}$

In practice, to determine if an n-dimensional vector z lies within a low-probability region, we can factorize n as $n = m \cdot k$, and divide z into m subvectors: $z = [z_1^1, ..., z_m^k]$, where $n = m \cdot k$ and $z_i = [z_i^1, ..., z_i^k] \sim \mathcal{N}(0, I_k)$). Then, we compute $M_1(z) = \left\|\frac{1}{m}\sum_{i=1}^{m} z_i\right\|$ and $M_2(z) = \left\|\frac{1}{m}\sum_{i=1}^{m} z_i z_i^\top - I_k\right\|$. Finally, we can determine that z lies in a low-probability region if both $p_1(M_1(z))$ and $p_2(M_2(z))$ are low.

Remark 3.2. As one can see, the computed probability is related to the factors m and k of n. According to (Wainwright, 2019), the two inequalities (5) and (6) are tight when m/k is large. On the other hand, $k = 1$ is not advisable, as it examines only the mean and variance of the noise vector, but not the covariance of different subvectors. In this work, we empirically found that $k = 2$ serves as a good default choice. In Appendix H, we provide a more detailed analysis for choosing an appropriate k.

An important thing to note is that the standard Gaussian distribution is invariant to permutation, i.e., for any permutation matrix Π, if z follows a standard Gaussian distribution, the permuted vector Πz will have the same probability behavior. With this insight, to increase the robustness of the probability measure p_1 and p_2, a natural idea is to examine the probability of many permuted vectors. Specifically, given q permutation matrices $\Pi_1, ..., \Pi_q$, we define the following indicator metric, $P(z) \overset{\text{def.}}{=} \min\{P_1(z), \}$, where $P_1(z) \overset{\text{def.}}{=} \min_{i\in\{1,...,q\}} p_1\left(M_1(\Pi_i z)\right)$ and $P_2(z) \overset{\text{def.}}{=} \min_{i\in\{1,...,q\}} p_2\left(M_2(\Pi_i z)\right)$

Interpretation of $P(z)$. If the probability $P(z)$ is low, it implies that there exists a permutation matrix Π_i such that the noise vector $\Pi_i z$ is in the low-probability region of the standard Gaussian distribution. Therefore, due to the permutation-invariant property, the noise vector z is also less likely to be sampled from the standard Gaussian distribution. In practice, we found that setting $q = 100$ and using randomly generated permutation matrices do well empirically. In Appendix F, we provide some visualized empirical evidence to show that $P(z)$ serves as a good indicator for determining if the generated samples are OOD.

3.2. Probability-Regularized Noise Optimization

With the insights discussed above, a natural idea for preventing OOD reward hacking is to regularize noise vectors to remain within the high-probability region of the Gaussian distribution. To achieve this, we propose the following **Probability-Regularized Noise Optimization (PRNO)** problem:

$$\max_z r(M_\theta(z)) + \gamma \mathbb{E}_\Pi \left[\log p_1(M_1(\Pi z)) + \log p_2(M_2(\Pi z)),\right. \tag{7}$$

where γ is the coefficient used to control the regularization effect. For the regularization term, we use the expectation of the log probabilities over the permutation matrices, rather than the minimum probability $P(z)$. This is because the expectation is smoother for optimization purposes.

4. Examining the Effectiveness of Probability Regularization

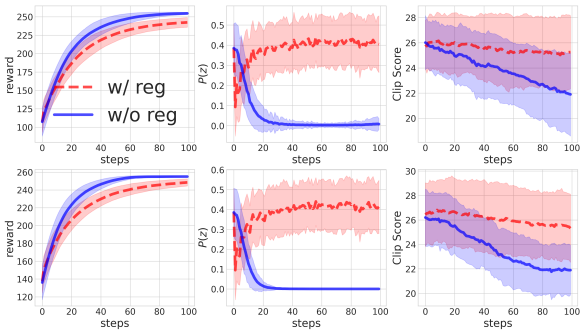


Figure 1. Upper row: Optimizing whiteness reward. Lower row: Optimizing blackness reward.

In this section, we design a novel experiment to demonstrate the effectiveness of probability regularization proposed in Section 3.2, utilizing Stable Diffusion v1.5 (Rombach et al., 2022) as the base model for DNO. For each figure, we perform the optimization using 1,000 different random seeds and report the average value along with the standard deviation (std) of the results. For comprehensive details regarding the implementation of our proposed methods, as well as information on hyperparameters, we refer readers to Appendices G and H. Additionally, we provide examples to visualize the optimization process in Appendix I.

(a) w/o reg (b) w/ reg (c) w/o reg (d) w/ reg

Figure 2. Examples of optimized images.

Setting. We consider two settings: the first involves optimizing the whiteness reward, i.e., the average value of all pixels—the larger this value is, the whiter the image becomes—with the prompt "black {animal}", where the token {animal} is randomly selected from a list of animals. The second setting involves optimizing the blackness reward, i.e., the negative of the whiteness reward, with the prompt "white {animal}". The primary insight behind designing such experiments is the clear contradiction between the prompt and the reward, making it easier to trigger the OOD reward-hacking phenomenon. Moreover, it is straightforward to verify whether the generated samples fall within the support of the prompt by simply examining the color of the generated animals. In these experiments, we will compare the noise optimization process with and without probability regularization to see whether probability regularization can prevent the OOD reward-hacking phenomenon.

Measuring the Degree of OOD. As observed in this experiment, the reward function contradicts the input prompt, leading to inconsistency between the generated samples and the prompt. Therefore, we utilize the CS (Radford et al., 2021), a commonly used metric for measuring the semantic similarity between images and text descriptions, to gauge the degree of OOD for the generated samples.

Results. In Figure 1, we first observe that adding the regularization term leads to a mildly slower optimization process. However, the generated samples are much more consistent with the prompt throughout the entire optimization process, as reflected by the CS curve. The trajectory of $P(z)$ further corroborates that it is a good indicator of the OOD phenomenon, as it is positively associated with the CS. It is worth noting that the regularization effect provided by probability cannot be fully replicated by simply employing early-stopping in non-regularized optimization, as illustrated in Figure 1. We offer qualitative examples in Figure 2 to highlight the differences between optimized samples with and without regularization. Figure 2a and 2b are optimized images with whiteness reward and prompt "black rabbit". Figure 2c and 2d are optimized images with blackness reward and prompt "white chicken". In the two pairs of images, we select the optimized images that achieved the almost same reward value from the optimizations with and without regularization, respectively. It is evident that at an equivalent level of reward, the non-regularized samples are more prone to being out-of-distribution, as the color of the generated animals is less likely to match the prompt.

5. Extensions

Due to the page limit, we relegate some important extensions of DNO to the Appendix. In Appendix C, we discuss the extensions of DNO for optimizing non-differentiable reward functions. In Appendix B, we discuss an important design choice regarding the use of ODE or SDE. In Appendix D, we benchmark DNO against existing tuning-based alignment methods on three reward models trained on human feedback data.

6. Conclusions

In this work, we study Direct Noise Optimization (DNO) as a tuning-free method for aligning diffusion generative models. We introduce a probability regularization technique to efficiently address the out-of-distribution reward-hacking problem in DNO. The primary limitation of DNO lies in its

integration with the sampling process of diffusion models, leading to a substantial increase in processing time compared to direct sampling. Nonetheless, we argue that the additional time cost, being within a reasonable and acceptable range, is a worthwhile trade-off for attaining high-reward generated samples in a wide range of real-world applications.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-flow: Differentiating through flows for controlled generation. *arXiv preprint arXiv:2402.14017*, 2024.

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Lichang Chen, Chen Zhu, Davit Soselia, Jiuhai Chen, Tianyi Zhou, Tom Goldstein, Heng Huang, Mohammad Shoeybi, and Bryan Catanzaro. Odin: Disentangled reward mitigates hacking in rlhf. *arXiv preprint arXiv:2402.07319*, 2024a.

Weifeng Chen, Jiacheng Zhang, Jie Wu, Hefeng Wu, Xuefeng Xiao, and Liang Lin. Id-aligner: Enhancing identity-preserving text-to-image generation with reward feedback learning, 2024b.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. *arXiv preprint arXiv:2309.17400*, 2023.

Fei Deng, Qifei Wang, Wei Wei, Matthias Grundmann, and Tingbo Hou. Prdp: Proximal reward difference prediction for large-scale reward finetuning of diffusion models. *arXiv preprint arXiv:2402.08714*, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models, 2023.

Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. *arXiv preprint arXiv:2401.08639*, 2023.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling, 2023.

Zinan Guo, Yanze Wu, Zhuowei Chen, Lang Chen, and Qian He. Pulid: Pure and lightning id customization via contrastive alignment, 2024.

Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. *arXiv preprint arXiv:2212.09611*, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Jian Hu, Li Tao, June Yang, and Chandler Zhou. Aligning language models with offline learning from human feedback, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

Korrawe Karunratanakul, Konpat Preechakul, Emre Aksan, Thabo Beeler, Supasorn Suwajanakorn, and Siyu Tang. Optimizing diffusion noise can serve as universal motion priors. *arXiv preprint arXiv:2312.11994*, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation, 2023.

Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Bhalerao, Christopher L. Buckley, Jason Phang, Samuel R. Bowman, and Ethan Perez. Pretraining language models with human preferences, 2023.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models, 2019.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. Languages are rewards: Hindsight finetuning using human feedback, 2023a. URL https://arxiv.org/abs/2302.02676.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *arXiv preprint arXiv:2309.06380*, 2023b.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023.

Yuchun Miao, Sen Zhang, Liang Ding, Rong Bao, Lefei Zhang, and Dacheng Tao. Mitigating reward hacking via information-theoretic reward modeling. *arXiv preprint arXiv:2402.09345*, 2024.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled decoding from language models, 2023.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Zachary Novack, Julian McAuley, Taylor Berg-Kirkpatrick, and Nicholas J Bryan. Ditto: Diffusion inference-time t-optimization for music generation. *arXiv preprint arXiv:2401.12179*, 2024.

OpenAI. Chatgpt,https://openai.com/ blog/chatgpt/, 2022. URL https://openai.com/blog/chatgpt/.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022a.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022b.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pages 32483–32498. PMLR, 2023a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023b.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Jiao Sun, Deqing Fu, Yushi Hu, Su Wang, Royi Rassin, Da-Cheng Juan, Dana Alon, Charles Herrmann, Sjoerd van Steenkiste, Ranjay Krishna, and Cyrus Rashtchian. Dreamsync: Aligning text-to-image generation with image understanding feedback, 2023.

Zhiwei Tang, Dmitry Rybin, and Tsung-Hui Chang. Zeroth-order optimization meets human feedback: Provable learning via ranking oracles. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=TVDUVpgu9s.

Zhiwei Tang, Jiasheng Tang, Hao Luo, Fan Wang, and Tsung-Hui Chang. Accelerating parallel sampling of diffusion models. *arXiv preprint arXiv:2402.09970*, 2024b.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Sergey Levine, and Tommaso Biancalani. Feedback efficient online fine-tuning of diffusion models. *arXiv preprint arXiv:2402.16359*, 2024.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. *arXiv preprint arXiv:2311.12908*, 2023a.

Bram Wallace, Akash Gokul, Stefano Ermon, and Nikhil Naik. End-to-end diffusion latent optimization improves classifier guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7280–7290, 2023b.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. Openchat: Advancing open-source language models with mixed-quality data, 2023a.

Weizhi Wang, Khalil Mrini, Linjie Yang, Sateesh Kumar, Yu Tian, Xifeng Yan, and Heng Wang. Finetuned multimodal language models are high-quality image-text data filters. *arXiv preprint arXiv:2403.02677*, 2024.

Zijie J. Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models, 2023b.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis, 2023.

Yazhou Xing, Yingqing He, Zeyue Tian, Xintao Wang, and Qifeng Chen. Seeing and hearing: Open-domain visual-audio generation with diffusion latent aligners. *arXiv preprint arXiv:2402.17723*, 2024.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation, 2023.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Hui Yuan, Kaixuan Huang, Chengzhuo Ni, Minshuo Chen, and Mengdi Wang. Reward-directed conditional diffusion: Provable distribution estimation and reward improvement, 2023.

Huizhuo Yuan, Zixiang Chen, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning of diffusion models for text-to-image generation. *arXiv preprint arXiv:2402.10210*, 2024.

Shu Zhang, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, et al. Hive: Harnessing human feedback for instructional visual editing. *arXiv preprint arXiv:2303.09618*, 2023.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *arXiv preprint arXiv:2302.04867*, 2023.

A. Methods for Aligning Diffusion Models

A.1. Online/Offline Reinforcement Learning for Fine-tuning Diffusion Models

Aligning generative models with specific objectives, such as human preferences, has attracted extensive research efforts, following a series of successful attempts that aligns LLMs with Reinforcement Learning from Human Feedback (RLHF). For diffusion models, developing alignment algorithms has also gained sufficient attention recently, where a majority of recent research has focused on leveraging Reinforcement Learning (RL)-based approaches. A common mathematical formulation in RL-based methods is to maximize the expected reward while ensuring the resulting distribution does not deviate excessively from the original distribution. This can be expressed as the following KL-regularized optimization problem: $\max_p \mathbb{E}_{x\sim p(x)}[r(x)] - \lambda \mathrm{KL}(p||p_\theta)$. Current RL-based methods can be categorized into online and offline methods based on the data used. In the online method, the algorithm has the capacity to query the reward model throughout the entire optimization process. Two notable online RL methods are DDPO (Black et al., 2023) and DPOK (Fan et al., 2023). Alternatively, research has also delved into the offline RL optimization setting, where an explicit reward model is not accessible and only a fixed preference dataset is utilized. Noteworthy works in this category include Diffusion-DPO (Wallace et al., 2023a) and SPIN-Diffusion (Yuan et al., 2024).

A.2. Direct Fine-tuning of Diffusion Models with Differentiable Rewards

Two recent studies, AlignProp (Prabhudesai et al., 2023) and DRaFT (Clark et al., 2023), consider directly fine-tune diffusion models using differentiable rewards. Specifically, their optimization objective is formulated as: $\max_\theta \mathbb{E}_{z\sim\mathcal{N}(0,I)}[r(M_\theta(z))]$.

A.3. Loss-Guided Diffusion

A recent work focusing on loss-guided diffusion models (LGD) (Song et al., 2023a) also examines the concept of aligning diffusion models with differentiable rewards. Unlike the methods mentioned previously, LGD is **Tuning-Free**, meaning it does not necessitate modifications to the pretrained model θ. In essence, the core idea of LGD is that, during the sampling process for the ODE (1), it considers a modified version of the ODE by introducing a new term that guides the generation toward areas of higher reward. Specifically, the new ODE is: $dx_t = \left(f(t)x_t - \frac{1}{2}g^2(t)\epsilon(x_t,t) + \nabla_{x_t} r(x_0(x_t))\right) dt$, where $x_0(x_t)$ denotes the solution of this ODE starting from x_t. However, the gradient term $\nabla_{x_t} r(x_0(x_t))$ is not readily available during generation. To address this, the authors suggest utilizing Monte Carlo estimation to approximate the gradient. Nevertheless, this estimation tends to be noisy and imprecise, leading to suboptimal performance, particularly with complex reward models.

A.4. Comparing Existing Methods

Existing works can generally be categorized based on two criteria: whether it requires fine-tuning and whether the reward model needs to be differentiable.

Tuning-based or Tuning-free. All RL-based methods and the direct fine-tuning method are tuning-based, meaning they necessitate adjustments to the network models θ. There are two main disadvantages associated with tuning-based methods. The first is that they require fine-tuning for each new reward model, a process that can consume considerable resources, especially when faced with extensive choices for reward models. The second drawback is that the fine-tuning process typically relies on a limited set of input prompts, which challenges the model to generalize to new and unseen prompts. In contrast, methods such as LGD (Song et al., 2023a) belong to the tuning-free category. The main advantage of this approach is its elimination of the need for fine-tuning, allowing it to be flexibly applied in conjunction with any reward models and input prompts. Further, tuning-free methods require significantly fewer computing resources than the tuning-based methods. However, the major drawback of tuning-free methods is the substantial increase in the time required for the generation process. Also, our experiments in Section D demonstrate that the tuning-free method LGD cannot match the performance of tuning-based method.

Differentiable or Non-Differentiable. Current RL-based methods can work by utilizing solely the value or preference information of the reward functions. In contrast, the existing direct fine-tuning methods and LGD require the reward function to be differentiable.

B. Optimizing ODE vs. Optimizing SDE

As previously introduced, there are two primary methods for sampling from pretrained diffusion models: one based on solving the ODE (1) and the other on solving the SDE (2). A critical difference lies in that ODE sampling depends exclusively on the initial noise x_T, whereas SDE sampling is additionally influenced by the noise w_t added at every step of the generation process. It has been noted that existing works on noise optimization (Ben-Hamu et al., 2024; Novack et al., 2024; Karunratanakul et al., 2023) have mainly concentrated on optimizing the initial noise x_T for the ODE sampler.

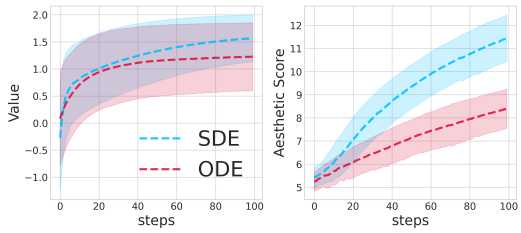


Figure 3. ODE vs. SDE for optimization

Different from preceding studies, we explore the utilization of the SDE sampler for noise optimization. Specifically, we employ the DDIM with $\eta = 1$ (Song et al., 2020a) as the SDE sampler and propose to optimize both the added noise w_t at every timestep and the initial noise x_T. In this context, the dimensionality of the optimized noise significantly surpasses that of ODE sampler. Empirically, we have observed that optimizing the SDE-based generation process can be significantly faster than its ODE-based counterpart. This is illustrated in Figure 3, where we juxtapose the optimization speeds between the ODE (DDIM with $\eta = 0$) and SDE samplers using both a toy example described above (Left) and optimizing for stable diffusion in alignment with the aesthetic reward (Right), a major experiment detailed in the subsequent section D. Intuitively, this acceleration in optimization speed can be attributed to the finer-grained control over the generation process afforded by the SDE sampler. Given these observations, our work will concentrate on optimizing the SDE sampler for the remainder of the study.

C. Tackling Non-Differentiable Reward Functions

As another major contribution of this work, we explore adapting the noise optimization approach to handle the optimization of non-differentiable reward functions by estimating the gradient with function values. Specifically, we explore three methods under this setting.

Method 1. Concerning optimization with only function value, a major family of optimization approaches is zeroth-order optimization algorithms, including ZO-SGD (Nesterov and Spokoiny, 2017). This method treats the entire mapping $r \circ M_\theta(\cdot)$ as a black-box function and seeks to estimate the gradient of $r \circ M_\theta(\cdot)$ via function value queries.

Method 2. It worth to note that for the mapping $r \circ M_\theta(\cdot)$, only the gradient of the reward function $r(x)$ is not available, and we are still able to compute the gradient of $M_\theta(z)$. Therefore, a straightforward idea is to adopt a hybrid gradient approach—only to estimate the gradient of r, while using the ground truth gradient for $M_\theta(z)$. Specifically, we denote that the initial noise is z, and the generated sample is $x = M_\theta(z)$. Firstly, we can estimate the gradient of $\nabla r(x)$ in a similar fashion with the ZO-SGD (Nesterov and Spokoiny, 2017):

$$H_1(x) = \mathbb{E}_{\xi \sim \mathcal{N}(0,I)} \left[(r(x + \mu\xi) - r(x))\, \xi \right] \approx C_1 \nabla r(x), \tag{8}$$

where μ is the coefficient for perturbation, and C_1 is some constant. With the estimated gradient $H_1(x)$, we can use the following gradient $H_1(x) \cdot \nabla_z M_\theta(z)$ for optimization, where the main idea is to replace the ground truth $\nabla r(x)$ in the chain-rule of differentiating $r(M_\theta(z))$. We refer this method as **Hybrid-1** in the following sections.

Method 3. We found that there is a crucial drawback in the gradient estimator (8), that one needs to query the reward model $r(\cdot)$ with noisy input $x + \mu\xi$. When the reward model is only defined on some manifold $\mathcal{M}$, e.g., defined on the image manifold, rather than the whole space $\mathbb{R}^n$, this can lead to severe problems, because, for some $x \in \mathcal{M}$, the noisy sample

$x + \mu\xi$ may no longer stay within $\mathcal{M}$. To remedy this issue, we propose to perturb the sample through the latent noise, rather than directly in the sample space. Specifically, our proposed new gradient estimator for $\nabla r(x)$ is

$$H_2(x) = \mathbb{E}_{\xi\sim\mathcal{N}(0,I)}\left[\left(r(M_\theta(z+\mu\xi)) - r(x)\right)\left(M_\theta(z+\mu\xi) - x\right)\right]. \tag{9}$$

Following a similar proof in (Nesterov and Spokoiny, 2017), we can also show that $H_2(x) \approx C_2\nabla r(x)$ for some constants C_2. As we can see, when computing the gradient (9), we ensure that we query the reward model $r(\cdot)$ with only samples that are within the manifold of the pretrained distribution. We refer this method as **Hybrid-2**. As we will see in Section C.1, this Hybrid-2 method is significantly faster than the other two in terms of optimization speed. In Appendix G, we describe an efficient way to implement (8) and (9) with auto-differentiation technique.

C.1. Experiment for Optimizing Non-Differentiable Reward Functions

Setting. In this section, we aim to explore the three extensions proposed in Section C for handling non-differentiable reward functions. For Method 1, we use ZO-SGD (Nesterov and Spokoiny, 2017), and compare it with our proposed Hybrid-1 and Hybrid-2. We consider using two reward functions: The Jpeg Compressibility score, which is the file size of the image after compressing it using the JPEG algorithm. This reward model was also used in (Black et al., 2023) and is intrinsically non-differentiable. The second reward function is the aesthetic score used in the previous section. We compare the three methods in terms of optimization steps and the final score of the reward function. For ease of demonstration, we do not add the probability regularization term to the optimization process. Moreover, it is important to note that in these three methods, the major time expenditure comes from estimating the gradient with finite samples. For a fair comparison among these methods, we set the number of samples for estimating the gradient separately so that the total time spent estimating the gradient is the same for all three methods.

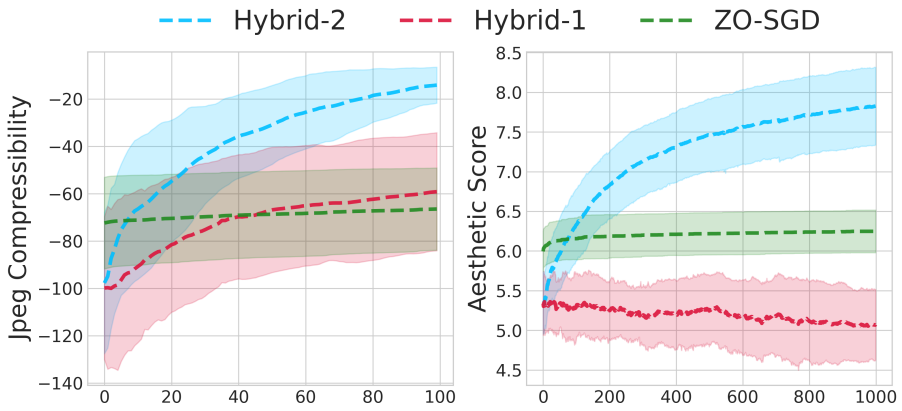


Figure 4. Comparing three methods on two reward functions. For better visualization purposes, when plotting the line for ZO-SGD, we compute and plot the current best reward instead of the current reward due to the extremely high variance.

Results. The main results are visualized in Figure 4. Initially, we observe that the Hybrid-2 method is significantly faster than the other two methods, and the final score is also higher for both reward functions used in this experiment. Furthermore, in both scenarios, the ZO-SGD method exhibits the slowest optimization speed. Another interesting finding is the poor performance of Hybrid-1 in optimizing the Aesthetic Reward. This mainly occurs because the aesthetic reward model can only work with image inputs, which validates our initiative to propose the Hybrid-2 method in Section C. Finally, when comparing the optimization of the aesthetic score in Figures 5 and 4, we can also note that using the true gradient results in substantially fewer steps than using the estimated gradient.

D. Benchmarking Three Reward Models Trained on Human Feedback Data

Setting. In this section, we investigate the performance of the proposed method using three common reward models trained from human feedback data, specifically Aesthetic Score (Schuhmann et al., 2022b), HPS-v2 score (Wu et al., 2023), and

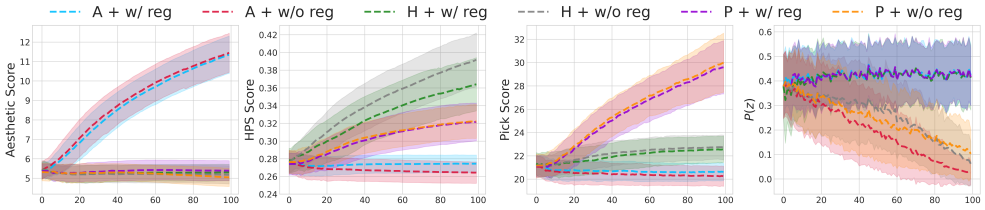


Figure 5. A, H, P are short for Aesthetic Score, HPS Score, and Pick Score, respectively. The name for each line comprises the used reward model and whether the regularization is used. For example, A + w/ reg means optimizing aesthetic score with regularization. The x-axis is the gradient steps.

Method	SD v1.5	LGD		SPIN	DDPO	AlignProp	PRNO (This work)			
	2s	n=10	n=100	$\sim$ 20h	$\sim$ 56h	$\sim$ 12h	1 min	3 min	5 min	10 min
Aesthetic ↑	5.367	5.340	5.224	6.248	7.180	8.940	5.754	7.202	8.587	10.68
HPS ↑	0.278	0.276	0.271	0.276	0.287	0.330	0.285	0.303	0.324	0.354
PickScore ↑	21.11	21.01	21.09	22.00	/	/	21.25	23.17	25.13	25.16

Table 1. Performance comparison. For SD v1.5 and PRNO, we annotate the generation time below the name. For LGD, we annotate the number of samples used for Monte Carlo approximation. For SPIN, DDPO, and AlignProp, we annotate the estimated time for fine-tuning. All time are measured with respect to the GPU time on an A800.

PickScore (Kirstain et al., 2023), respectively. In this experiment, we also compare noise optimization with and without probability regularization. However, compared to the reward function used in the previous section, using these reward models present a lesser chance for the optimized image to be OOD. In this case, to measure the benefit of probability regularization in maintaining the quality of the generated sample, we consider using the other two reward models as test metrics when optimizing one of them. For example, when optimizing the Aesthetic Score, we use the HPS Score and Pick Score as test metrics.

Results. Firstly, we observe that the effect of probability regularization is less pronounced than that in Section 4. This observation is also reflected by our proposed indicator $P(z)$; if no regularization is applied, the value of $P(z)$ in Figure 5 decreases much slower than that in Figure 1. Nonetheless, by adding the regularization term to noise optimization, we can stabilize the value of $P(z)$ throughout the optimization process and also improve the test metrics. For instance, when optimizing the aesthetic reward, the regularization has no significant effect on the optimization speed, while it prevents the test metrics, i.e., the HPS score and Pick Score, from decreasing throughout the process.

Comparing to Existing Alignment Methods. We also summarize the performance of our proposed method in Table 1 and compare it to the major existing alignment methods discussed in the introduction. As we can see, the proposed method can easily outperform state-of-the-art tuning-based alignment methods without any fine-tuning on the network models, within a reasonable time budget for generation. We found that another tuning-free method, LGD (Song et al., 2023a), performs badly with these complex reward functions, since it is impossible to estimate the gradient of reward functions without a complete generation process. Finally, we wish to highlight that this alignment process can run on only **1 consumer-level GPU**, with memory usage less than **15GB**. In contrast, current tuning-based methods require much more computing resources, typically 8 advanced GPUs.

E. Theoretical Results

Proof for Theorem 2.1. To leverage the L-smoothness assumption, we need to state a classical lemma for smooth optimization.

Lemma E.1 (Descent Lemma (Bertsekas, 1997))**.** *For any z_1 and z_2, we have*

$$r \circ M_\theta(z_2) \geq r \circ M_\theta(z_1) + \nabla r \circ M_\theta(z_1) \cdot (z_2 - z_1) - \frac{L}{2}\|z_2 - z_1\|_2^2. \tag{10}$$

Now for any z and steps $t \geq 1$, with the descent lemma, we have

$$r \circ M_\theta(g_{t+1}(z)) \geq r \circ M_\theta(g_t(z)) + \nabla r \circ M_\theta(g_t(z)) \cdot (g_{t+1}(z) - g_t(z)) - \frac{L}{2}\|g_{t+1}(z) - g_t(z)\|_2^2.$$

Notice that by the definition of $g_t\cdot$ we have

$$\begin{aligned} g_{t+1}(z) - g_t(z) &= g_t(z) + \ell \nabla r \circ M_\theta(g_t(z)) - g_t(z) \\ &= \ell \nabla r \circ M_\theta(g_t(z)). \end{aligned}$$

Therefore, we have

$$r \circ M_\theta(g_{t+1}(z)) \geq r \circ M_\theta(g_t(z)) + \left(\ell - \frac{\ell^2 L}{2}\right) \|\nabla r \circ M_\theta(g_t(z))\|_2^2.$$

Taking the expectation over $z \sim \mathcal{N}(0, I)$ we have

$$\begin{aligned} \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[r \circ M_\theta(g_{t+1}(z))\right] \geq \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[r \circ M_\theta(g_t(z))\right] + \\ \left(\ell - \frac{\ell^2 L}{2}\right) \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\|\nabla r \circ M_\theta(g_t(z))\|_2^2\right]. \end{aligned}$$

By using the change of variable formula for distribution, we can easily see that

$$\mathbb{E}_{x\sim p_{t+1}(x)} r(x) = \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[r \circ M_\theta(g_{t+1}(z))\right],$$

and

$$\mathbb{E}_{x\sim p_t(x)} r(x) = \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[r \circ M_\theta(g_t(z))\right],$$

Therefore, we conclude with

$$\begin{aligned} \mathbb{E}_{x\sim p_{t+1}(x)} r(x) &\geq \mathbb{E}_{x\sim p_t(x)} r(x) + \left(\ell - \frac{\ell^2 L}{2}\right) \mathbb{E}_{z_0\sim N(0,I)} \left\|\nabla_z r \circ M_\theta(z)_{|z=g_t(z_0)}\right\|_2^2 \\ &\geq \mathbb{E}_{x\sim p_t(x)} r(x). \end{aligned} \tag{11}$$

□

When Does the Distribution Stop Improving? As observed, the distribution ceases to improve when the second term in Equation (11) becomes zero. Initially, we note that the optimized distribution stops improving when $\mathbb{E}_{z_0\sim N(0,I)} \left\|\nabla_z r \circ M_\theta(z)_{|z=g_t(z_0)}\right\|_2^2 = 0$. In statistical terms, this implies that $\nabla_z r \circ M_\theta(z)_{|z=g_t(z_0)}$ is a zero vector with probability one.

To discern the circumstances under which this zero vector occurs, let us assume that z_0 is some fixed noise vector and consider the scenario where

$$\nabla_x r(x)_{|x=M_\theta(g_t(z_0))} \cdot \nabla_z M_\theta(z)_{|z=g_t(z_0)} = \vec{0}, \tag{12}$$

Here, we denote $G_1(z_0) = \nabla_x r(x)_{|x=M_\theta(g_t(z_0))}$ as the gradient of the reward models on the generated sample, and $G_2(z_0) = \nabla_z M_\theta(z)_{|z=g_t(z_0)}$ representing the Jacobian matrix of the noise-to-sample mapping. We categorize the situation in Equation (12) into three cases:

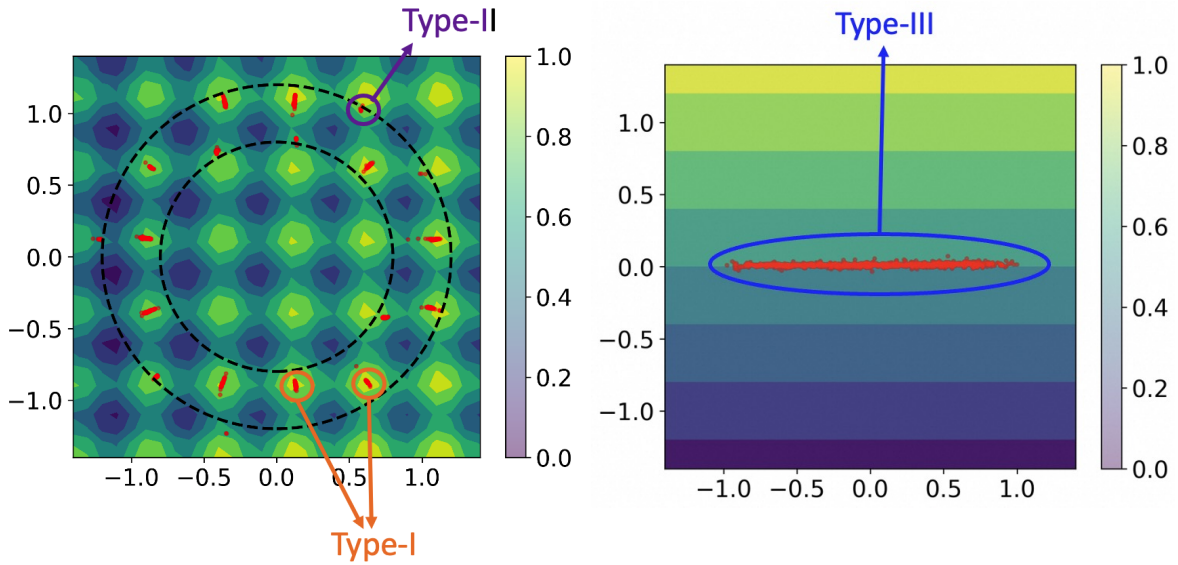
Type-I: $\|G_1(z_0)\| = 0$ and $\|G_2(z_0)\| > 0$. Here, the gradient of the reward model on the generated sample is zero, indicating that the generated sample has reached a stationary point (or local solution) of the reward model.

Type-II: $\|G_2(z_0)\| = 0$ and $\|G_1(z_0)\| > 0$. This indicates that the Jacobian matrix of the noise-to-sample mapping is zero, which often suggests that the generated sample is at the boundary of the support of the distribution, as a zero Jacobian means that changes in the noise will not affect the generated sample.

Type-III: $\|G_1(z_0)\| > 0$ and $\|G_2(z_0)\| > 0$, but $\|G_1(z_0) \cdot G_2(z_0)| = 0$. In this scenario, the gradient of the reward model on the generated sample is orthogonal to the Jacobian matrix of the noise-to-sample mapping.

In summary, the distribution will halt its improvement after the t-th step if it almost surely holds that z_0 corresponds to a Type-I, Type-II, or Type-III noise vector.

We provide examples for the three scenarios, respectively, in the following figures. First, in Figure 6a, we display examples of Type-I and Type-II by reutilizing the experiment from Figure **??**. To determine the type of the noise vector, we empirically compute $\|G_1(z_0)\|$, $\|G_2(z_0)\|$, and $\|G_1(z_0) \cdot G_2(z_0)\|$ for each noise vector.



(a) Example 1 (b) Example 2

Figure 6. Examples of generated samples with Type-I, Type-II and Type-III noise vectors in the toy examples.

To showcase an example of Type-III noise vectors, we introduce a new toy example illustrated in Figure 6b. Specifically, the ground-truth distribution learned by diffusion models is uniform across a horizontal line spanning from $(-1, 0)$ to $(1, 0)$. The reward function is defined as $r(x, y) = y$. It can be readily confirmed that, for every point on this line, the gradient of the reward model is orthogonal to the Jacobian matrix of the noise-to-sample mapping. Consequently, all points along the line segment $[(-1, 0), (1, 0)]$ qualify as Type-III noise vectors.

F. Empirical Investigation of $P(z)$

In this section, we provide several empirical evidence to demonstrate that $P(z)$ acts as an effective indicator for the out-of-distribution phenomenon.

Firstly, in Figure 7a, we revisit the examples from Figures **??** and **??**, coloring each sample based on the value of $P(z)$. As depicted in Figure 7a, $P(z)$ proves to be an efficient metric to separate in-distribution samples from out-of-distribution samples; those in-distribution have high values for $P(z)$, whereas those out-of-distribution exhibit values of $P(z)$ near zero.

Secondly, we manually construct several noise vectors that reside in the low-probability region of the standard Gaussian distribution. To establish a baseline comparison, we first draw one sample from the standard Gaussian distribution and use it to generate an image with Stable Diffusion v1.5 and the prompt "black duck". As can be seen, this leads to a normal image with $P(z)$ also within a reasonably large value. We then construct the low-probability vectors in two ways. The first one is to use all-zero vectors, which obviously reside in the low-probability zone of high-dimensional Gaussian distributions. The generated images with all-zero vectors are visualized in Figure 7c, showcasing that there is nothing discernible in the image while $P(z)$ approximates zero. The second method is to repeat parts of the noise vectors, such that the noise vectors exhibit high covariance in the elements. Specifically, we construct the repeated vectors by first generating an $n/4$ dimensional z_0 from the standard Gaussian distribution, and then constructing the noise vectors as $z = [z_0, z_0, z_0, z_0]$, making z an n dimensional vector. The figure corresponding to these repeated vectors, shown in Figure 7d, once again results in a poor image, with $P(z)$ illustrating that the noise vectors also come from a low-probability region.

We further visualize the entire optimization trajectory for the examples in Figures **??** and **??**, i.e., optimizing the whiteness reward for Stable Diffusion v1.5 with the prompt "black duck" in Figure 8. Specifically, from Figure 8 we can clearly see that the value of $P(z)$ gradually decreases, and the generated image also gradually diverges from the distribution associated with a black duck. Notably, at around 20 steps, the value of $P(z)$ becomes near-zero, and at the same time, the generated image more closely resembles a blue duck rather than the specified black duck.

Similarly, we visualize the optimization trajectory for optimizing the Aesthetic Score for SD v1.5 with the prompt "black duck". The results are in Figure 9. A clear conclusion is that in this case, it is less likely for the optimized samples to be out-of-distribution. This is mainly because the Aesthetic Score itself penalizes those OOD samples. It is noteworthy to observe that this insight is also captured by our proposed indicator $P(z)$, because when comparing the trend of $P(z)$ in Figure 8 and Figure 9, we can see that optimizing the Aesthetic Score leads to a much less significant decrease in the $P(z)$ value.

(a) Toy Example (b) $P(z) = 0.36$ (c) $P(z) = 0.00$ (d) $P(z) = 0.02$

Figure 7. Examples of generated samples with corresponding values of $P(z)$.

G. Implementation Details

In this section, we discuss some implementation details of our proposed method, as well as clarify some omissions in the experimental section.

G.1. Algorithm Implementation

It is clear that to solve the direct noise optimization problem stated in Problem 3, differentiation of the noise-to-sample mapping M_θ is required. It is worth noting that this differentiation cannot be handled by standard auto-differentiation in

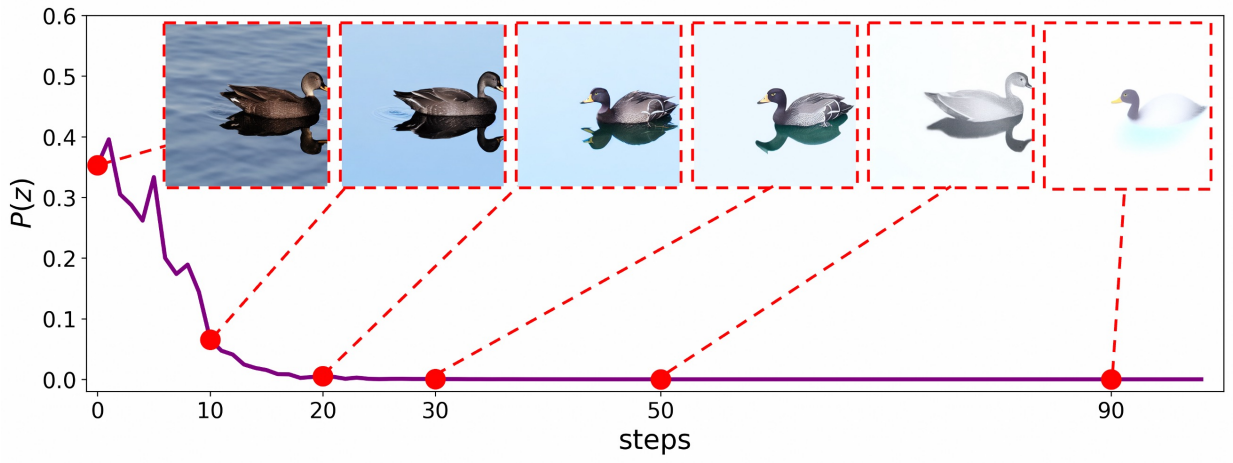


Figure 8. Trajectory of $P(z)$ on optimizing whiteness reward.

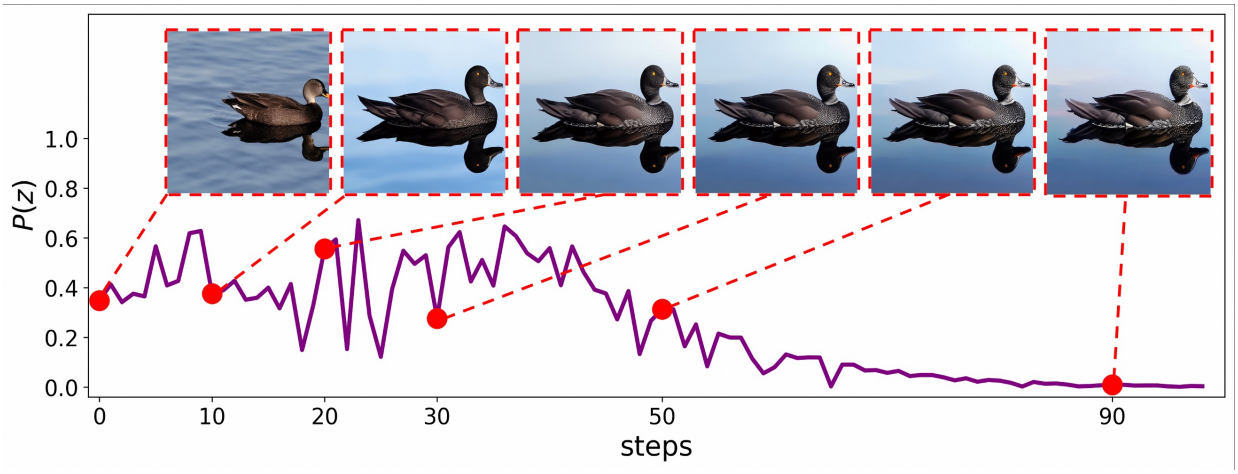


Figure 9. Trajectory of $P(z)$ on optimizing Aesthetic Score.

PyTorch (Paszke et al., 2019), as it can lead to a memory explosion. A common technique to resolve this issue is gradient checkpointing, which has also been adopted by other related works on noise optimization (Wallace et al., 2023b; Novack et al., 2024; Karunratanakul et al., 2023).

Here, we describe an efficient method to implement our proposed hybrid gradient estimators detailed in Section C, along with the optimization process, by utilizing the built-in auto-differentiation in PyTorch (Paszke et al., 2019). Specifically, suppose we wish to use q samples to estimate the gradient in Equation (9); that is, we draw q noise vectors for perturbation: $\xi_1, ..., \xi_q$. We then generate the corresponding samples $x_i = M_\theta(z + \mu\xi_i)$ for $i = 1, ..., q$. At this point, we should compute the estimated gradient of the reward models in a non-differentiable mode as follows:

$$\hat{H}_2(x) = \frac{1}{q}\sum_{i=1}^{q}(r(M_\theta(x_i)) - r(x))(x_i - x). \tag{13}$$

Finally, we can execute gradient backpropagation with the loss function,

$$loss(z) = \langle \hat{H}_2(x), M_\theta(z) \rangle,$$

which produces the exact gradient estimator for z.

G.2. Experiment Details

In this section, our goal is to provide the experimental details that were omitted from Sections 4, D, and C.1.

Details for Section 4. In this experiment, to solve the PRNO problem as formulated in Equation (7), we employ the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.01. For optimization with regularization, we set the regularization coefficient γ to 1. To compute the minibatch stochastic gradient for the regularization term in Equation (7), we set the batch size b—the number of random permutations drawn at each step—to 100. For each optimization run, we utilize a single A800 GPU, with the total memory consumption being approximately 15 GB.

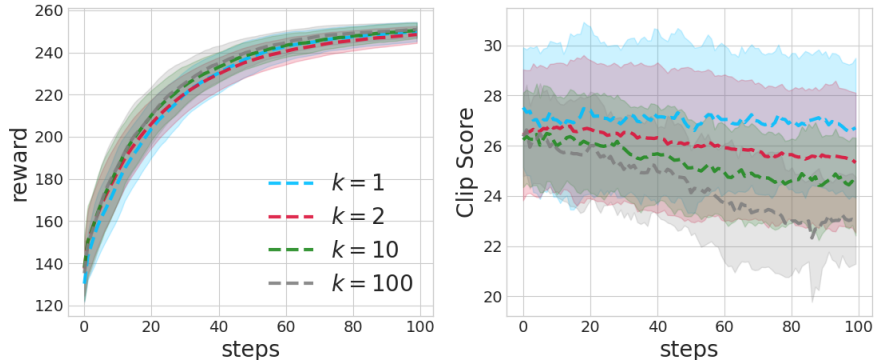
Details for Section D. In this second set of experiments, we continue using the Adam optimizer with a learning rate of 0.01. For optimization with regularization, though, we reduce the regularization coefficient to $\gamma = 0.1$ because optimizing these human-like reward functions is less susceptible to the OOD reward-hacking issue, while maintaining the batch size for the permutation matrix b at 100. Each optimization run also uses a single A800 GPU, but the total memory consumption is around 20 GB. Regarding the baselines in Table 1, we implemented LGD (Song et al., 2023a) ourselves, following the algorithm from their paper on these reward models. For other baselines, we reuse the statistics presented in their corresponding papers.

Details for Section C.1. In this section, the primary hyperparameters for the three tested algorithms are the perturbation coefficient μ and the number of samples q used to approximate the gradient (as formulated in Equation (13)). Clearly, q plays a crucial role in determining the running time of each algorithm. For an equitable comparison, we tune q separately for each algorithm to achieve roughly the same time cost per gradient step. Specifically, we set q values for ZO-SGD, Hybrid-1, and Hybrid-2 to 16, 8, and 4 respectively. For μ, we also adjust them individually for each algorithm, as they have varying sensitivity to μ. Through trial and error, we select μ values of 0.01 for ZO-SGD and Hybrid-1, and 0.02 for Hybrid-2. Finally, for optimizing JPEG Compressibility, we use the Adam optimizer with a learning rate of 0.01, but for the Aesthetic Score experiment, we reduce the learning rate to 0.001, as we found that 0.01 can lead to divergence during optimization for the Aesthetic Score. Each optimization run continues to use a single A800 GPU.

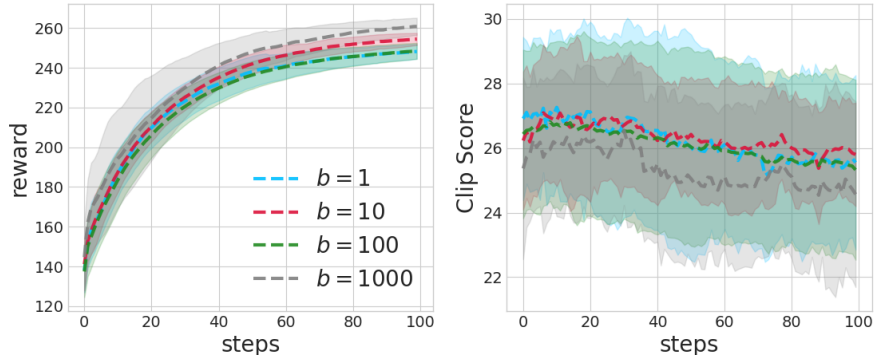
H. Hyperparameters Analysis

In this section, we conduct a thorough analysis of the hyperparameters for the proposed method. Our objective is to offer a concise guideline for selecting the hyperparameters in the proposed method.

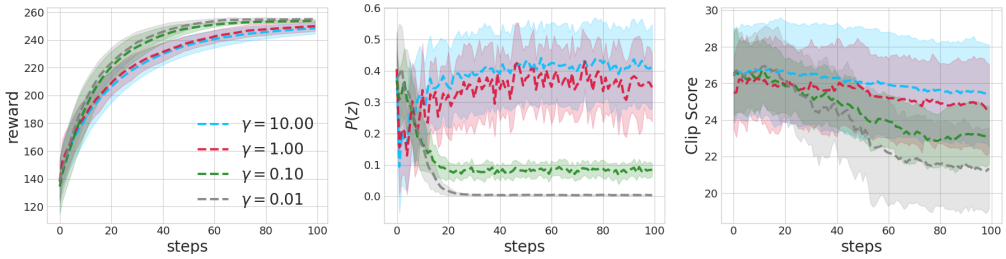
As discussed in Section 3.1, the concentration inequalities involve a hyperparameter k, which represents the dimension of subvectors from the noise vectors z that we aim to assess probabilistically. As noted in Remark 3.2, the dimension k should be neither too large nor too small. Additionally, another critical hyperparameter is the number of permutation matrices b employed to compute the stochastic gradient for the probability regularization in Equation (7). Furthermore, we aim to explore the impact of the regularization coefficient γ in the probability regularization term.



(a) The effect of k.



(b) The effect of b.



(c) The effect of γ.

Figure 10. Hyperparamet Analysis

To examine the effects of k, b, and γ on mitigating the OOD (Out-Of-Distribution) reward-hacking problem, we revisit the experiment of optimizing blackness reward with the prompt "white ¡animals¿" from Section 4. In Figure 10, we illustrate how these three hyperparameters influence both the reward and the consistency score (CS), across four different values.

Firstly, Figure 10a supports the notion that k should be carefully chosen—not too large, yet not overly small. We observe that $k = 1$ underperforms compared to $k = 2$ and $k = 10$, as selecting $k = 1$ fails to account for the covariance among noise vectors. Conversely, $k = 100$ proves to be a poor choice because it entails a smaller m, potentially rendering the concentration inequalities detailed in Lemma 3.1 less precise.

Secondly, as demonstrated in Figure 10b, the number of permutation matrices b seems to have a minor impact on the optimization process, provided b is sufficiently large. Based on empirical evidence, $b = 100$ emerges as an optimal selection for the proposed method.

Lastly, the effects of γ are depicted in Figure 10c. Adjusting the value of γ clearly presents a trade-off between convergence speed and the propensity for OOD reward-hacking problems. Given this observation, we recommend empirically tuning the value of γ for different reward functions and prompts using a limited number of samples and a few optimization steps.

I. Visualization of Optimization with Images

To assist the reader in understanding the optimization process, we offer visualizations of this process through images. The aim is to provide a qualitative evaluation of the proposed method.

I.1. Comparing With and Without Regularization

In this section, our objective is to demonstrate the impact of the probability regularization term on the optimization process through visual examples. Specifically, we present examples from three settings. The first two examples are derived from the experiments in Section 4, while the last example is from the experiment on optimizing the aesthetic score in Section D.

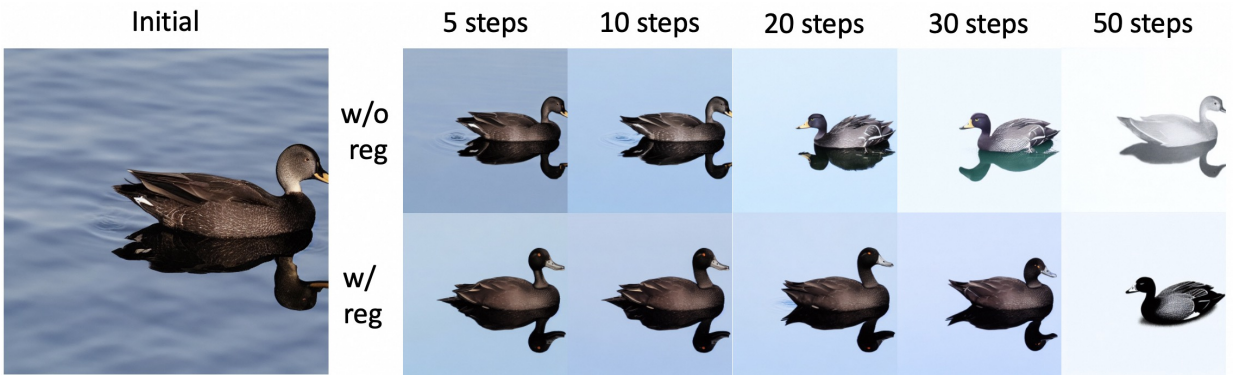
The examples can be seen in Figure 11. As observed across all examples, the optimization process that integrates the regularization term consistently prevents the generated samples from falling into the category of being Out-Of-Distribution (OOD).

I.2. More Examples

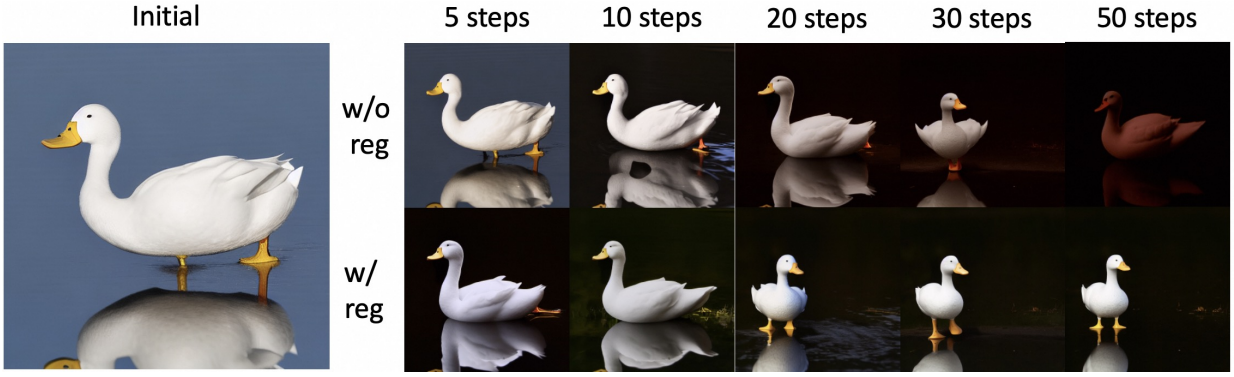
In this section, we aim to provide additional visualized examples for our proposed method.

Firstly, in Figure 12, we present examples from the optimization of all three popular human-level reward models discussed in Section D. As can be observed, the optimization process indeed results in an increase in human preference throughout.

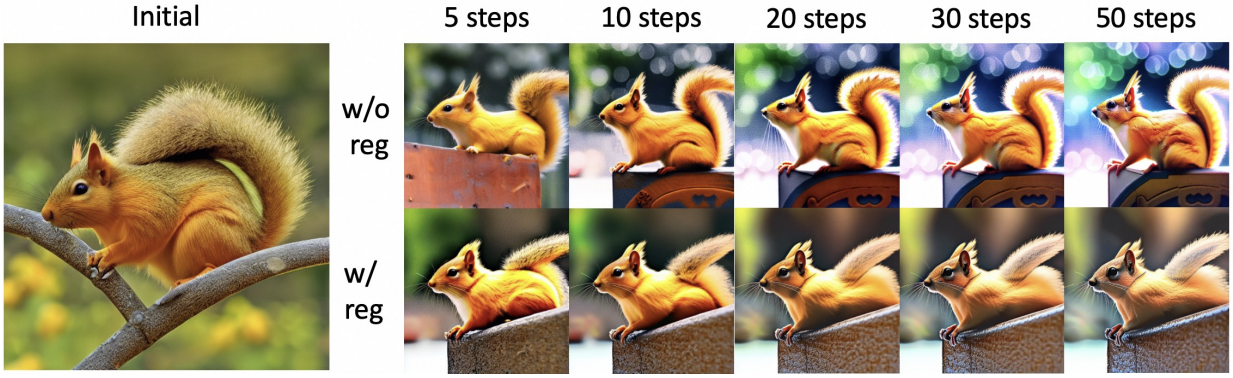
Furthermore, we also include examples from the experiments in Section C.1, that is, optimizing JPEG Compressibility Score and Aesthetic Score using the Hybrid-2 method for non-differentiable optimization. These examples in Figure 13 effectively showcase the efficiency of Hybrid-2 in both estimating the gradient and optimizing.



(a) Optimizing whiteness reward with and without probability regularization. Prompt: ”black duck”.



(b) Optimizing blackness reward with and without probability regularization. Prompt: ”white duck”.

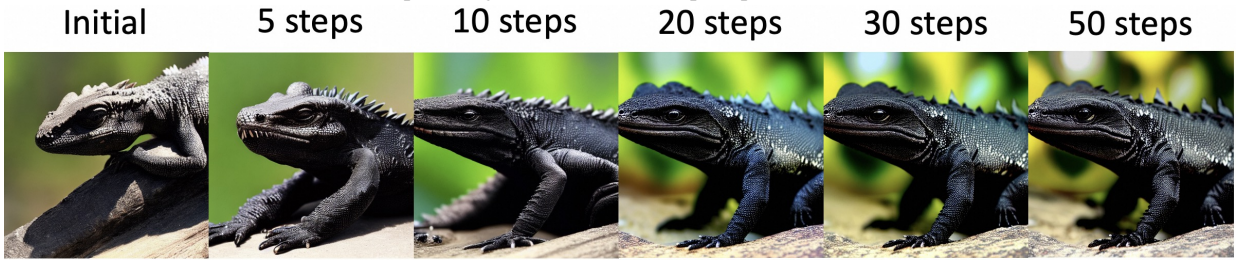


(c) Optimizing Aesthetic Score with and without probability regularization. Prompt: ”yellow squirrel”

Figure 11. Examples of optimized samples with and without regularization

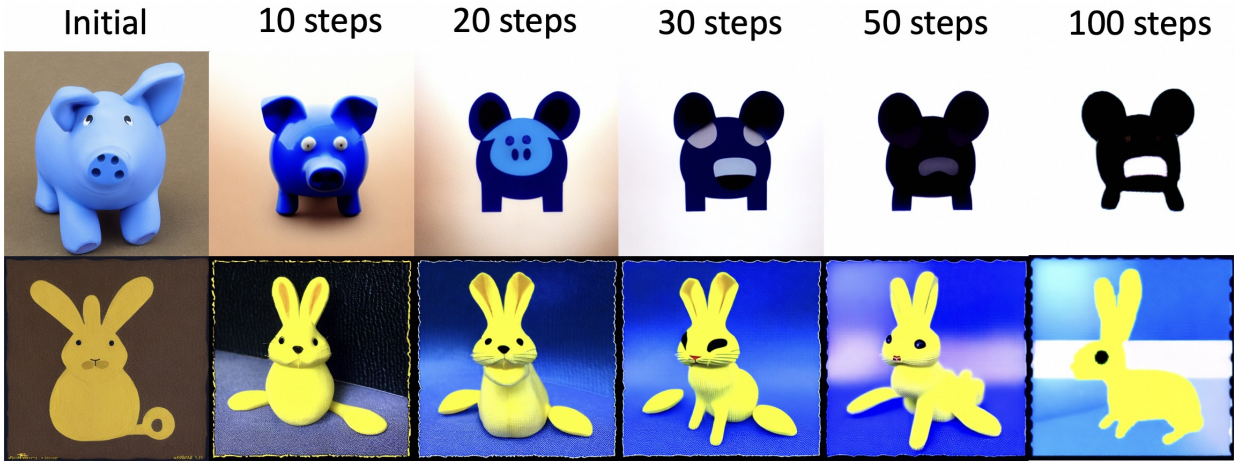
(a) Optimizing Aesthetic Score with prompt ”gray lion”

(b) Optimizing HPS v2 Score with prompt ”black deer”

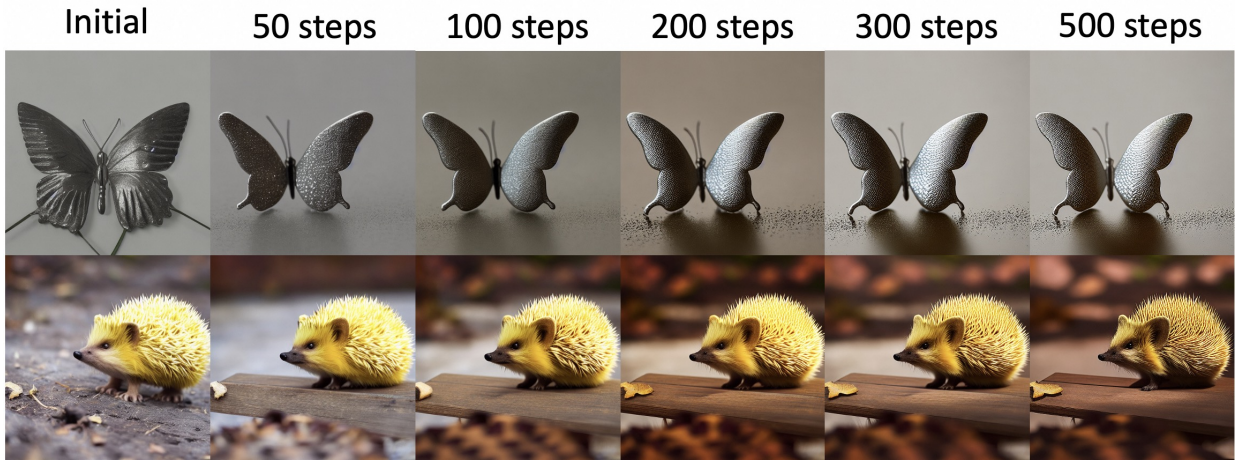


(c) Optimizing PickScore with prompt ”black lizard”

Figure 12. Representative examples of optimizing reward models trained on human feedback data.



(a) Optimizing Jpeg Compressiblity with Hybrid-2 gradient approximation. Upper: prompt "blue pig". Lower: prompt "yellow rabbit".



(b) Optimizing Aesthetic Score with Hybrid-2 gradient approximation. Upper: prompt "silver butterfly". Lower: prompt "yellow hedgehog".

Figure 13. Representative examples for non-differentiable optimization