Stabilizing Policy Gradients for Stochastic Differential Equations via Consistency with Perturbation Process

Xiangxin Zhou [1 2] **Liang Wang** [1 2] **Yichi Zhou** [3]

Abstract

Considering generating samples with high rewards, we focus on optimizing deep neural networks parameterized stochastic differential equations (SDEs), the advanced generative models with high expressiveness, with policy gradient, the leading algorithm in reinforcement learning. Nevertheless, when applying policy gradients to SDEs, since the policy gradient is estimated on a finite set of trajectories, it can be ill-defined, and the policy behavior in data-scarce regions may be uncontrolled. This challenge compromises the stability of policy gradients and negatively impacts sample complexity. To address these issues, we propose constraining the SDE to be consistent with its associated perturbation process. Since the perturbation process covers the entire space and is easy to sample, we can mitigate the aforementioned problems. Our framework offers a general approach allowing for a versatile selection of policy gradient methods to effectively and efficiently train SDEs. We evaluate our algorithm on the task of structure-based drug design and optimize the binding affinity of generated ligand molecules. Our method achieves the best Vina score (-9.07) on the CrossDocked2020 dataset.

1 Introduction

Deep neural networks parameterized stochastic differential equations (SDEs) have garnered significant interest within the machine learning community, owing to their robust theoretical underpinnings and exceptional expressiveness. Recently, SDEs have taken center stage in generative modeling (Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021b), with a myriad of applications ranging from image generation to molecule generation, and beyond (see Yang et al. (2022) for a comprehensive survey). SDE generates samples by simulating the following process from time 1 to 0:

$$dx_t = \pi_\theta(x_t, t)dt + g(t)d\bar{\omega} \tag{1}$$

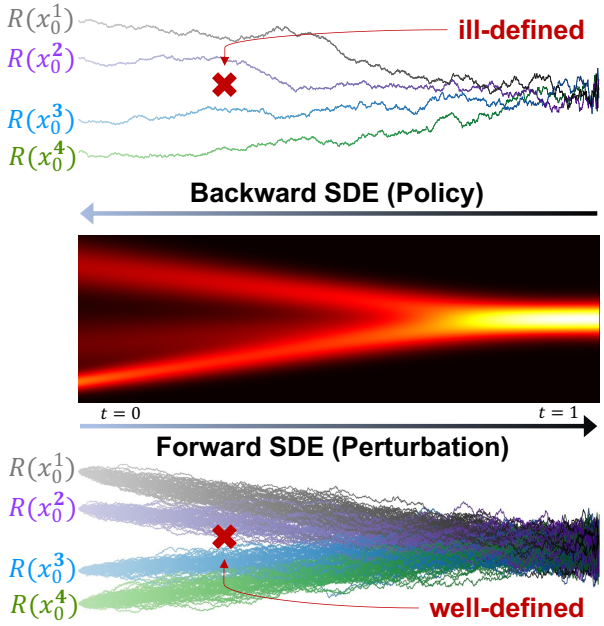

Figure 1. Illustration of our motivation and the advantages of our method. **Top**: Trajectories sampled by SDE-based policy. The terminal states (i.e., generated samples at $t = 0$) are denoted as $x_0^1, x_0^2, x_0^3, x_0^4$. The reward function is denoted as $R(\cdot)$. A data-scarce region is marked with a red cross. **Middle**: Marginal distributions of consistent forward and backward SDEs over time t. **Bottom**: Trajectories perturbed from $x_0^1, x_0^2, x_0^3, x_0^4$ via the forward SDE. In vanilla SDE-based policy gradient methods (top), due to substantial expense of computation and time required by SDE simulation, the sampled trajectories and rewards are usually sparse. Therefore, the policy gradients in data-scarce regions are ill-defined, leading to instability. The consistency which can be ensured via score matching allows us to correctly estimate the policy gradients with sufficient data that can be efficiently sampled from the forward SDE (i.e., perturbation).

[1]School of Artificial Intelligence, University of Chinese Academy of Sciences [2]New Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA) [3]ByteDance Research. Correspondence to: Yichi Zhou <zhouyichi.123@bytedance.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

where x_0 is the generated sample, x_1 is typically drawn from $\mathcal{N}(0, I)$, π_θ is the drift parameterized by a neural network ϵ_θ, $\bar{\omega}$ is the reverse Wiener process, and $g(t)$ is a scalar function of time and known as diffusion coefficient. Generative modeling aims to approximate the data distribution and, therefore, trains SDEs to maximize the likelihood. However, in many real-world applications, the objective is to maximize a reward. For example, in structure-based drug design (Anderson, 2003), our objective is usually to generate molecules with some desired properties, such as high binding affinity (Du et al., 2016). In these cases, it is natural to train SDEs with reinforcement learning (RL) (Sutton & Barto, 2018; Black et al., 2023; Fan et al., 2023).

RL focuses on learning a policy that maximizes rewards through interactions with a given environment. The foremost class of policy optimization algorithms in RL is the policy gradient method (Sutton & Barto, 2018), which refines the policy network by following the gradient of expected rewards with respect to the policy network parameters. Policy gradient algorithms have demonstrated efficiency and effectiveness in a wide range of applications, such as control systems, robotics, natural language processing, and game playing, among others.

The SDE in Equation (1) can be interpreted as a Markov Decision Process (MDP) (Black et al., 2023). Therefore, we can directly apply policy gradient to training SDEs. However, in practice, policy gradients are typically estimated using a limited set of sampled trajectories, which can lead to two practical issues (please see Section 3 for detailed discussion): (a) Insufficient data in the vicinity of a trajectory can result in an inaccurate estimation of the policy gradient. This ill-defined policy gradient may cause instability during the training process; and (b) When the simulation of the SDE in Equation (1) encounters data-scarce regions, the policy in these regions may not be well-trained, leading to uncontrolled behavior and less efficient interactions with the environment. These challenges have hindered the application of policy gradients to SDEs.

Contribution: To address the aforementioned challenge, we present a novel framework for training SDEs using policy gradient methods. The main idea behind our method is to enforce the SDE in Equation (1) to be consistent with its associated perturbation process (refer to Definition 4.1 for details). Given that the perturbation process covers a wide space and is easy to sample, we can resolve the aforementioned challenge, and further estimate the policy gradient in a more stable and efficient way. This consistency can be easily maintained using techniques derived from diffusion models. Moreover, we introduce an innovative actor-critic policy gradient algorithm tailored for consistent SDEs. We empirically validate our algorithm on structure-based drug design (SBDD, Anderson, 2003), and optimizes the binding affinity, one primary objective in SBDD, of generated molecules. Our algorithm achieves state-of-the-art Vina scores (-9.07) on the CrossDocked2020 dataset (Francoeur et al., 2020). Additionally, we have demonstrated the effectiveness and efficiency of our method on text-to-image generation task. This reveals the generalizability of our method and shows its great potential in various real-world applications.

2 Preliminary

We discuss preliminary information in this section including reinforcement learning, SDE-based generative models, and the approach to modeling SDEs as a Markov Decision Process. Extended preliminaries about SDEs and diffusion models can be found in Appendix A.

2.1 Reinforcement Learning and Policy Gradients

Reinforcement learning uses the Markov Decision Process (MDP) to model the decision-making process. An MDP is a tuple $(S, A, \mathcal{T}, R, \tilde{P})$ where S is the state space, A is the action space, $\mathcal{T} : S \times A \to \Delta(S)$ is a transition kernel where $\Delta(S)$ denote the set of distributions over S, $R : S \times A \to \mathbb{R}$ is a reward function and $\tilde{P}$ is the initial state distribution. Without loss of generality, we assume the MDP terminates after N steps.

The RL agent takes actions sampled from a policy π, which is a mapping from $S \times [N]$ to the distribution over A, where $[N] := 1, \ldots, N$. The reward of π is defined as $R(\pi) = \sum_{i=1}^{N} \sum_s P_i(s|\pi) \mathbb{E}_{a\sim\pi(s,i)} R(s, a)$ where $P_i(s|\pi)$ is the probability of arriving in s at time step i with policy π. RL aims to learn a policy π_θ, parameterized by θ, to maximize the reward.

Policy gradient is the leading class of algorithms to train policy networks. Intuitively, policy gradient optimizes the policy network by following the gradient of the expected reward regarding the policy parameters. The policy gradient is typically estimated on a finite collection of paths (i.e., trajectories) which are the past history of interactions between the policy and the environment. There are a lot of ways to estimate the policy gradient, including REINFORCE (Sutton et al., 1999), PPO (Schulman et al., 2017), DDPG (Lillicrap et al., 2015) and so on. The choice of policy gradients is independent of our method.

Generally speaking, there are two steps to estimate the policy gradient: (1) collecting a dataset of paths $D = \{\text{Path}^j\}_{j=1}^n$, where $\text{Path}^j = \{(s_i^j, a_i^j, r_i^j := R(s_i^j, a_i^j)), i = 1, \ldots, N\}$ denotes the j-th path. (2) estimating policy gradient on D. Usually, D should be sampled from the latest policy or be the collection of all past histories. For convenience, let D_i denote the data at time step i in D. As we will discuss in Section 3, this way of constructing D turned out to be less sample efficient for high-dimensional SDE policy.

As for estimating the policy gradient, we consider two popular algorithms: REINFORCE, which allows us to get unbiased estimation from samples, and DDPG, which directly

calculates policy gradient through back-propagation from the critic. Many popular algorithms are developed upon REINFORCE and DDPG. REINFORCE and DDPG calculate policy gradients as follows:

- **REINFORCE:** REINFORCE updates model parameters as:

$$\theta \leftarrow \theta + \eta \mathbb{E}_{\substack{j\sim U([n]) \\ i\sim U([N])}} \left(\nabla_\theta \log \pi_\theta(a_i^j | s_i^j, i) \sum_{i'=i}^{N} r_{i'}^j \right), \quad (2)$$

  where $U(\cdot)$ denote the uniform distribution. In practice, a critic network $Q_\phi(s_i^j, a_i^j, i)$ can be used to approximate $\sum_{i'=i}^{N} r_{i'}^j$.
- **Deep deterministic policy gradient (DDPG):** DDPG trains the actor π_θ as

$$\theta \leftarrow \theta + \eta \mathbb{E}_{\substack{j\sim U([n]) \\ i\sim U([N])}} \nabla_\theta Q_\phi(s_i^j, a, i), \quad (3)$$

  where $a \sim \pi_\theta(\cdot|s_i^j, i)$ and the gradient from a to θ is typically calculated by the reparameterization trick (Schulman et al., 2015).

Let $y_i^j = \sum_{i'=i}^{N} r_{i'}^j$, we train Q_ϕ by minimizing the loss:

$$\mathcal{L}(\phi) = \sum_{j=1}^{n} \sum_{i=1}^{N} (y_i^j - Q_\phi(s_i^j, a_i^j, i))^2 \quad (4)$$

The critic may also be trained by Bellman difference loss (Bellman, 1966).

2.2 Generative Modeling via Stochastic Differential Equations

Generative modeling aims to approximate an unknown distribution p_0 given samples. Neural networks parameterized by SDEs turned out to be a super powerful tool for generative modeling. The leading generative SDEs are known as diffusion models. The core idea behind diffusion models is that we can construct a forward process by injecting noise into samples from p_0 and directly learn the corresponding backward generative SDE by maximum likelihood methods like score matching. More specifically, for any distribution p_0, we can construct the forward process:

$$dx = f(x,t)dt + g(t)d\omega, \quad (5)$$

which induces the marginal distribution $p_t(x)$ at time t. $p_1(x)$ is the prior distribution that is easy to sample from, e.g., Gaussian. According to Anderson (1982), there is a corresponding backward process.

$$dx = (f(x,t) - g^2(t)\nabla_x \log p_t(x))dt + g(t)d\bar{\omega}. \quad (6)$$

Equations (5) and (6) share the same marginal distribution and $\nabla_x \log p_t(x_t)$ is known as the score function (Song et al., 2021b). And we can learn $\epsilon_\theta(x_t, t)$ to approximate $\nabla_x \log p_t(x_t)$ by minimizing the score-matching loss:

$$\mathcal{L}_{\text{score}}(\theta) = \mathbb{E}_{t\sim U(0,1)} \mathbb{E}_{x_0\sim q_0(x_0)} \mathbb{E}_{x_t\sim q_{t0}(x_t|x_0)} \|\epsilon_\theta(x_t,t) - \nabla_{x_t} \log p_{t0}(x_t|x_0)\|^2, \quad (7)$$

where q_t denotes the marginal distribution of estimated backward SDE induced by the model ϵ_θ in Equation (1) at time $t \in [0,1]$. And $p_{tt'}(x_t|x_{t'})$ (resp. $q_{tt'}(x_t|x_{t'})$) denotes the conditional distribution of x_t given $x_{t'}$ in forward (resp. estimated backward) SDE. In this case, $\pi_\theta(x_t,t) := f(x_t,t) - g^2(t)\epsilon_\theta(x_t,t)$.

Once ϵ_θ is fixed, generating samples can be done by using solvers (Song et al., 2021a; Lu et al., 2022). The idea of learning backward process from a given forward process is further to ODEs using flow matching (Lipman et al., 2023). Notably, the forward process will play a central role in our method.

2.3 SDE as Markov Decision Process

We can consider the SDE as an MDP with infinitely small time steps which is extended from Black et al. (2023). More specifically, in the limit $N \to \infty$, the discrete-time MDP from time 0 to N becomes a continuous MDP from time 1 to 0 [1]. We map the SDE in Equation (1) to a continuous MDP as:

$$s_t \triangleq x_t, \quad \pi_\theta(s_t,t) \triangleq f(x,t) - g^2(t)\epsilon_\theta(x_t,t),$$
$$\rho_0 \triangleq \mathcal{N}(0,I), \quad s_{t-dt} \triangleq \pi_\theta(x_t,t)dt + g(t)d\bar{\omega}$$
$$R(s_t,a_t,t) \triangleq \begin{cases} 0, & \text{if } t > 0, \\ R(s_0), & \text{if } t = 0, \end{cases}$$

where s_t is the state, $\pi_\theta(s_t,t)$ is the policy network (i.e., the actor), ρ_0 is initial state distribution, s_{t-dt} is the transition kernel, and $R(s_t,a_t,t)$ is the reward. Therefore, we may directly apply existing policy gradients to SDEs. However, as we will analyze in the next section, the naive application of policy gradients will result in an unstable training process and unsatisfactory sample complexity.

3 Challenge of applying Policy Gradient to train SDEs

In this section, we take a close look at the practical issues of directly applying policy gradient to train SDEs. These issues are caused by the fact that we estimate the policy gradient on a finite set of trajectories D, which result in ill-defined and instable policy gradients. These challenges motivate us to regularize the SDE around a perturbation forward process and exploit the perturbation nature of the forward process to stabilize the training process. For the sake of simplicity, we focus on DDPG in this section,and the analysis for the REINFORCE algorithm is similar.

3.1 Ill-defined Policy Gradient

According to chain rule, the policy gradient in Equation (3) is $\nabla_{\pi_\theta(x_t,t)} Q_\phi(x_t, \pi_\theta(x_t,t), t) \nabla_\theta \pi_\theta(x_t,t)$. Re-

[1] We let the time flow from 1 to 0 to make the notation consistent with that in diffusion models.

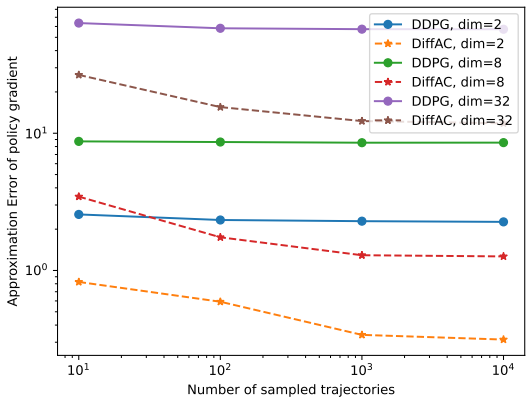

Figure 2. Comparison on the prediction error of policy gradients with respect to the number of trajectories under different settings of dimensionality. We evaluate $\mathbb{E}_{x_t}\|\nabla_{x_t}Q_\phi(x_t,\pi_\theta(x_t,t),t) - \nabla_{x_t}Q_{\phi^*}(x_t,\pi_\theta(x_t,t),t)\|$ where ϕ^* is trained on a large number of trajectories and ϕ is trained on a small number of trajectories. We can see the prediction error on policy gradient of our method is much lower than that of DDPG. Please refer to appendix for more details of this experiment.

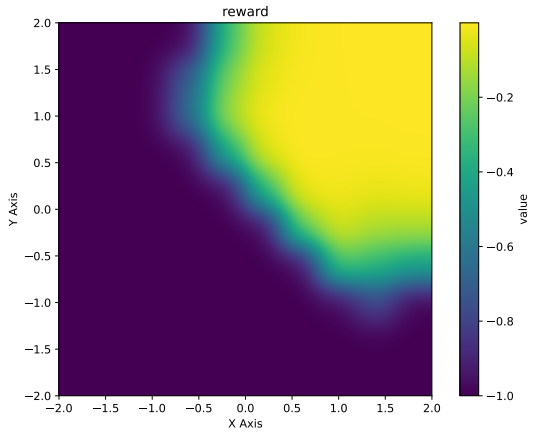

Figure 3. The reward of $\pi_\theta(x_1,1)$ in different region. The policy receives high reward in bright colored region. We can see that the policy only works well on region close to training set.

call that Q_ϕ is trained by minimizing loss in Equation (4). Since we do not have any direct training signal on $\nabla_{\pi_\theta(x_t,t)}Q_\phi(x_t,\pi_\theta(x_t,t),t)$, the gradient should be inferred from data around $\pi_\theta(x_t,t)$. Therefore, when there is insufficient data in the vicinity of $(x_t,\pi_\theta(x_t,t),t)$, it is difficult to reliably estimate the gradient $\nabla_{\pi_\theta(x_t,t)}Q_\phi(x_t,\pi_\theta(x_t,t),t)$. Hence, to obtain an accurate estimation for $\nabla_{\pi_\theta(x_t,t)}Q_\phi(x_t,\pi_\theta(x_t,t),t)$, it is essential to gather more samples around the data. As the dimensionality increases, the demand for data becomes greater, leading to unsatisfactory sample complexity.

To see this, we present a toy example to describe the relationship between the estimation error on policy gradient and the number of sampled trajectories. We compare the conventional DDPG against our method described in the next section. We can see that given the same number of sampled trajectories, our method provides a better policy gradient estimation.

3.2 Uncontrolled Behavior in Data-scarce Region

If (x_t,t) is distant from the data distribution, the behavior of $\pi_\theta(x_t,t)$ may be uncontrolled, as the loss in Equation (3) does not define the behavior of $\pi_\theta(x_t,t)$ in regions with low data density. Training π_θ solely on the limited collected paths inevitably encounter sparse data areas, and the SDE encounters these data-scarce regions. In such cases, the policy selects actions indiscriminately, consequently diminishing the effectiveness of obtaining feedback signals from the environment.

To give a better illustration, let's consider a one-step MDP in $\mathbb{R}^2$, in which the agent observes one state s, outputs an action a and the MDP terminates. We suppose the reward is $-\|s+a\|^2$. It is easy to see that the optimal policy should always output $-s$. Consider the case that we only train $\pi_\theta(s,1)$ with s such that each dimension is randomly sampled from interval $(1,2)$, and remains the policy for other regions untrained. We present the result in Figure 3. We can see that the policy only works well on the data-intensive area and poorly behaves in data-scarce areas. While this example seems to be contrived, when the policy is a high-dimensional SDE, it is inevitable that the simulation runs into data-scarce regions. Therefore, the policy might have uncontrolled behavior and further hurt the sample complexity.

4 Method

To resolve the above challenge, we introduce a novel actor-critic policy gradient algorithm which is specifically designed for SDEs. The key is to constrain the SDE to be consistent with its associated forward perturbation process. We illustrate our motivation and the key factor of our method in Figure 1. Recall that the process of estimating the policy gradient consists of two stages: (1) generating a dataset $D\sim\pi_\theta$, and (2) estimating the policy gradient based on D. We will discuss these two parts respectively.

4.1 Generating samples

As presented in Section 3, the challenges stem from the fact that the policy gradient is estimated using a finite set $D\sim\pi_\theta$ of sampled paths. Accurate estimation of the policy gradient occurs only when D is relatively large, leading to unsatisfactory sample complexity. Our key observation is that by ensuring the SDE aligns with its associated perturbation process, the policy gradient estimation can be made more robust.

Definition 4.1 (Consistent SDE)**.** An SDE is the associated

Algorithm 1 DiffAC-v1

Input: Initialized actor ϵ_θ and critic V_ϕ, reward function $R(\cdot)$, $D_0 = \emptyset$
Output: θ, ϕ

```
for each iteration do
    Sample D_0 ← {x_0^j}_{j=1}^n from Equation (1)
    Train ε_θ by minimizing the score matching loss
    Train V_φ according to Equation (9)
    for each iteration do
        Calculate policy gradient PG(θ) as Equation (10) or Equation (11)
        θ ← θ + ηPG(θ)
    end for
end for
```

forward perturbation process for the backward SDE in Equation (1) if and only if $q_0 = p_0$. Furthermore, if $q_t = p_t$ for all $t \in [0, 1]$, then the forward SDE and backward SDE are called consistent.

It is noteworthy that a backward SDE is consistent with its associated forward SDE if and only if the score loss in Equation (7) is minimized at ϵ_θ. More importantly, when the backward SDE is consistent, we have a more robust and efficient way to sample from q_t and further estimate the policy gradient more accurately.

Lemma 4.2. *If the SDE defined by ϵ_θ is consistent, let $x_t \sim p_{t0}(x_t|x_0)$ where $x_0 \sim q_0(x_0)$. Then, we have $x_t \sim q_t(x_t)$.*

Proof. The proof is straightforward. If the SDE is consistent, we have

$$\begin{aligned}&\int_{x_0} q_0(x_0)p_{t0}(x_t|x_0)dx_0\\ =&\int_{x_0} p_0(x_0)p_{t0}(x_t|x_0)dx_0\\ =&\int_{x_0} p(x_t, x_0)dx_0 = p_t(x_t) = q_t(x_t).\end{aligned}$$

□

Generating $\tilde{D}_t$ for policy gradient estimation by perturbing D_0: Considering the initial dataset $D_0 = \{x_0^j \sim q_0(x_0)\}_{j=1,\dots,n}$, it is feasible to directly produce an arbitrarily large set $\tilde{D}_t = \{x_t^j\}_{j=1,\dots,N}$, where $x_t^j \sim p_{t0}(x_t|x_0)$ for a given $x_0 \in D_0$. Lemma 4.2 demonstrates that $x_t^j \sim q_t(x_t)$, thereby implying that it is possible to estimate the policy gradient on samples perturbed from D_0.

Intuitively, the construction of $\tilde{D}_t$ mitigates the practical issues in Section 3 as follows: Firstly, it is evident that the perturbation process encompasses the entire space, resulting in a well-defined policy gradient. Although the samples from $\tilde{D}_t$ are not entirely independent, Figure 2 indicates that policy gradients estimated on $\tilde{D}_t$ exhibit accuracy when n is comparatively small. Secondly, for consistent SDEs, the distribution of training data is guaranteed to have the same distribution. Therefore, the probability for a consistent SDE to run into data-scarce region is relatively small.

4.2 Estimation of policy gradient

Given the process of generating samples mentioned above, it is easy to extend the policy gradient in Equation (2) and Equation (3). We consider the actor-critic framework, where a critic is trained to predict the cumulative reward and the actor is trained based on the reward signal provided by the critic. We observe that with SDE policy, it is more convenient to train the critic $V_\phi(x_t, t)$ to predict the reward given x_t rather than $Q_\phi(x_t, a_t, t)$. Hence, we will focus on the training of V_ϕ instead. The combination of the actor and critic training procedures, as well as the score-matching loss and the construction of $\tilde{D}_t$ in Section 4.1, results in our first actor-critic algorithm presented in Algorithm 1. Since the forward perturbation process draws inspiration from diffusion models, we name our algorithm Diffusion Actor-Critic (DiffAC).

Critic Training: For any consistent SDE ϵ_θ, $V_{\phi^*}(x_t, t) = \mathbb{E}_{x_0\sim q_{0t}(x_0|x_t)}R(x_0)$ if and only if

$$\phi^* = \arg\min_\phi \mathbb{E}_{\substack{t\sim U(0,1)\\ x_0\sim q_0(x_0)\\ x_t\sim p_{t0}(x_t|x_0)}} (V_\phi(x_t, t) - R(x_0))^2. \quad (8)$$

The derivation of Equation (8) is straightforward. Therefore, to train the critic, we just need to perturb D_0 to get $\tilde{D}_t$ and minimize the loss in Equation (8), leading to:

$$\text{(Critic loss):} \qquad \frac{1}{n}\sum_{j=1}^{n}(V_\phi(x_t^j, t) - R(x_0^j))^2, \quad (9)$$

where $x_t^j \sim p_{t0}(\cdot|x_0^j)$, $x_0^j \in D_0$, and $R(x_0^j)$ is the reward of x_0^j.

Actor Training: We now proceed to extend the REINFORCE and DDPG algorithms to SDEs. Consider the sample $\bar{x}_t^j$, which is generated by simulating Equation (1) for an infinitesimal time step dt starting from x_t^j, and can be represented as $\bar{x}_{t-dt}^j \sim \pi_\theta(x_t^j, t)dt + \sigma_t d\bar{\omega}$. Furthermore, let $\mathcal{P}_\theta(\bar{x}_{t-dt}|x_t)$ denote the density of this sample. We have:

Theorem 4.3. *For SDE policy, we can estimate the policy gradient as:*

SDE-REINFORCE :

$$\mathcal{PG}(\theta) \leftarrow \frac{1}{n}\sum_{j=1}^{n} \nabla_\theta \log \mathcal{P}_\theta(\bar{x}_{t-dt}^j|x_t^j)V_\phi(\bar{x}_{t-dt}^j, t\text{–}dt), \quad (10)$$

SDE-DDPG :

$$\mathcal{PG}(\theta) \leftarrow \frac{1}{n}\sum_{j=1}^{n} \nabla_\theta V_\phi(\bar{x}_t^j, t - dt), \quad (11)$$

where x_0^j is sampled from D_0, $x_t^j \sim p_{t0}(x_t^j|x_0^j)$, and $\bar{x}_{t-dt}^j \sim \mathcal{P}_\theta(\bar{x}_{t-dt}^j|x_t^j)$.

Algorithm 2 DiffAC-v2

Input: Initialized $\epsilon_\theta, \epsilon_{\theta'}$ and critic V_ϕ, reward function $R(\cdot)$, $D_0 = \emptyset$
Output: θ

1: **for** each iteration **do**
2: Sample $\{x_0^j\}_{j=1}^n$ by simulating $dx_t = \pi_\theta(x_t, t)dt + \sigma_t d\bar{\omega}$
3: Sample $D_0 \leftarrow \{x_0^j\}_{j=1}^n$ from Equation (1)
4: Train V_ϕ according to Equation (9)
5: Train $\epsilon_{\theta'}$ by score-matching in Equation (7)
6: **for** each iteration **do**
7: Update θ according to Equation (10) or Equation (11) along with Equation (13)
8: **end for**
9: **end for**

In practice, the infinitesimal time step can be replaced by discrete time steps in solvers. We suppose that an SDE solver simulates Equation (1) from a iterative procedure: $x_{\tau-1} = \alpha_\tau(x_\tau - \beta_\tau \pi_\theta(x_\tau, t_\tau)) + \zeta_\tau z_\tau$ where $z_\tau \sim \mathcal{N}(0, I)$, $\tau = 1, \ldots, T$, $t_{\tau-1} < t_\tau$ with $t_0 = 0, t_T = 0$, $x_1 \sim \mathcal{N}(0, I)$, $\alpha_\tau, \beta_\tau, \zeta_\tau$ are specified by solvers. Many popular solvers for diffusion models fall into this formulation, e.g., DDIM (Song et al., 2021a) and DDPM (Ho et al., 2020). Similarly, we can replace t, x_t^j, $t - dt$, and $\bar{x}_{t-dt}^j$ in Equations (10) and (11) with t_τ, x_τ, $t_{\tau-1}$, and $x_{t_{\tau-1}}$, respectively.

4.3 A Practical Implementation

In Algorithm 1, we first run score-matching to make sure ϵ_θ is consistent, and then apply policy gradient to optimize the reward. However, during the training step, the model may rapidly forget the knowledge learned during score-matching, which leads to frustrating inconsistency. Therefore, we introduce an additional policy which is trained to maximize reward under the regularization of score-matching policy to alleviate this inconsistency. The regularization is:

$$\mathbb{D}_{\mathrm{KL}}(\epsilon_\theta, \epsilon_{\theta'}, D) = \frac{1}{|D|} \sum_{x_t \in D} \mathbb{D}_{\mathrm{KL}}(\pi_\theta(x_t, t), \pi_{\theta'}(x_t, t)). \quad (12)$$

And we update our policy ϵ_θ as

$$\theta \leftarrow \theta + \eta_1 \mathcal{PG}(\theta) - \eta_1 \eta_2 \nabla_\theta \mathbb{D}_{\mathrm{KL}}(\epsilon_\theta, \epsilon_{\theta'}, D), \quad (13)$$

where $\epsilon_{\theta'}$ is trained on D_0 via score matching, η_1 is the learning rate, and η_2 is the regularization coefficient. Equation (13) leads to Algorithm 2. Moreover, we show that the objective in Algorithm 2 also leads to the optimal policy.

Lemma 4.4. *Let x_0^* denote the optimal point, that is, for any x_0', $R(x_0^*) \geq R(x_0')$. Let ϵ_θ^* denote the consistent SDE with $q_0 = \delta(x_0^*)$. Then, ϵ_θ^* is the minimizer of loss in Equation (13).*

Proof. The proof is straight-forward as ϵ_θ^* is the minimizer for both terms in Equation (13). □

In practice, with the regularization, the consistency is usually approximately but not exactly ensured. Thus we provide a upper bound of the approximation error of the policy gradient estimation as the theoretical justification of our practical method.

Theorem 4.5. *Let $\mathcal{PG}(\theta)$ be the unbiased policy gradient estimated with only SDE policy q parameterized by ϵ_θ, and $\widetilde{\mathcal{PG}}(\theta)$ be the approximated policy gradient estimated with the perturbation process p whose consistent backward SDE $\tilde{q}$ is parameterized by $\epsilon_{\theta'}$. And the SDEs are discretized into T time steps, i.e., $\{t_\tau\}_{\tau=0}^T$ with $\tau_0 = 0$ and $\tau_T = 1$. Let $M_{t_\tau} := \|\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau}) R(x_0)\|_\infty$, we have*

$$\|\widetilde{\mathcal{PG}}(\theta) - \mathcal{PG}(\theta)\| \leq \sum_{\tau=1}^{T} M_{t_\tau} \cdot \sqrt{2 \ln 2 \sum_{\nu=0, \nu \neq \tau-1}^{T-1} \mathbb{D}_{KL}\big(\tilde{q}(x_{t_\nu}|x_{t_{\nu+1}}) \big\| q(x_{t_\nu}|x_{t_{\nu+1}})\big)}$$

5 Related Work

Diffusion Models and Forward / Backward SDEs Diffusion models (Song & Ermon, 2019; Song et al., 2021b; Ho et al., 2020; Yang et al., 2022) have emerged as powerful tools in the field of generative models. Their primary objective is to maximize the likelihood of data distribution. To achieve this, Song et al. (2021b); Ho et al. (2020) construct forward stochastic differential equations (SDEs) by injecting noise and utilize their corresponding backward SDEs for generating samples. These backward SDEs can be efficiently trained using denoising score-matching (Vincent, 2011). SDEs exhibit a remarkable capability for modeling complex distributions and generating intricate images (Rombach et al., 2022) and molecules (Guan et al., 2023). This potential of SDEs inspires us to explore their use as the foundation for policy networks.

RL with Diffusion Models Recently progress in RL has identified diffusion models as a powerful tool in policy modeling, due to its generative capability and standardized training process. In offline RL where we need to learn a policy from a given dataset, researchers exploit diffusion models to deal with heterogeneous datasets, generating in-distribution strategies or modeling complex strategy distribution (Janner et al., 2022; Wang et al., 2023; Ajay et al., 2023; Hansen-Estruch et al., 2023; Lu et al., 2023). In online RL, where we need to interact with an environment, Chen et al. (2023) uses diffusion models to model multi-modal distribution for exploration. Black et al. (2023); Fan et al. (2023) proposed to fine-tune pretrained diffusion models with REINFORCE, but they didn't address the stability issue.

Structure-based Drug Design Structure-based drug design (SBDD, Anderson, 2003). aims to generate ligand molecules given a protein binding site (i.e., protein pocket),

Table 1. Summary of Vina Score of ligand molecules generated by all baselines and our method. Note that a smaller score is better. All online algorithms are tested using the best checkpoint (denoted as Best Run) and the last checkpoint (denoted as Last Run), respectively. The improvements (resp. deteriorations) compared with TargetDiff are higlighed in green (resp. red). The standard deviation is highlighted in blue.

Method	Best Run		Last Run	
	Avg.	Med.	Avg.	Meg.
Reference	-6.36	-6.46	-	-
AR	-5.75 ±1.39	-5.64	-	-
Pocket2Mol	-5.14 ±1.60	-4.70	-	-
TargetDiff	-5.45 ±2.46	-6.30	-	-
EEGSDE-0.001	-5.66 ±2.78 (-0.20)	-6.51 (-0.21)	-	-
EEGSDE-0.01	-6.40 ±2.61 (-0.95)	-7.05 (-0.75)	-	-
EEGSDE-0.1	-6.53 ±3.08 (-1.08)	-7.35 (-1.05)	-	-
EEGSDE-1	-3.30 ±1.59 (+2.15)	-4.67 (+1.63)	-	-
Online EEGSDE-0.001	-7.17 ±1.86 (-1.72)	-7.16 (-0.86)	-6.50 ±2.47(-1.05)	-6.61 (-0.31)
Online EEGSDE-0.01	-8.22 ±1.89 (-2.77)	-8.06 (-1.76)	-7.56 ±2.46 (-2.11)	-7.51 (-1.21)
Online EEGSDE-0.1	-8.58 ±1.70 (-3.13)	-8.52 (-2.22)	-7.78 ±2.33 (-2.33)	-7.78 (-1.48)
Online EEGSDE-1	-7.13 ±1.08 (-1.68)	-7.28 (-0.98)	-2.13 ±2.10 (+3.32)	-4.29 (+2.01)
DiffAC	**-9.07** ±1.99 **(-3.62)**	**-9.04 (-2.74)**	**-8.50** ±2.11 **(-3.05)**	**-8.38 (-2.08)**

which is a key tool in drug discovery. The ligand molecules are usually expected to have desired properties, such as high binding affinity to the target protein. Luo et al. (2021); Liu et al. (2022); Peng et al. (2022) proposed to generate atoms (and bonds) of 3D ligands based on 3D protein pockets in an auto-regressive way. More recently, Guan et al. (2023); Lin et al. (2022); Schneuing et al. (2022) employed SE(3)-equivariant diffusion models for SBDD. To design molecules with desired properties (i.e., inverse design), Bao et al. (2023) proposed equivariant energy-guided stochastic differential equations (EEGSDE). We test our method on SBDD and achieve superior performance than EEGSDE and its stronger variants.

6 Experiments

We demonstrate the effectiveness of our methods on structure-based drug design (SBDD, Anderson, 2003). Here we apply our methods to promote the binding affinity of ligand molecules generated by diffusion models. We also provide experiment results in text-to-image generation task, which shows the generalizability of our method. Please see Appendix G for details.

6.1 Experimental Setup

Dataset Following the previous work (Luo et al., 2021; Peng et al., 2022; Guan et al., 2023), we use the CrossDocked2020 dataset (Francoeur et al., 2020) for both training and optimization. We follow the same dataset preprocessing and splitting procedure as Luo et al. (2021). $100,000$ pocket-ligand pairs are used for training, and 100 pockets are used for testing. The goal is to generate ligands with high binding affinity towards the pockets in the test set.

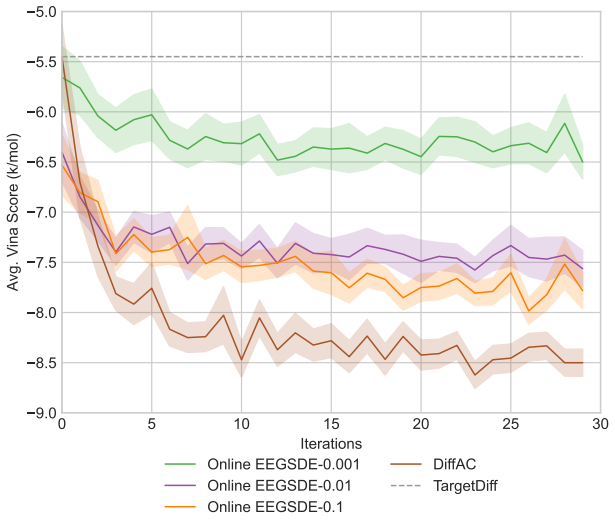

Figure 4. Optimization curves which show how average Vina Score of generated ligand molecules changes over optimization iterations.

Baselines We implement and evaluate some baselines and DiffAC: (1) AR (Luo et al., 2021). (2) Pocket2Mol (Peng et al., 2022). (3) TargetDiff (Guan et al., 2023). (4) EEGSDE (Bao et al., 2023). We implement the equivariant energy guidance on TargetDiff. The energy function is the same as the pre-trained critic that we have mentioned above. $\{0.001, 0.01, 0.1, 1\}$ are used as coefficients of energy guidance during sampling. (5) Online EEGSDE. To better leverage interactions with the environment, we online train the energy function as we do for DiffAC. Here we also follow the setting about the number of optimization itera-

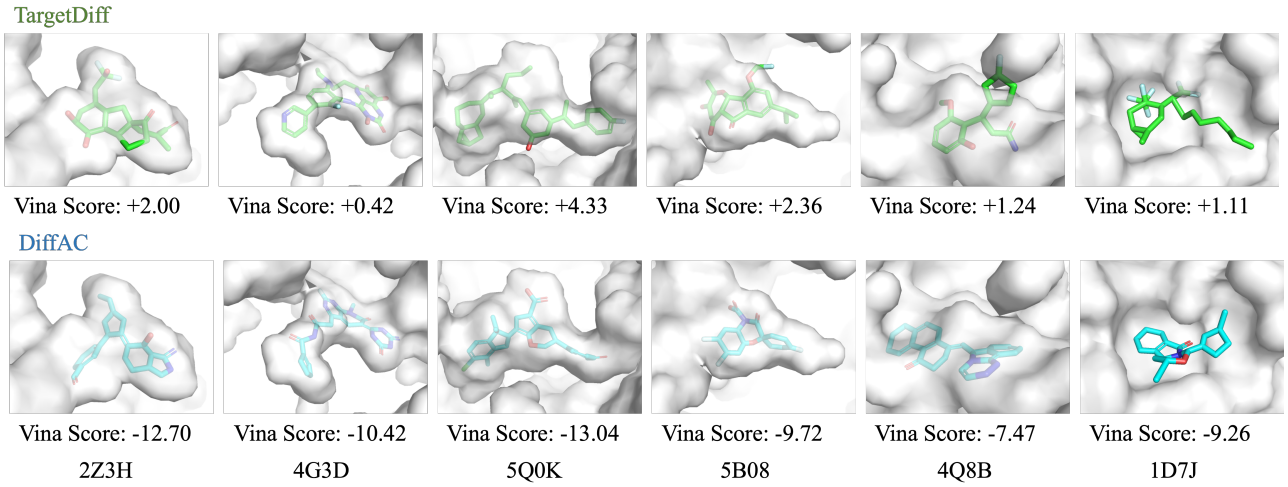

Figure 5. Examples of generated ligands. Carbon atoms in ligand molecules by TargetDiff (Guan et al., 2023) and DiffAC are visualized in green and cyan, respectively. Here we select some cases where TargetDiff easily generates unrealistic ligand molecules that clash with protein surfaces physically which usually leads to extremely bad Vina scores. DiffAC can sample realistic ligand molecules with high quality in these hard cases.

tions and the number of generated samples and updates of the critic in each iteration. Similarly, $\{0.001, 0.01, 0.1, 1\}$ are also used was guidance coefficients. (5) We implement DiffAC based in the framework of TargetDiff. The hyperparameters in our method are the same for all target pockets. Refer to Appendix D for more implementation details. We have not reported the results of DDPO (Black et al., 2023) and DPOK (Fan et al., 2023) as they do not have satisfactory performances in our evaluation.

Evaluation Designing molecules with desired properties (i.e., molecular inverse design) is a fundamental and valuable task. Here we choose binding affinity as our target due to its importance in structure-based drug design. We employ AutoDock Vina (Eberhardt et al., 2021) to estimate the binding affinity of pairs of the protein pockets and the generated ligands, following the same setup as Luo et al. (2021); Guan et al. (2023). Optimizing Vina Score is a challenging task because not only the molecules themselves but also their chemical and spatial interaction with the 3D protein pockets need to be considered. We generate 100 ligand molecules across 100 pockets in the test set using TargetDiff and EEGSDE. For online EEGSDE and DiffAC, we first finish the online fine-tuning process on each pocket separately and use the best and the last checkpoint to generated 100 ligand molecules, respectively. For all methods, we collect all generated molecules across 100 test proteins and report the mean and median of Vina Score.

6.2 Main Results

We evaluate all baselines and our method under the setting introduced in Section 6.1. As Table 1 shows, our method performs better than all other baselines. EEGSDE with proper energy guidance coefficient can indeed improve the property of generated molecules. And the online variants of EEGSDE can further improve the performance, which shows the benefits of online training the critic. Our method can even outperform online EEGSDE with the best energy guidance coefient by a large margin and achieve the best Avg. Vina Score over all methods for structure-based drug design, which demonstrates the effectiveness of DiffAC. We visualize examples of ligand molecules generated by TargetDiff and DiffAC on some hard cases in Figure 5. As Figure 4 shows, DiffAC converges faster than all other online optimization algorithms, which demonstrates the superiority of our method in terms of sample complexity and training efficiency. We also provide the optimization curves of each protein pocket in Appendix E. The experiments has revealed the great potential of our method in important real-world applications, such as drug discovery.

Besides, we conduct ablation studies on the effects of different regularization coefficients on optimization. Please refer to Appendix F for the details. The results show that a proper regularization coefficient indicates both approximated consistency and ample room for optimization. The experiment also demonstrates that even if the consistency is not strictly ensured, the approximated policy gradient can still effectively optimize the reward in practice, which aligns with our theoretical result Theorem 4.5.

7 Conclusions

This paper proposes DiffAC, a stabilized policy gradient method for SDEs, and demonstrate its superiority on structure-based drug design. This is a general framework with great potential. In terms of future work, it would be

interesting to apply this method to many valuable applications where user preferences or design requirements can be specified, such as protein design, chip design, etc.

Acknowledgements

We thank Huayu Chen, Min Zhao, Cheng Lu, and Chongxuan Li for useful discussions.

Impact Statement

This paper presents work which proposes improved techniques for training SDEs with policy gradients and demonstrates the effectiveness of the proposed method in the structure-based drug design task. Our methodology can be adapted in other design scenarios and should not be used to design products that are harmful to society.

References

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J. B., Jaakkola, T. S., and Agrawal, P. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

Anderson, A. C. The process of structure-based drug design. *Chemistry & biology*, 10(9):787–797, 2003.

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

Bao, F., Zhao, M., Hao, Z., Li, P., Li, C., and Zhu, J. Equivariant energy-guided SDE for inverse molecular design. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=r0otLtOwYW.

Bellman, R. Dynamic programming. *Science*, 153(3731): 34–37, 1966.

Black, K., Janner, M., Du, Y., Kostrikov, I., and Levine, S. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

Chen, H., Lu, C., Ying, C., Su, H., and Zhu, J. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=42zs3qa2kpy.

Du, X., Li, Y., Xia, Y.-L., Ai, S.-M., Liang, J., Sang, P., Ji, X.-L., and Liu, S.-Q. Insights into protein–ligand interactions: mechanisms, models, and methods. *International journal of molecular sciences*, 17(2):144, 2016.

Eberhardt, J., Santos-Martins, D., Tillack, A. F., and Forli, S. Autodock vina 1.2. 0: New docking methods, expanded force field, and python bindings. *Journal of chemical information and modeling*, 61(8):3891–3898, 2021.

Fan, Y., Watkins, O., Du, Y., Liu, H., Ryu, M., Boutilier, C., Abbeel, P., Ghavamzadeh, M., Lee, K., and Lee, K. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *arXiv preprint arXiv:2305.16381*, 2023.

Francoeur, P. G., Masuda, T., Sunseri, J., Jia, A., Iovanisci, R. B., Snyder, I., and Koes, D. R. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of chemical information and modeling*, 60(9):4200–4215, 2020.

Guan, J., Qian, W. W., Peng, X., Su, Y., Peng, J., and Ma, J. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *International Conference on Learning Representations*, 2023.

Hansen-Estruch, P., Kostrikov, I., Janner, M., Kuba, J. G., and Levine, S. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

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

Li, J., Li, D., Xiong, C., and Hoi, S. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pp. 12888–12900. PMLR, 2022.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lin, H., Huang, Y., Liu, M., Li, X., Ji, S., and Li, S. Z. Diffbp: Generative diffusion of 3d molecules for target protein binding. *arXiv preprint arXiv:2211.11214*, 2022.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The*

Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Liu, M., Luo, Y., Uchino, K., Maruhashi, K., and Ji, S. Generating 3d molecules for target protein binding. In *International Conference on Machine Learning*, 2022.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

Lu, C., Chen, H., Chen, J., Su, H., Li, C., and Zhu, J. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. *arXiv preprint arXiv:2304.12824*, 2023.

Luo, S., Guan, J., Ma, J., and Peng, J. A 3d generative model for structure-based drug design. *Advances in Neural Information Processing Systems*, 34:6229–6239, 2021.

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., and Hutchison, G. R. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 3(1): 1–14, 2011.

Peng, X., Luo, S., Guan, J., Xie, Q., Peng, J., and Ma, J. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. In *International Conference on Machine Learning*, pp. 17644–17655. PMLR, 2022.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Schneuing, A., Du, Y., Harris, C., Jamasb, A., Igashov, I., Du, W., Blundell, T., Lió, P., Gomes, C., Welling, M., et al. Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695*, 2022.

Schuhmann, C., Vencu, R., Beaumont, R., Kaczmarczyk, R., Mullis, C., Katta, A., Coombes, T., Jitsev, J., and Komatsuzaki, A. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., 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, 2022.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

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

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., and Dong, Y. Imagereward: Learning and evaluating human preferences for text-to-image generation. *arXiv preprint arXiv:2304.05977*, 2023.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Shao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

A Extended Preliminaries

We here provide a more comprehensive background of SDEs as follows:

- (See Equation (5)) $dx = f(x,t)dt + g(t)d\omega$, which is the forward SDE that corresponds to the perturbation process (i.e., the forward process of diffusion models). $d\omega$ is a Wiener process. $f(\cdot,t) : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function called the drift coefficient of $x(t)$. $g(t)$ is a scalar function of time and known as diffusion coefficient of the underlying dynamic.
- Following Song et al. (2021b), applying Kolmogorov's forward equation (Fokker-Planck equation) to an SDE, we can derive a probability ODE which describes how the marginal distribution $p_t(x)$ evolves through time t.
- (See Equation (6)) According to Anderson (1982), an SDE as Equation (5) has a backward SDE: $dx = (f(x,t) - g^2(t)\nabla_x \log p_t(x))dt + g(t)d\bar{\omega}$, which shares the same marginal distribution $p_t(x)$ at time t. $d\bar{\omega}$ is the reverse Wiener process.
- (See Equation (1)) $dx_t = \pi_\theta(x_t,\theta)dt + g(t)d\bar{\omega}$, which is the approximated backward SDE parameterized by θ. Conventionally, we use $\epsilon_\theta(x_t,t)$ to approximate the score $\nabla_x \log p_t(x)$. So we can let $\pi_\theta(x_t,\theta) := f(x_t,t) - g^2(t)\epsilon_\theta(x_t,\theta)$. The approximated backward SDE corresponds to the generative process of diffusion models. Specifically, x_T is sampled from the prior distribution, evolves following this SDE, and arrives at x_0. x_0 is the generated sample.
- p_0 and q_0 in Definition 4.1. Specifically, we denote the marginal distribution at time t of Equations (1) and (5) as p_t and q_t, respectively. Thus, p_0 (resp. q_0) is p_t (resp. q_t) at time $t = 0$.

Then we provide the discrete-time version of SDE-based generative models, also known as diffusion models. The following introduction follows DDPM (Ho et al., 2020). Let $q_0(x_0)$ denote an arbitrary distribution, we have

- The forward process (i.e., the perturbation process): $p(x_{1:T}|x_0) = \prod_{t=1,\dots,T} p(x_t|x_{t-1})$ where $p(x_t|x_{t-1}) = \mathcal{N}(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ is a Gaussian distribution. β_t are pre-defined parameters which is linearly scheduled from $\beta_1 = 1 \times 10^{-4}$ to $\beta_T = 0.02$ in DDPM. It is noteworthy that there is no trainable parameters in the forward process and for sufficiently large T, $p(x_T|x_0) \approx \mathcal{N}(0, I)$ the standard Gaussian distribution. Moreover, it is obvious that $p(x_t|x_0)$ is also a closed-form Gaussian process. In DiffAC, we exploited the forward process to stabilize the policy-gradient estimation.
- The parameterized backward process: $q_\theta(x_{0:T}) = q_T(x_T)\prod_{t=T,\dots,1} q_\theta(x_{t-1}|x_t)$ where $q_T(x_T) = \mathcal{N}(0, I)$ is the standard Gaussian distribution. And $q_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}|\mu_\theta(x_t,t), \sigma_\theta(x_t,t))$. Generally speaking, the score matching loss is to minimize KL divergence between the marginal distributions of the forward and the backward process.
- Consistency: once the backward process is fixed, we can define an associated forward process by letting $p_0(x_0) := q_0(x_0) = \mathbb{E}_{x_T} q_\theta(x_0|x_T)$. A backward process is called consistency if the associated forward process has the same marginal distribution with the backward process at every time step $t = 0, \dots, T$. And the backward process is consistent if and only if the score matching loss is minimized.

B Toy Example in Section 3.1

We compare the error on $\nabla_{\pi_\theta(x_t,t)} Q_\phi(x_t, \pi_\theta(x_t,t), t)$. Let ϕ' denote the critic trained by Equation (4) on n trajectories and ϕ'' denote the trained by Equation (8) on D_0 with $|D_0| = n$. We evaluate $|\nabla_{\pi_\theta(x_t,t)} Q_{\phi'}(x_t, \pi_\theta(x_t,t), t) - \nabla_{\pi_\theta(x_t,t)} Q_{\phi'^*}(x_t, \pi_\theta(x_t,t), t)|$ and $|\nabla_{\pi_\theta(x_t,t)} Q_{\phi''}(x_t, \pi_\theta(x_t,t), t) - \nabla_{\pi_\theta(x_t,t)} Q_{\phi''^*}(x_t, \pi_\theta(x_t,t), t)|$ where ϕ'^* and ϕ''^* are trained on 10^5 trajectories.

The architecture of the critic network is 3 layered-MLP with hidden dimemsion 256. We train each network for 10^5 iteration with batchsize 256. And A For reward function, we use Rastrigin function which is a toy function, with many local minimas, designed for testing zero order optimization algorithm.

C Proofs

Theorem C.1. *Let $\mathcal{PG}(\theta)$ be the unbiased policy gradient estimated with only SDE policy q parameterized by ϵ_θ, and $\widetilde{\mathcal{PG}}(\theta)$ be the approximated policy gradient estimated with the perturbation process p whose consistent backward SDE $\tilde{q}$ is parameterized by $\epsilon_{\theta'}$. And the SDEs are discretized into T time steps, i.e., $\{t_\tau\}_{\tau=0}^T$ with $\tau_0 = 0$ and $\tau_T = 1$. Let $M_{t_\tau} := \|\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})R(x_0)\|_\infty$, we have*

$$\|\widetilde{\mathcal{PG}}(\theta) - \mathcal{PG}(\theta)\| \le \sum_{\tau=1}^{T} M_{t_\tau} \sqrt{2\ln 2 \sum_{\nu=0,\nu\neq\tau-1}^{T-1} \mathbb{D}_{KL}\big(\tilde{q}(x_{t_\nu}|x_{t_{\nu+1}}) \big\| q(x_{t_\nu}|x_{t_{\nu+1}})\big)}$$

Proof. The policy gradient estimated with the exact SDE policy q can be derived as follows:

$$\begin{aligned}
\mathcal{PG}(\theta) =&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_1\sim q(x_1),x_{t_\tau}\sim q(x_{t_\tau}|x_1)}\mathbb{E}_{x_{t-dt}\sim q(x_{t_{\tau-1}}|x_t)}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})V_{\phi^*}(x_{t_{\tau-1}},t_\tau-1)\\
=&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_1\sim q(x_1),x_{t_\tau}\sim q(x_{t_\tau}|x_1)}\mathbb{E}_{x_{t_{\tau-1}}\sim q(x_{t_{\tau-1}}|x_{t_\tau})}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})\mathbb{E}_{x_0\sim q(x_0|x_{t_{\tau-1}})}R(x_0)\\
=&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_1\sim q(x_1),x_{t_\tau}\sim q(x_{t_\tau}|x_1),x_{t_{\tau-1}}\sim q(x_{\tau-1}|x_{t_\tau}),x_0\sim q(x_0|x_{t_{\tau-1}})}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})R(x_0)\\
=&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_{t_\tau}\sim q(x_{t_\tau}),x_{t_{\tau-1}}\sim q(x_{t_{\tau-1}}|x_{t_\tau}),x_0\sim q(x_0|x_{t_{\tau-1}})}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})R(x_0)
\end{aligned}$$

Note that the SDE policy q may not be consistent, i.e., there may not exist a forward SDE with the the simple formula as in Equation (5) that induces the same marginal distribution with the SDE policy.

As for the approximated policy gradient estimated with the perturbation process q, the only difference from the above estimation is that we sample x_{t_τ} by first sampling x_0' by the SDE policy followed by perturbation, i.e., $x_0'\sim q(x_0')$ and $x_{t_\tau}\sim p(x_{t_{\tau-1}}|x_0')$. Given $q(x_0')$ and its corresponding perturbation process, we can always derive a backward SDE $\tilde{q}$ that is consistent with the perturbation process via score matching, which is the backward SDE parameterized by the reference model $\epsilon_{\theta'}$. And we have $\int_{x_0'} q(x_0')p(x_t|x_0')\,dx_0=\int_{x_1}q(x_1)\tilde{q}(x_t|x_1)\,dx_1,\forall t$. Note that $q(x_1)=\mathcal{N}(0,I)$. Therefore, the policy gradient estimated with the perturbation process p can be derived as follows:

$$\begin{aligned}
\widetilde{\mathcal{PG}}(\theta) =&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_0'\sim q(x_0'),x_{t_\tau}\sim p(x_{t_\tau}|x_0')}\mathbb{E}_{x_{t_{\tau-1}}\sim q(x_{t_{\tau-1}}|x_{t_\tau})}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})\widetilde{V}_{\phi^*}(x_{t_{\tau-1}},t_\tau-1)\\
=&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_1\sim q(x_1),x_{t_\tau}\sim \tilde{q}(x_{t_\tau}|x_1)}\mathbb{E}_{x_{t_{\tau-1}}\sim q(x_{t_{\tau-1}}|x_{t_\tau})}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})\widetilde{V}_{\phi^*}(x_{t_{\tau-1}},t_\tau-1)\\
=&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_1\sim q(x_1),x_{t_\tau}\sim \tilde{q}(x_{t_\tau}|x_1)}\mathbb{E}_{x_{t_{\tau-1}}\sim q(x_{t_{\tau-1}}|x_{t_\tau})}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})\mathbb{E}_{x_0\sim \tilde{q}(x_0|x_{t_{\tau-1}})}R(x_0)\\
=&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_1\sim q(x_1),x_{t_\tau}\sim \tilde{q}(x_{t_\tau}|x_1),x_{t_{\tau-1}}\sim q(x_{t_{\tau-1}}|x_{t_\tau}),x_0\sim \tilde{q}(x_0|x_{t_{\tau-1}})}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})R(x_0)\\
=&\mathbb{E}_{\tau\sim U([N])}\mathbb{E}_{x_{t_\tau}\sim \tilde{q}(x_{t_\tau}),x_{t_{\tau-1}}\sim q(x_{t_{\tau-1}}|x_{t_\tau}),x_0\sim \tilde{q}(x_0|x_{t_{\tau-1}})}\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})R(x_0)
\end{aligned}$$

Note that we have Let $M_{t_\tau}=\|\nabla_\theta \log q(x_{t_{\tau-1}}|x_{t_\tau})R(x_0)\|_\infty$. Let $Q_{t_\tau}(x_{t_\tau},x_{t_{\tau-1}},x_0):=q(x_{t_\tau})q(x_{t_{\tau-1}}|x_{t_\tau})q(x_0|x_{t_{\tau-1}})$ and $\widetilde{Q}_{t_\tau}(x_{t_\tau},x_{t_{\tau-1}},x_0):=\tilde{q}(x_{t_\tau})q(x_{t_{\tau-1}}|x_{t_\tau})\tilde{q}(x_0|x_{t_{\tau-1}})$. With Pinsker's inequality, We have:

$$\|\widetilde{\mathcal{PG}}(\theta)-\mathcal{PG}(\theta)\|\le\sum_{\tau=1}^{T}M_{t_\tau}\|\widetilde{Q}_{t_\tau}-Q_{t_\tau}\|_1\le\sum_{\tau=1}^{T}M_{t_\tau}\sqrt{2\ln 2\mathbb{D}_{\mathrm{KL}}(\widetilde{Q}_{t_\tau}\|Q_{t_\tau})}$$

With the chain rule of KL-Divergence, we have

$$\begin{aligned}
\mathbb{D}_{\mathrm{KL}}(\widetilde{Q}_{t_\tau}\|Q_{t_\tau}) =&\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\tau})q(x_{t_{\tau-1}}|x_{t_\tau})\tilde{q}(x_0|x_{t_{\tau-1}})\big\|q(x_{t_\tau})q(x_{t_{\tau-1}}|x_{t_\tau})q(x_0|x_{t_{\tau-1}})\big)\\
\le&\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\tau})\big\|q(x_{t_\tau})\big)+\mathbb{D}_{\mathrm{KL}}\big(q(x_{t_{\tau-1}}\big|x_{t_\tau})\|q(x_{t_{\tau-1}}|x_{t_\tau})\big)+\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_0|x_{t_{\tau-1}})\big\|q(x_0|x_{t_{\tau-1}})\big)\\
=&\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\tau})\big\|q(x_{t_\tau})\big)+\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_0|x_{t_{\tau-1}})\big\|q(x_0|x_{t_{\tau-1}})\big)
\end{aligned}$$

By introducing the latent variables and with the Markov property of SDE $\tilde{q}$ and the chain rule of KL divergence, we have

$$\begin{aligned}
\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\tau})\big\|q(x_{t_\tau})\big)\le&\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\tau},x_{t_{\tau+1}},\dots,x_{t_{T-1}},x_1)\big\|q(x_{t_\tau},x_{t_{\tau+1}},\dots,x_{t_{T-1}},x_1)\big)\\
=&\sum_{\nu=\tau}^{T-1}\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\nu}|x_{t_{\nu+1}})\big\|q(x_{t_\nu}|x_{t_{\nu+1}})\big)
\end{aligned}$$

$$\begin{aligned}
\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_0|x_{t_{\tau-1}})\big\|q(x_0|x_{t_{\tau-1}})\big)\le&\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_0,x_{t_1},\dots,x_{t_{\tau-2}}|x_{t_{\tau-1}})\big\|q(x_0,x_{t_1},\dots,x_{t_{\tau-2}}|x_{t_{\tau-1}})\big)\\
=&\sum_{\nu=0}^{\tau-2}\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\nu}|x_{t_{\nu+1}})\big\|q(x_{t_\nu}|x_{t_{\nu+1}})\big)
\end{aligned}$$

Therefore, we have

$$\|\widetilde{\mathcal{PG}}(\theta)-\mathcal{PG}(\theta)\|\le\sum_{\tau=1}^{T}M_{t_\tau}\sqrt{2\ln 2\Big[\sum_{\nu=\tau}^{T-1}\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\nu}|x_{t_{\nu+1}})\big\|q(x_{t_\nu}|x_{t_{\nu+1}})\big)+\sum_{\nu=0}^{\tau-2}\mathbb{D}_{\mathrm{KL}}\big(\tilde{q}(x_{t_\nu}|x_{t_{\nu+1}})\big\|q(x_{t_\nu}|x_{t_{\nu+1}})\big)\Big]}$$

$$= \sum_{\tau=1}^{T} M_{t_\tau} \sqrt{2 \ln 2 \sum_{\nu=0, \nu \neq \tau-1}^{T-1} \mathbb{D}_{\mathrm{KL}}\left(\tilde{q}(x_{t_\nu} | x_{t_{\nu+1}}) \middle\| q(x_{t_\nu} | x_{t_{\nu+1}})\right)}$$

□

D Implementation Details

In this section, we will describe the implementation of experiments in detail.

Guan et al. (2023) employed an SE(3)-equivariant diffusion model, named TargetDiff, for structure-based drug design. Given a protein binding site, TargetDiff generates the atom coordinates in 3D Euclidean space and atom types by iteratively denoising from a prior distribution. After the reverse (generative) process of the diffusion model, the chemical bonds of the generated ligand molecules are defined as post-processing by OpenBabel (O'Boyle et al., 2011) according to the distances and types of atom pairs. We use TargetDiff as the actor and strictly follows the setting in Guan et al. (2023), such as noise schedules, model architecture, training objectives, etc.

We first pretrain TargetDiff on the training set. After that, we use the pretrained TargetDiff to first sample 100 ligand molecules for each pocket in the test set and evaluate their binding affinity by oracle. We pretrain the critic, which predicts the binding affinity based on the perturbed samples, on the $10,000$ generated pocket-ligand pair data. The model architecture of the critic is almost the same with TargetDiff. The only difference is that the critic has an aggregation layer at last to output a scalar based on global features. We finetune the pretrained TargetDiff (Guan et al., 2023) for 30 iterations for each pocket in the test set, respectively. In each iteration, we sample 34 ligand molecules induced by the diffusion model (i.e., the actor), evaluate the binding affinity by oracle, and then online update the diffusion model (i.e., the actor) and train the policy and critic following Algorithm 2. We keep all sampled molecules in D_0 which falls into the class of off-policy policy gradient.

We use Adam (Kingma & Ba, 2014) with `init_learning_rate=0.001`, `betas=(0.95, 0.999)`, `batch_size=8`, and `clip_gradient_norm=8.0` to pretrain TargetDiff (i.e., the actor) and the critic. We use Adam with `init_learning_rate=0.0003` for online updating the actor and critic. As for regularization in Equation (13), we set $\eta_2 = 0.05$ for atom types and $\eta_2 = 0.00025$ for atom positions.

E Optimization Curves of 100 Protein Pockets

We plot the optimization curves of DiffAC for the pocket protein in the test separately in Figure 6, Figure 7, and Figure 8. Given a pocket protein, at each iteration, the average Vina Score of the sampled ligand molecules in this iteration is plotted as a point in the figure. The curves show how the binding affinity change with the number of optimization iterations. Generally, in most cases, DiffAC performs better than the baselines.

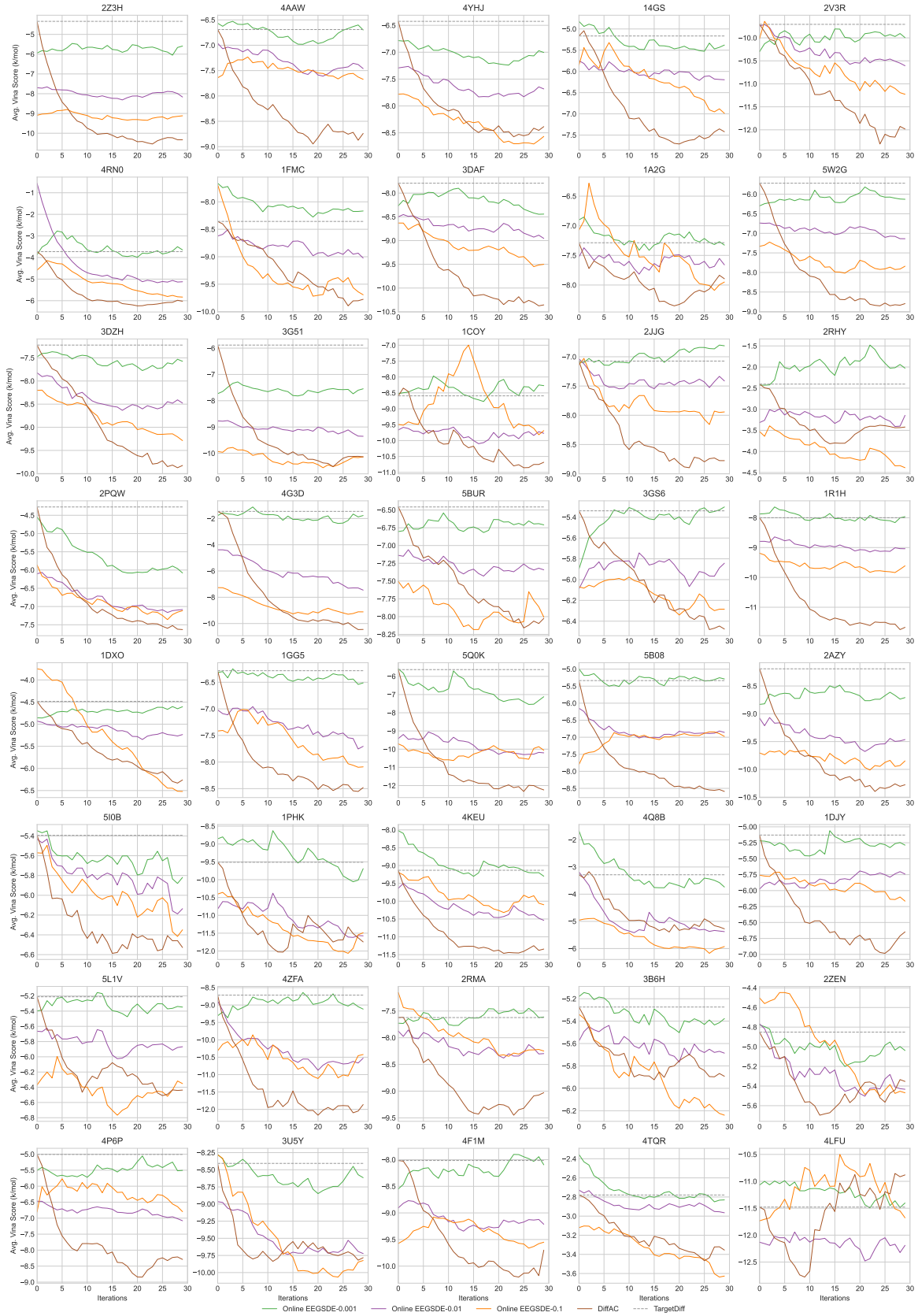

Figure 6. Optimization curves of the 1st to 40th protein pockets in the test set.

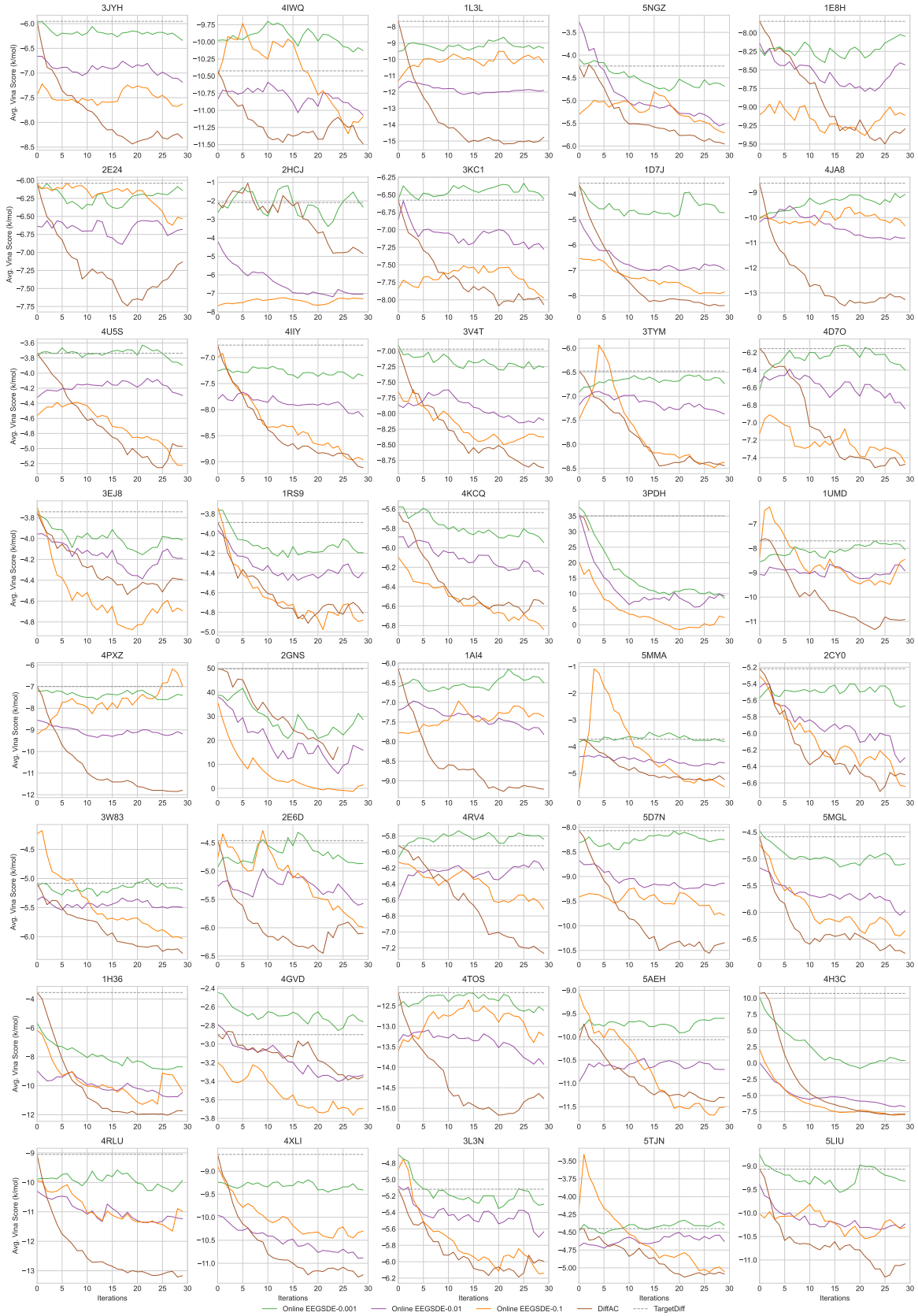

Figure 7. Optimization curves of the 41st to 80th protein pockets in the test set.

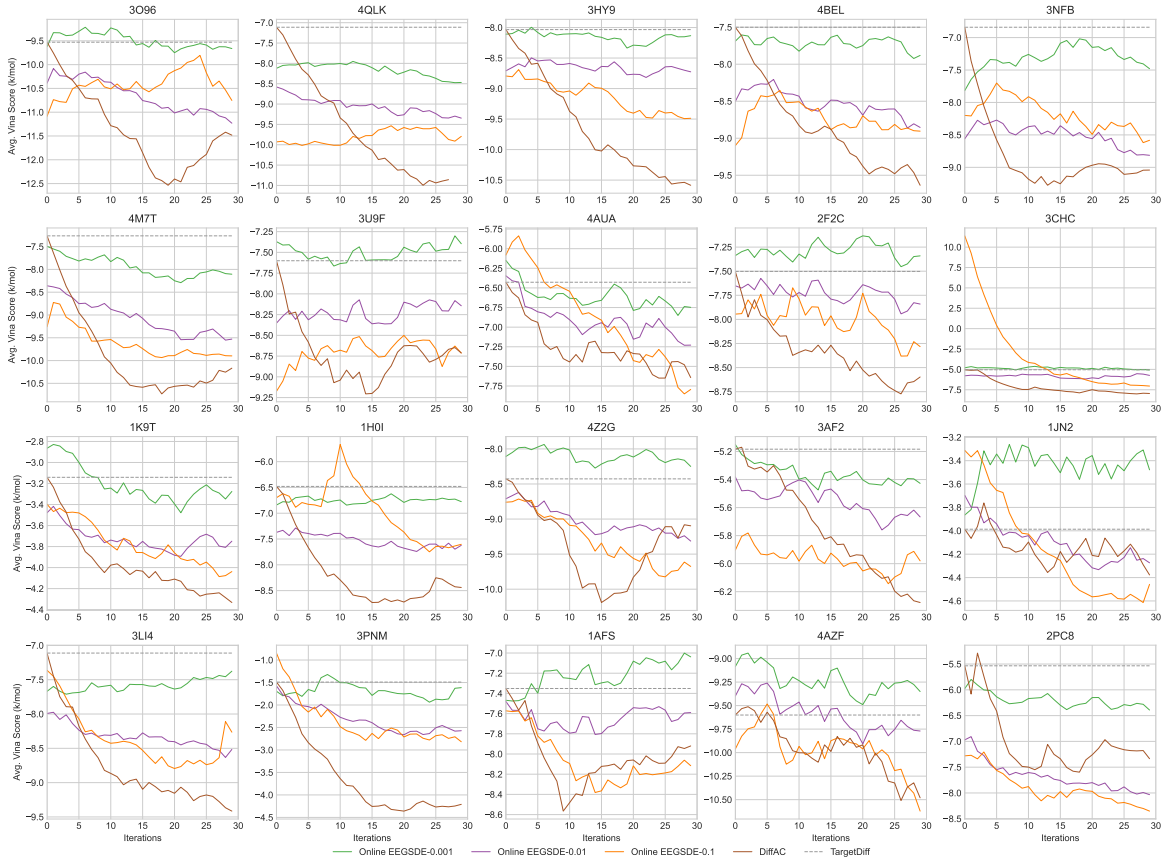

Figure 8. Optimization curves of the 81st to 100th protein pockets in the test set.

F Ablation Studies

To verify our hypothesis that the correct gradient estimation in our method requires the forward SDE and backward SDE to be consistent, we apply our method to structure-based drug design for three protein targets (PDB ID: 2Z3H, 4AAW, and 4YHJ) with different coefficients of regularization η_2 in Equation (13), respectively. The corresponding optimization curves are shown in Figure 9.

Optimization with too small regularization coefficients is superior to or on par with other settings at the first several steps and soon fails after a few optimization steps. This reflects two aspects: (1) The policy gradient estimation is approximately accurate so it has the better performance at the beginning of the optimization process; (2) The policy gradient estimation becomes incorrect just after a few steps due to significantly increased inconsistency between forward and backward SDEs. Optimization with too large regularization coefficients decreases the Vina scores slowly. Large regularization coefficients push the SDE policy to be close to the original backward SDE so that the performance is similar to the original. Only optimization with proper regularization coefficients can effectively decrease the Vina scores. This also aligns with our expectation. A proper regularization coefficient indicates both approximated consistency and ample room for optimization.

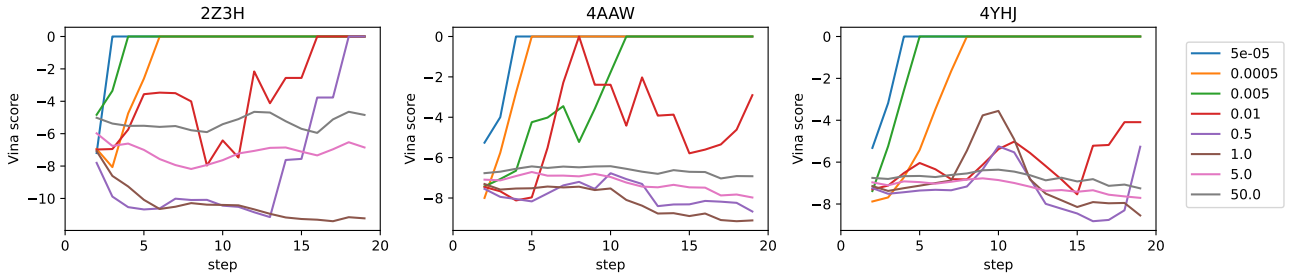

Figure 9. Optimization curves for protein target 2Z3H, 4AAW, and 4YHJ, respectively. The regularization coefficients η_2 in Equation (13) are $\{5 \times 10^{-5}, 0.0005, 0.005, 0.01, 0.5, 1.0, 5.0, 50.0\} \times 0.05$ (resp. 0.00025) for atom types (resp. atom positions). The Vina score of each point is averaged over 32 samples at this step and capped with 0.0 as the maximum value for clearer visualization.

G Experiments on Text-to-Image Generation

To demonstrate the generalizability of our method beyond the SBDD task, we also apply our method to text-to-image generation. In this experiment, we use DiffAC to fine-tune text-to-image generative models to better align with human preferences.

G.1 Experimental Setup

We use Stable Diffusion v1.5 (Rombach et al., 2022) as the baseline, which has been pre-trained on large image-text datasets (Schuhmann et al., 2021; 2022). For compute-efficient fine-tuning, we use Low-Rank Adaption (LoRA) (Hu et al., 2022), which freezes the parameters of the pre-trained model and introduces low-rank trainable weights. We apply LoRA to the UNet (Ronneberger et al., 2015) module and only update the added weights. For the reward model, we use ImageReward (Xu et al., 2023) which is trained on a large dataset comprised of human assessments of images. Compared to other scoring functions such as CLIP (Radford et al., 2021) or BLIP (Li et al., 2022), ImageReward has a better correlation with human judgments, making it the preferred choice for fine-tuning our baseline diffusion model. In practice, we use DiffAC (the REINFORCE version, i.e., Equation (10)) to fine-tune Stable Diffusion.

We also compare our method with DPOK (Fan et al., 2023). DPOK is a strong baseline that updates the pre-trained text-to-image diffusion models using policy gradient with KL regularization to maximize the reward. Notably, the difference between DPOK and our method is that DPOK estimates policy gradient with real trajectories sampled by backward process while our method estimates policy gradient with efficient forward process. And this difference is the key factor for stabler policy gradient.

We adopt a straightforward setup that uses one text prompt "A green colored rabbit" during fine-tuning and compares ImageReward scores of all methods. For both DPOK and our method, we perform 5 gradient steps per sampling step. The sampling batch size is 10 and the training batch size is 32.

G.2 Experimental Results

We plot the optimization curves of all methods as shown in Figure 10. As the results indicates, our method can efficiently improve ImageReward scores and outperform baselines by a large margin.

We provide image examples as shown in Figure 11. Stable Diffusion tends to generate images with obvious mistakes like generating a rabbit with a green background given the prompt "A green colored rabbit", while our method generates much more satisfying images that are well aligned with the given text prompt.

The experiments on text-to-image generation along with structure-based drug design demonstrate the generalizability of our method and reveal its great potential on many real-world applications.

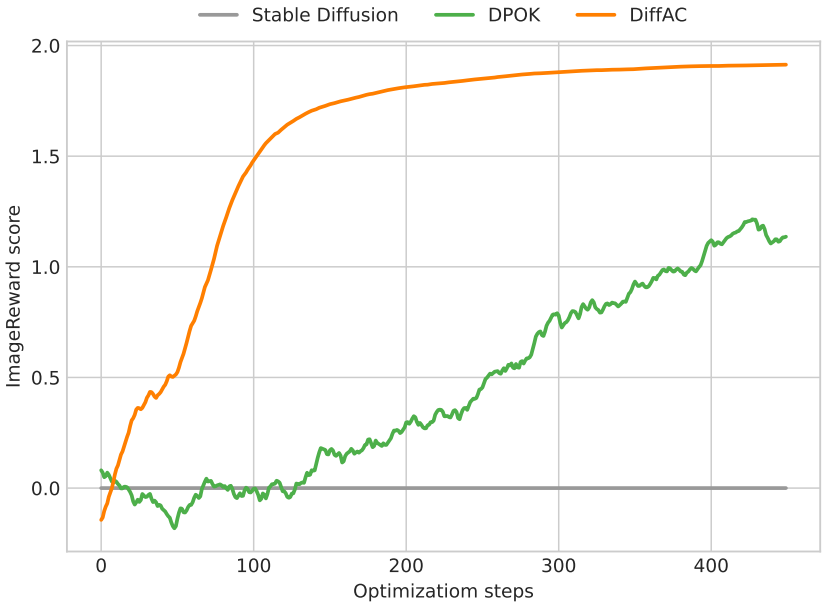

Figure 10. Optimization curves of ImageReward scores.

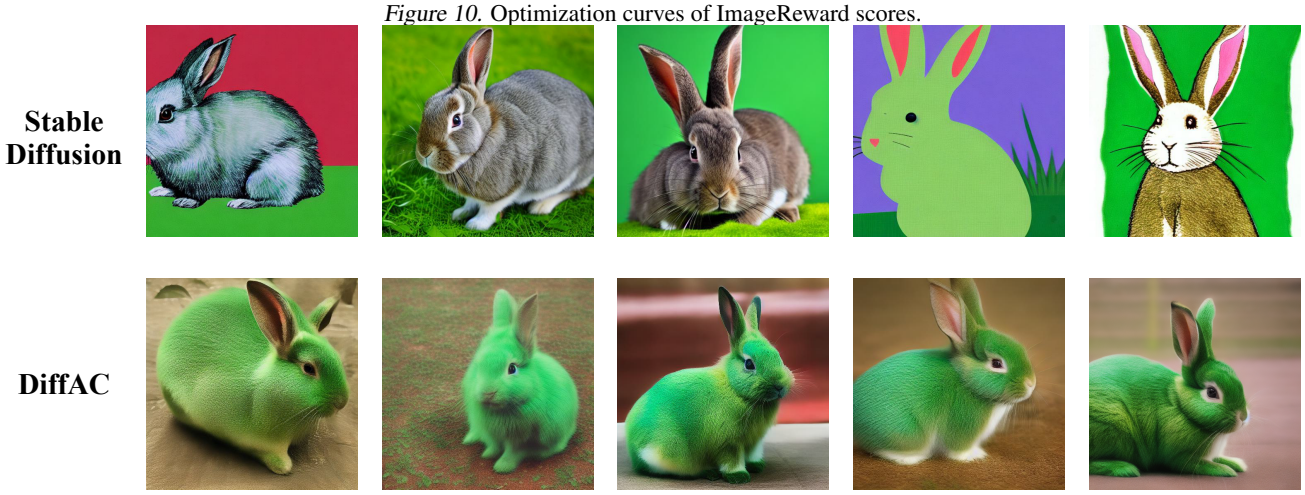

Figure 11. Example images generated by Stable Diffusion (Rombach et al., 2022) (top row) and our method (bottom row) givne text prompt "A green colored rabbit".