Extracting Reward Functions from Diffusion Models

Felipe Nuti[* 1]   Tim Franzmeyer[* 1]   João F. Henriques[1]

Abstract

Diffusion models have achieved remarkable results in image generation, and have similarly been used to learn high-performing policies in sequential decision-making tasks. Decision-making diffusion models can be trained on lower-quality data, and then be steered with a reward function to generate near-optimal trajectories. We consider the problem of extracting a reward function by comparing a decision-making diffusion model that models low-reward behavior and one that models high-reward behavior; a setting related to inverse reinforcement learning. We first define the notion of a *relative reward function of two diffusion models* and show conditions under which it exists and is unique. We then devise a practical learning algorithm for extracting it by aligning the gradients of a reward function – parametrized by a neural network – to the difference in outputs of both diffusion models. Our method finds correct reward functions in navigation environments, and we demonstrate that steering the base model with the learned reward functions results in significantly increased performance in standard locomotion benchmarks. Finally, we demonstrate that our approach generalizes beyond sequential decision-making by learning a reward-like function from two large-scale image generation diffusion models. The extracted reward function successfully assigns lower rewards to harmful images.

1. Introduction

Recent work [24, 2] demonstrates that diffusion models – which display remarkable performance on image generation – are similarly applicable to sequential decision-making. Leveraging a well-established framing of reinforcement learning as conditional sampling [36], Janner et al. [24] show that diffusion models can be used to parameterize a reward-agnostic prior distribution over trajectories, learned from offline demonstrations alone. Using classifier guidance [59], the diffusion model can then be steered with a (cumulative) reward function, parametrized by a neural network, to generate near-optimal behaviors in various se-
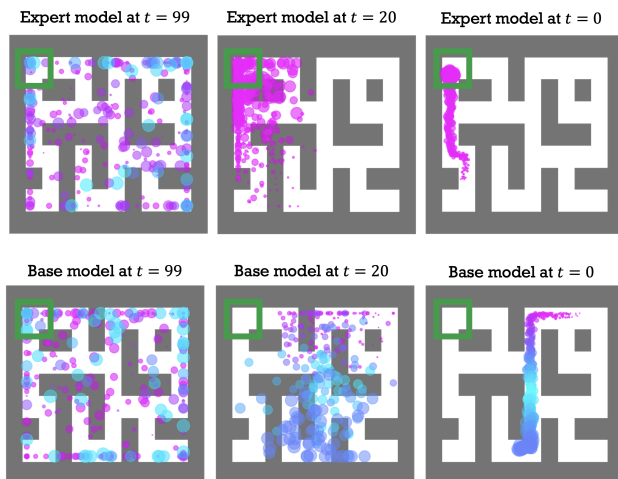

Figure 1: We see a 2D navigation environment in which an agent has to move through a maze to reach the goal (green box) in the top left corner. Earlier states of the agent are indicated by smaller dots. The top row shows the progression from an initial noised distribution over states (at diffusion timestep $t = 99$, top left) to a denoised high-reward expert trajectory (top right). The bottom row depicts the same process but for a low-reward trajectory where the agent moves in the wrong direction. We observe that our method correctly assigns high rewards (red colors) to the expert trajectory (top right), low rewards (blue colors) to the base trajectory that moves in the wrong direction (bottom right), and uniform rewards to the diffused trajectories ($t = 99$).

quential decision-making tasks. Hence, it is possible to learn successful policies both through (a) training an *expert diffusion model* on a distribution of optimal trajectories, and (b) by training a *base diffusion model* on lower-quality or reward-agnostic trajectories and then steering it with the given reward function. This suggests that the reward function can be extracted by comparing the distributions produced by such base and expert diffusion models.

The problem of learning the preferences of an agent from observed behavior, often expressed as a reward function, is considered in the Inverse Reinforcement Learning (IRL) literature [55, 42]. In contrast to merely imitating the observed behavior, learning the reward function behind it allows for better robustness, better generalization to distinct environments, and for combining rewards extracted from multiple behaviors.

Prior work in reward learning is largely based on the Maximum Entropy Inverse Reinforcement Learning (MaxEntIRL) framework [68], which relies on alternating between policy optimization and reward learning. This often comes with the assumption of access to the environment (or a simulator) to train the policy. Reward learning is also of independent interest outside of policy learning, for understanding agents' behavior or predicting their actions, with applications in Value Alignment, interpretability, and AI Safety [3, 14, 54]. For example, reward functions can also be utilized to better understand an existing AI system's explicit or implicit "preferences" and tendencies [56].

In this work, we introduce a method for extracting a reward function from decision-making diffusion models, illustrated in Figure 1. In contrast to prior work, our method does not require environment access, simulators, or iterative policy optimization. Further, it is agnostic to the architecture of the diffusion models used, being applicable to continuous and discrete models, and making no assumption of whether the models are unguided, or whether they use either classifier guidance [59] or classifier-free guidance [21].

We first derive a notion of a *relative reward function of two diffusion models*. We show that, under mild assumptions on the trajectory distribution and diffusion sampling process, our notion of reward exists and is unique, up to an additive constant. Further, we show that the derived reward is equivalent to the true reward under the probabilistic RL framework [36]. Finally, we propose a practical learning algorithm for extracting the relative reward function of two diffusion models by aligning the gradients of the learned reward function with the *differences of the outputs* of the base and expert models.

We empirically evaluate our reward learning method along three axes. In the Maze2D environments [12], we learn a reward function by comparing a base diffusion model trained on exploratory trajectories and an expert diffusion model trained on goal-directed trajectories, as illustrated in Figure 1. We can see in Figure 2 that our method learns the correct reward function for varying maze configurations. In the common locomotion environments Hopper, HalfCheetah, and Walker2D [12, 7], we learn a reward function by comparing a low-performance base model to an expert diffusion model and demonstrate that steering with the base model with the learned reward function results in a significantly improved performance. Beyond sequential-decision making, we learn a reward-like function by comparing a base image generation diffusion model (Stable Diffusion, [53]) to a *safer* version of Stable Diffusion [58]. Figure 4 shows that the learned reward function penalizes images with harmful content, such as violence and hate, while rewarding harmless images.

In summary, our work makes the following contributions:

- We introduce the concept of *relative reward functions* of diffusion models, and provide a mathematical analysis of their relation to rewards in sequential decision-making.
- We propose a practical learning algorithm for extracting relative reward functions by aligning the reward function's gradient with the difference in outputs of two diffusion models.
- We empirically validate our method in long-horizon planning environments, in high-dimensional control environments, and show generalization beyond sequential decision-making.

2. Related Work

Diffusion models, originally proposed by Sohl-Dickstein et al. [59], are an expressive class of generative models that generate samples by learning to invert the process of noising the data. The work of Ho et al. [20] led to a resurgence of interest in the method, followed by [43, 10, 60]. Song et al. [63] proposed a unified treatment of diffusion models and score-based generative models [61, 62] through stochastic differential equations [44], used e.g. in [31]. Diffusion models have shown excellent performance in image generation [53, 10, 58, 51], molecule generation [22, 25, 26], 3D generation [47, 40], video generation [19], language [37], and, crucially for this work, sequential decision-making, among many others. Part of their appeal as generative models is due to their *steerability*. Since sample generation is gradual, other pre-trained neural networks can be used to steer the diffusion model during sampling (i.e. classifier guidance, Section A.1). An alternative approach is classifier-free guidance [21], often used in text-to-image models such as [53].

Decision-making with diffusion models has first been proposed by Janner et al. [24], who presented a hybrid solution for planning and dynamics modeling by iteratively denoising a sequence of states and actions at every time step, and executing the first denoised action. This method builds on the *Probabilistic inference framework for Reinforcement Learning*, reviewed by Levine [36], based on works including [64, 67, 68, 30, 29]. Diffusion models have been applied in the standard reinforcement learning setting [38, 2, 8], and also in 3D domains [23]. They have similarly been used in imitation learning for generating more human-like policies [46, 52], for traffic simulation [66], and for offline reinforcement learning [24, 65, 17]. Within sequential-decision making, much interest also lies in extracting reward functions from observed behavior. This problem of *Inverse Reinforcement learning* [1, 5] has mostly either been approached by making strong assumptions about the structure of the reward function [68, 42, 15], or, more recently, by employing adversarial training objectives [13].

The optimization objective of our method resembles that of

Physics-Informed Neural Networks [6, 16, 39, 49, 50, 11], which also align neural network gradients to a pre-specified function. In our case, this function is a difference in the outputs of two diffusion models.

To the best of our knowledge, no previous IRL methods are directly applicable to diffusion models, nor has extracting a relative reward function from two diffusion models been explored before.

3. Background

Our method leverages mathematical properties of diffusion-based planners to extract reward functions from them. To put our contributions into context and define notation, we give a more detailed overview of the probabilistic formulation of Reinforcement Learning presented in [36], then of diffusion models, and finally of how they come together in decision-making diffusion models [24] in Appendix A.1. $\Delta_{\mathcal{X}}$ denotes the set of probability distributions over a set $\mathcal{X}$.

Markov Decision Process (MDP). A tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho_0, T \rangle$ consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathbb{R}_{\geq 0}}$, an initial state distribution $\rho_0 \in \Delta_{\mathcal{S}}$, and an episode length T. The MDP starts at an initial state $s_0 \sim \rho_0$, and evolves by sampling $a_t \sim \pi(s_t)$, and then $s_{t+1} \sim \mathcal{T}(s_t, a_t)$ for $t \geq 0$. The reward received at time t is $r_t \sim r(s_t, a_t)$. The episode ends at $t = T$. The sequence of state-action pairs $\tau = ((s_t, a_t))_{t=0}^{T}$ is called a *trajectory*.

Reinforcement Learning as Probabilistic Inference. Levine [36] provides an in-depth exposition of existing methods for approaching sequential decision-making from a probabilistic and causal angle using Probabilistic Graphical Models (PGM) [27]. The framework in [36] recasts such an MDP as a PGM, illustrated in Appendix C, Figure 5b. It consists of a sequence of states $(s_t)_{t=0}^{T}$, actions $(a_t)_{t=0}^{T}$ and *optimality variables* $(\mathcal{O}_t)_{t=0}^{T}$. The reward function r is not explicitly present. Instead, it is encoded in the optimality variables via the relation: $\mathcal{O}_t \sim \text{Ber}(e^{-r(s_t, a_t)})$. One can apply Bayes's Rule to obtain:

$$p(\tau|\mathcal{O}_{1:T}) \propto p(\tau) \cdot p(\mathcal{O}_{1:T}|\tau) \tag{1}$$

which factorizes the distribution of optimal trajectories (up to a normalizing constant) as a *prior* $p(\tau)$ over trajectories and a *likelihood term* $p(\mathcal{O}_{1:T}|\tau)$. From the definition of $\mathcal{O}_{1:T}$ and the PGM structure, we have $p(\mathcal{O}_{1:T}|\tau) = e^{-\sum_t r(s_t, a_t)}$. Hence, the negative log-likelihood of optimality conditioned on a trajectory τ corresponds to its cumulative reward: $-\log p(\mathcal{O}_{1:T}|\tau) = \sum_t r(s_t, a_t)$.

Planning with Diffusion. Sequential decision-making can be framed as sampling from a posterior distribution $p(\tau|\mathcal{O}_{1:T})$ over trajectories. Section A.1 shows how a dif-

fusion model $p(\mathbf{x}_0)$ can be combined with a classifier to sample from a posterior $p(\mathbf{x}_0|y)$. These two observations point us to the approach in Diffuser [24]: using a diffusion model to model a prior $p(\tau)$ over trajectories, and a reward prediction function $\rho(x, t) \approx p(\mathcal{O}_{1:T}|\tau_t)$ to steer the diffusion model. This allows approximate sampling from $p(\tau|\mathcal{O}_{1:T})$, which produces (near-)optimal trajectories.

The policy generated by the Diffuser denoises a fixed-length sequence of future states and actions and executes the first action of the sequence. Diffuser can also be employed for goal-directed planning, by fixing initial and goal states during the denoising process.

4. Methods

In the previous section, we established the connection between the (cumulative) return $\sum_t r(s_t, a_t)$ of a trajectory $\tau = \{(s_t, a_t)\}_{t=1}^{T}$ and the value $p(y|\mathbf{x})$ used for classifier guidance (Section A.1). Hence, we can look at our goal as finding $p(y|\mathbf{x})$, or equivalently $p(\mathcal{O}_{1:T}|\tau)$, in order to recover the cumulative reward for a given trajectory τ. Additionally, in Section 4.2, we will demonstrate how the single-step reward can be computed by choosing a specific parametrization for $p(y|\mathbf{x})$.

Problem Setting. We consider a scenario where we have two decision-making diffusion models: a base model $\mathbf{s}_{\phi}^{(1)}$ that generates reward-agnostic trajectories, and an expert model $\mathbf{s}_{\Theta}^{(2)}$ that generates trajectories optimal under some *unknown* reward function r. Our objective is to learn a reward function $\rho(\mathbf{x}, t)$ such that, if ρ is used to steer the base model $\mathbf{s}_{\phi}^{(1)}$ through classifier guidance, we obtain a distribution close to that of the expert model $\mathbf{s}_{\Theta}^{(2)}$. From the above discussion, such a function ρ would correspond to the notion of relative reward in the probabilistic RL setting.

In the following, we present theory showing that:

1. In an idealized setting where $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$ have no approximation error (and are thus conservative vector fields), there exists a unique function ρ that exactly converts $\mathbf{s}^{(1)}$ to $\mathbf{s}^{(2)}$ through classifier guidance.
2. In practice, we cannot expect such a classifier to exist, as approximation errors might result in diffusion models corresponding to non-conservative vector fields.
3. However, the functions ρ that *best approximate* the desired property (to arbitrary precision ε) do exist. These are given by Def. 4.2 and can be obtained through a *projection* using an L^2 distance.
4. The use of an L^2 distance naturally results in an L^2 loss for learning ρ with Gradient Descent.

Algorithm 1: Relative reward function training.

Input: Base $\mathbf{s}^{(1)}$ and expert $\mathbf{s}^{(2)}$ diffusion models, dataset $\mathcal{D}$, number of iterations I.

Output: Relative reward estimator ρ_θ.

Initialize reward estimator parameters θ.

for $j \in \{1, ..., I\}$ **do**

    Sample batch: $\mathbf{X}_0 = [\mathbf{x}_0^{(1)}, ..., \mathbf{x}_0^{(N)}]$ from $\mathcal{D}$

    Sample times: $\mathbf{t} = [t_1, ..., t_N]$ independently in $\mathcal{U}(0, T]$

    Sample forward process: $\mathbf{X_t} \leftarrow [\mathbf{x}_{t_1}^{(1)}, ..., \mathbf{x}_{t_N}^{(N)}]$

    Take an optimization step on θ according to

    $\hat{L}_{\text{RRF}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}||\nabla_\mathbf{x}\rho_\theta(\mathbf{x}_{t_i}^{(i)}, t_i) - (\mathbf{s}_\Theta^{(2)}(\mathbf{x}_{t_i}^{(i)}, t_i) - \mathbf{s}_\phi^{(1)}(\mathbf{x}_{t_i}^{(i)}, t_i))||_2^2$

end

4.1. Relative Reward Function of Two Diffusion Models

In AppendixA.2 we provide a result saying that, once $\mathbf{s}_\phi^{(1)}$ and $\mathbf{s}_\Theta^{(2)}$ are fixed, we obtain a condition on the gradients of ρ that, if met, would allow us not only to match the *distributions* of $\mathbf{s}_\phi^{(1)}$ and $\mathbf{s}_\Theta^{(2)}$, but also their entire denoising processes, with probability 1 (i.e. *almost surely*, a.s.).

To get around the possibility that $\mathbf{s}^{(1)}$ and $\mathbf{s}^{(2)}$ are not conservative, we may instead look for the conservative field best approximating $\mathbf{s}^{(2)}(\mathbf{x}, t) - \mathbf{s}^{(1)}(\mathbf{x}, t)$ in $L^2(\mathbb{R}^n, \mathbb{R}^n)$ (i.e. the space of square-integrable vector fields, endowed with the L^2 norm). Using a well-known fundamental result on uniqueness of projections in L^2 (Th. D.8), we obtain the following:

Proposition 4.1 (Optimal Relative Reward Gradient)**.** *Let $\mathbf{s}_\Theta^{(2)}$ and $\mathbf{s}_\phi^{(1)}$ be any two diffusion models and $t \in (0, T]$, with the assumption that $\mathbf{s}_\Theta^{(2)}(\cdot, t) - \mathbf{s}_\phi^{(1)}(\cdot, t)$ is square-integrable. Then there exists a unique vector field $\mathbf{h}_t$ given by*

$$\mathbf{h}_t = \underset{\mathbf{f} \in \overline{\text{Cons}(\mathbb{R}^n)}}{\text{argmin}} \int_{\mathbb{R}^n} ||\mathbf{f}(\mathbf{x}) - (\mathbf{s}_\Theta^{(2)}(\mathbf{x}, t) - \mathbf{s}_\phi^{(1)}(\mathbf{x}, t))||_2^2 \, d\mathbf{x}$$

(2)

where $\overline{\text{Cons}(\mathbb{R}^n)}$ denotes the closed span of gradients of smooth $W^{1,2}$ potentials. Furthermore, for any $\varepsilon > 0$, there is a smooth, square-integrable potential Φ with a square-integrable gradient satisfying:

$$\int_{\mathbb{R}^n} ||\nabla_\mathbf{x}\Phi(\mathbf{x}) - \mathbf{h}_t(\mathbf{x})||_2^2 \, d\mathbf{x} < \varepsilon$$

(3)

*We call such an $\mathbf{h}_t$ the **optimal relative reward gradient of** $\mathbf{s}_\phi^{(1)}$ **and** $\mathbf{s}_\Theta^{(2)}$ **at time** t.*

For proof of Proposition 4.1 see Appendix D.4. It is important to note that for the projection to be well-defined,

Figure 2: Maze2D learned reward heatmaps. ★ denotes the ground-truth goal position of the expert diffusion model. For more examples, see Appendix G.

we required an assumption regarding the integrability of the diffusion models. Without this assumption, the integral would simply diverge.

The result in Proposition 4.1 tells us that we can get arbitrarily close to the optimal relative reward gradient using scalar potentials' gradients. Therefore, we may finally define the central notion in this paper:

Definition 4.2 (ε-Relative Reward Function)**.** For an $\varepsilon > 0$, an ε-relative reward function of diffusion models $\mathbf{s}_\phi^{(1)}$ and $\mathbf{s}_\Theta^{(2)}$ is a function $\rho : \mathbb{R}^n \times [0, T] \rightarrow \mathbb{R}$ such that

$$\forall t \in (0, T] : \int_{\mathbb{R}^n} ||\nabla_\mathbf{x}\rho(\mathbf{x}, t) - \mathbf{h}_t(\mathbf{x})||_2^2 \, d\mathbf{x} < \varepsilon \quad (4)$$

where $\mathbf{h}_t$ denotes the optimal relative reward gradient of $\mathbf{s}_\phi^{(1)}$ and $\mathbf{s}_\Theta^{(2)}$ at time t.

4.2. Extracting Reward Functions

We now set out to actually approximate the relative reward function ρ. Definition 4.2 naturally translates into an L^2 training objective for learning ρ:

$$L_{\text{RRF}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,T], \mathbf{x}_t \sim p_t}$$
$$\left[||\nabla_\mathbf{x}\rho_\theta(\mathbf{x}_t, t) - (\mathbf{s}_\Theta^{(2)}(\mathbf{x}_t, t) - \mathbf{s}_\phi^{(1)}(\mathbf{x}_t, t))||_2^2 \right]$$

(5)

where p_t denotes the marginal at time t of the forward noising process.

We optimize this objective via Empirical Risk Minimization and Stochastic Gradient Descent. See Algorithm 1 for a version assuming access to the diffusion models and their training datasets, and Algorithm 2 in Appendix E for one which does not assume access to any pre-existing dataset. Our method requires no access to the environment or to a simulator. Our algorithm requires computing a second-order mixed derivative $D_\theta(\nabla_\mathbf{x}\rho(\mathbf{x}, t)) \in \mathbb{R}^{m \times n}$ (where m is the number of parameters θ), for which we use automatic differentiation in PyTorch [45].

Recovering per-time-step rewards. We can parameterize ρ using a single-time-step neural network $g_\theta(s, a, t)$ as $\rho(\tau_t, t) = \frac{1}{N}\sum_{i=1}^{N} g_\theta(s_t^i, a_t^i, t)$, where N is the horizon and τ_t denotes a trajectory at diffusion timestep t, and s_t^i and a_t^i denote the ith state and action in τ_t. Then, $g_\theta(s, a, 0)$ predicts a reward for a state-action pair (s, a).

Table 1: Diffuser performance with different steering methods on 3 RL environments.

Environment	Unsteered	Discriminator	Reward (Ours)
Halfcheetah	30.38 ± 0.38	30.41 ± 0.38	$\mathbf{31.65 \pm 0.32}$
Hopper	24.67 ± 0.92	25.12 ± 0.95	$\mathbf{27.04 \pm 0.90}$
Walker2d	28.20 ± 0.99	27.98 ± 0.99	$\mathbf{38.40 \pm 1.02}$
Mean	27.75	27.84	**32.36**

5. Experiments

In this section, we conduct empirical investigations to analyze the properties of relative reward functions in practice. Our experiments focus on three main aspects: the alignment of learned reward functions with the goals of the expert agent, the performance improvement achieved through steering the base model using the learned reward, and the generalizability to learning reward-like functions to domains beyond decision-making (note that the relative reward function is defined for any pair of diffusion models). For implementation details and computational requirements please refer to the Appendix.

5.1. Learning Correct Reward Functions from Long-Horizon Plans

Maze2D [12] features various environments which involve controlling the acceleration of a ball to navigate it towards various goal positions in 2D mazes. It is suitable for evaluating reward learning, as it requires effective credit assignment over extended trajectories.

Implementation. To conduct our experiments, we generate multiple datasets of trajectories in each of the 2D environments, following the data generation procedure from D4RL [12], except sampling start and goal positions uniformly at random (as opposed to only at integer coordinates).

For four maze environments with different wall configurations (depicted in Figure 2), we first train a base diffusion model on a dataset of uniformly sampled start and goal positions, hence representing undirected behavior. For each environment and across five random seeds, we then train eight expert diffusion models on datasets with fixed goal positions. For each of the 32 expert models, we train a relative reward estimator ρ_θ, implemented as an MLP, via gradient alignment, as described in Alg. 1.

Discriminability results. We assess the effectiveness of the proposed reward learning method in assigning higher rewards to expert trajectories. We reserve a validation set comprising 10% of the data for testing purposes and compute the estimated rewards for samples from this set. The results are visualized in a histogram in App. H. We find that the distributions are clearly separated.

For a quantitative evaluation, we train a logistic regression classifier per expert model on a balanced dataset to label
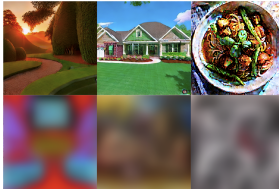

Figure 3: The 3 images with the highest learned reward in their batch ("safe"), and 3 with the lowest reward ("unsafe", blurred), respectively.

Figure 4: Learned rewards for base and expert diffusion models from Stable Diffusion (Sec. 5).

trajectories (as base or expert) using the predicted reward. We repeat this process for each goal position with 5 different seeds, and average accuracies across seeds. The achieved accuracies range from 65.33% to 97.26%, with a median of 84.49% and a mean of 83.76%. These results demonstrate that the learned reward effectively discriminates expert trajectories.

Visualization of results. To visualize the rewards, we use our model to predict rewards for fixed-length sub-trajectories in the base dataset. We then generate a 2D heatmap by averaging the rewards of all sub-trajectories that pass through each grid cell. Fig. 2 displays some of these heatmaps.

We observe that the network accurately captures the rewards, with peaks occurring at the true goal position of the expert dataset in 78.75% ± 8.96% of the cases for simpler mazes (maze2d-open-v0 and maze2d-umaze-v0), and 77.50% ± 9.15% for more advanced mazes (maze2d-medium-v1 and maze2d-large-v1). Overall, the network achieves an average success rate of 78.12% ± 6.40%.

Sensitivity to dataset size. Since training diffusion models typically demands a substantial amount of data, we investigate the sensitivity of our method to dataset size. Remarkably, we observe that the performance only experiences a slight degradation even with significantly smaller sizes. For complete results, please refer to Appendix G.

5.2. Steering diffusion models to improve their performance

Having established the effectiveness of relative reward functions in recovering expert goals in the low-dimensional Maze2D environment, we now examine their applicability in higher-dimensional control tasks. Specifically, we evaluate their performance in the HalfCheetah, Hopper, and Walker-2D environments from the D4RL offline locomotion suite [12]. These tasks involve controlling a multi-degree-of-freedom 2D robot to move forward with the highest speed.

Given that the state spaces of these environments are high-dimensional, visualizing the rewards as in Maze2D becomes impractical. Instead, we assess the learned reward functions by examining whether they can enhance the performance of a weak base model when used for classifier-guided steering. If successful, this would provide evidence that our learned relative rewards can indeed bring the base trajectory distribution closer to the expert distribution.

Implementation. In these locomotion environments, the notion of reward is primarily focused on moving forward. Therefore, instead of selecting expert behaviors from a diverse set of base behaviors, our reward function aims to guide a below-average-performing base model towards improved performance. Specifically, we train the base models using the `medium-replay` datasets from D4RL, which yield low rewards, and the expert models using the `expert` datasets, which yield high rewards. The reward functions are then fitted using gradient alignment as described in Algorithm 1. Finally, we employ classifier guidance (Section 3) to steer each base model using the corresponding learned reward function.

Results. We conduct 512 independent rollouts of the base diffusion model steered by the learned reward, using various guidance scales ω (Eq. 9). We use the unsteered base model as a baseline. We also compare our approach to a discriminator with the same architecture as our reward function, trained to predict whether a trajectory originates from the base or expert dataset. We train our models with 5 random seeds, and run the 512 independent rollouts for each seed. Steering the base models with our learned reward functions consistently leads to statistically significant performance improvements across all three environments. Notably, the Walker2D task demonstrates a 36.17% relative improvement compared to the unsteered model. This outcome suggests that the reward functions effectively capture the distinctions between the two diffusion models. See Table 1.

5.3. Learning a reward-like function for Stable Diffusion

While reward functions are primarily used in sequential decision-making problems, we propose a generalization to more general domains through the concept of relative reward functions, as discussed in Section 4. To empirically evaluate this generalization, we focus on one of the domains where diffusion models have demonstrated exceptional performance: image generation. Specifically, we examine Stable Diffusion [53], a widely used 859m-parameter diffusion model for general-purpose image generation, and Safe Stable Diffusion [58], a modified version of Stable Diffusion designed to mitigate the generation of shocking images.

Models. The models under consideration are *latent diffusion models*, where the denoising process occurs in a latent space and is subsequently decoded into an image. These models employ classifier-free guidance [21] during sampling and can be steered using natural language prompts, utilizing CLIP embeddings [48] in the latent space. Specifically, Safe Stable Diffusion [58] introduces modifications to the *sampling loop* of the open-source model proposed by Rombach et al. [53], without altering the model's actual weights. In contrast to traditional classifier-free guidance that steers samples toward a given prompt, the modified sampler of Safe Stable Diffusion also directs samples *away* from undesirable prompts [58, Sec. 3].

Prompt dataset. To investigate whether our reward networks can detect a "relative preference" of Safe Stable Diffusion over the base Stable Diffusion model for harmless images, we use the I2P prompt dataset introduced by Schramowski et al. [58]. This dataset consists of prompts specifically designed to deceive Stable Diffusion into generating imagery with unsafe content. However, we use the dataset to generate sets of *image embeddings* rather than actual images, which serve as training data for our reward networks. A portion of the generated dataset is set aside for evaluation.

Separating Image Distributions. Despite the complex and multimodal nature of the data distribution in this context, we observe that our reward networks are capable of distinguishing between base and expert images with over 90% accuracy, despite not being explicitly trained for this task. The reward histogram is visualized in Figure 4.

Qualitative Evaluation. We find that images that receive high rewards correspond to safe content, while those with low rewards typically contain unsafe or disturbing material, including hateful or offensive imagery. To illustrate this, we sample batches from the validation set, compute the rewards for each image, and decode the image with the highest reward and the one with the lowest reward from each batch. Considering the sensitive nature of the generated images, we blur the latter set as an additional safety precaution. Example images can be observed in Figure 3.

6. Conclusion

To the best of our knowledge, our work introduces the first method for extracting relative reward functions from two diffusion models. We provide theoretical justification for our approach and demonstrate its effectiveness in diverse domains and settings. We expect that our method has the potential to facilitate the learning of reward functions from large pre-trained models, improving our understanding and the alignment of the generated outputs. It is important to note that our experiments primarily rely on simulated environments, and further research is required to demonstrate its applicability in real-world scenarios.

References

[1] Abbeel, P. and Ng, A. Y. [2004], Apprenticeship learning via inverse reinforcement learning, *in* 'Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004'.

[2] Ajay, A., Du, Y., Gupta, A., Tenenbaum, J., Jaakkola, T. and Agrawal, P. [2022], 'Is conditional generative modeling all you need for decision-making?', *arXiv preprint arXiv:2211.15657* .

[3] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J. and Mané, D. [2016], 'Concrete problems in ai safety'.

[4] Anderson, B. D. [1982], 'Reverse-time diffusion equation models', *Stochastic Processes and their Applications* **12**(3), 313–326.
URL: *https://www.sciencedirect.com/science/article/pii/0304414982900515*

[5] Arora, S. and Doshi, P. [2021], 'A survey of inverse reinforcement learning: Challenges, methods and progress', *Artificial Intelligence* **297**, 103500.

[6] *Artificial neural networks for solving ordinary and partial differential equations* [1998], *IEEE transactions on neural networks* **9**(5), 987–1000.

[7] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W. [2016], 'Openai gym', *arXiv preprint arXiv:1606.01540* .

[8] Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B. and Song, S. [2023], 'Diffusion policy: Visuomotor policy learning via action diffusion', *arXiv preprint arXiv:2303.04137* .

[9] De Bortoli, V. [2022], 'Convergence of denoising diffusion models under the manifold hypothesis', *arXiv preprint arXiv:2208.05314* .

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

[11] Fang, J., Liu, C., Simos, T. and Famelis, I. T. [2020], 'Neural network solution of single-delay differential equations', *Mediterranean Journal of Mathematics* **17**, 1–15.

[12] Fu, J., Kumar, A., Nachum, O., Tucker, G. and Levine, S. [2020], 'D4rl: Datasets for deep data-driven reinforcement learning', *arXiv preprint arXiv:2004.07219* .

[13] Fu, J., Luo, K. and Levine, S. [2017], 'Learning robust rewards with adversarial inverse reinforcement learning', *arXiv preprint arXiv:1710.11248* .

[14] Gabriel, I. [2020], 'Artificial Intelligence, Values, and Alignment', *Minds and Machines* **30**(3), 411–437.
URL: *https://doi.org/10.1007/s11023-020-09539-2*

[15] Hadfield-Menell, D., Russell, S. J., Abbeel, P. and Dragan, A. [2016], Cooperative inverse reinforcement learning, *in* 'Advances in neural information processing systems', pp. 3909–3917.

[16] Han, J., Jentzen, A. and E, W. [2018], 'Solving high-dimensional partial differential equations using deep learning', *Proceedings of the National Academy of Sciences* **115**(34), 8505–8510.

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

[18] Hendrycks, D. and Gimpel, K. [2016], 'Gaussian error linear units (gelus)', *arXiv preprint arXiv:1606.08415* .

[19] Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J. et al. [n.d.], 'Imagen video: High definition video generation with diffusion models'.

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

[21] Ho, J. and Salimans, T. [n.d.], Classifier-free diffusion guidance, *in* 'NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications'.

[22] Hoogeboom, E., Satorras, V. G., Vignac, C. and Welling, M. [2022], Equivariant diffusion for molecule generation in 3d, *in* 'International Conference on Machine Learning', PMLR, pp. 8867–8887.

[23] Huang, S., Wang, Z., Li, P., Jia, B., Liu, T., Zhu, Y., Liang, W. and Zhu, S.-C. [2023], 'Diffusion-based generation, optimization, and planning in 3d scenes', *arXiv preprint arXiv:2301.06015* .

[24] Janner, M., Du, Y., Tenenbaum, J. B. and Levine, S. [2022], 'Planning with diffusion for flexible behavior synthesis'.
URL: *https://arxiv.org/abs/2205.09991*

[25] Jing, B., Corso, G., Chang, J., Barzilay, R. and Jaakkola, T. S. [n.d.], Torsional diffusion for molecular conformer generation, *in* 'Advances in Neural Information Processing Systems'.

[26] Jing, B., Erives, E., Pao-Huang, P., Corso, G., Berger, B. and Jaakkola, T. S. [n.d.], Eigenfold: Generative protein structure prediction with diffusion models, *in* 'ICLR 2023-Machine Learning for Drug Discovery workshop'.

[27] Jordan, M. I. [2004], 'Graphical models', *Statistical Science* **19**(1), 140–155.
URL: *http://www.jstor.org/stable/4144379*

[28] Jost, J. [2012], *Partial differential equations*, Vol. 214, Springer Science & Business Media.

[29] Kappen, H. [2011], 'Optimal control theory and the linear bellman equation'.

[30] Kappen, H. J., Gómez, V. and Opper, M. [2012], 'Optimal control as a graphical model inference problem', *Machine learning* **87**, 159–182.

[31] Karras, T., Aittala, M., Aila, T. and Laine, S. [n.d.], Elucidating the design space of diffusion-based generative models, *in* 'Advances in Neural Information Processing Systems'.

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

[33] Kreyszig, E. [1978], 'Introduction to functional analysis with application, book'.

[34] Lax, P. D. and Terrell, M. S. [2017], *Multivariable calculus with applications*, Springer.

[35] Le Gall, J.-F. [2016], *Brownian motion, martingales, and stochastic calculus*, Springer.

[36] Levine, S. [2018], 'Reinforcement learning and control as probabilistic inference: Tutorial and review', *arXiv preprint arXiv:1805.00909* .

[37] Li, X., Thickstun, J., Gulrajani, I., Liang, P. S. and Hashimoto, T. B. [2022], 'Diffusion-lm improves controllable text generation', *Advances in Neural Information Processing Systems* **35**, 4328–4343.

[38] Liang, Z., Mu, Y., Ding, M., Ni, F., Tomizuka, M. and Luo, P. [2023], 'Adaptdiffuser: Diffusion models as adaptive self-evolving planners', *arXiv preprint arXiv:2302.01877* .

[39] Magill, M., Qureshi, F. and de Haan, H. [2018], 'Neural networks trained to solve differential equations learn general representations', *Advances in neural information processing systems* **31**.

[40] Melas-Kyriazi, L., Rupprecht, C., Laina, I. and Vedaldi, A. [2023], 'Realfusion: 360 {\deg} reconstruction of any object from a single image', *arXiv preprint arXiv:2302.10663* .

[41] Misra, D. [n.d.], 'Mish: A self regularized non-monotonic activation function'.

[42] Ng, A. Y., Russell, S. et al. [2000], Algorithms for inverse reinforcement learning., *in* 'Icml', Vol. 1, p. 2.

[43] Nichol, A. Q. and Dhariwal, P. [2021], Improved denoising diffusion probabilistic models, *in* 'International Conference on Machine Learning', PMLR, pp. 8162–8171.

[44] Øksendal, B. and Øksendal, B. [2003], *Stochastic differential equations*, Springer.

[45] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. [2019], Pytorch: An imperative style, high-performance deep learning library, *in* 'Advances in Neural Information Processing Systems 32', Curran Associates, Inc., pp. 8024–8035.
URL: *http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf*

[46] Pearce, T., Rashid, T., Kanervisto, A., Bignell, D., Sun, M., Georgescu, R., Macua, S. V., Tan, S. Z., Momennejad, I., Hofmann, K. et al. [2023], 'Imitating human behaviour with diffusion models', *arXiv preprint arXiv:2301.10677* .

[47] Poole, B., Jain, A., Barron, J. T. and Mildenhall, B. [2023], Dreamfusion: Text-to-3d using 2d diffusion, *in* 'The Eleventh International Conference on Learning Representations'.
URL: *https://openreview.net/forum?id=FjNys5c7VyY*

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

[49] Raissi, M. [2018], 'Deep hidden physics models: Deep learning of nonlinear partial differential equations', *The Journal of Machine Learning Research* **19**(1), 932–955.

[50] Raissi, M., Perdikaris, P. and Karniadakis, G. E. [2019], 'Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations', *Journal of Computational physics* **378**, 686–707.

[51] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C. and Chen, M. [2022], 'Hierarchical text-conditional image generation with clip latents', *arXiv preprint arXiv:2204.06125* .

[52] Reuss, M., Li, M., Jia, X. and Lioutikov, R. [2023], 'Goal-conditioned imitation learning using score-based diffusion policies', *arXiv preprint arXiv:2304.02532* .

[53] Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B. [2022], High-resolution image synthesis with latent diffusion models, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 10684–10695.

[54] Russel, S. [n.d.], 'Human compatible: Artificial intelligence and the problem of control (great britain: Allen lane, 2019). ibo van de poel,"embedding values in artificial intelligence (ai) systems."', *Minds and Machines* **30**, 385–409.

[55] Russell, S. [1998], Learning agents for uncertain environments, *in* 'Proceedings of the eleventh annual conference on Computational learning theory', pp. 101–103.

[56] Russell, S. [2019], *Human compatible: Artificial intelligence and the problem of control*, Penguin.

[57] Särkkä, S. and Solin, A. [2019], *Applied stochastic differential equations*, Vol. 10, Cambridge University Press.

[58] Schramowski, P., Brack, M., Deiseroth, B. and Kersting, K. [2022], 'Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models', *arXiv preprint arXiv:2211.05105* .

[59] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. and Ganguli, S. [2015], Deep unsupervised learning using nonequilibrium thermodynamics, *in* 'International Conference on Machine Learning', PMLR, pp. 2256–2265.

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

[61] Song, Y. and Ermon, S. [2019], Generative modeling by estimating gradients of the data distribution, *in* 'Proceedings of the 33rd International Conference on Neural Information Processing Systems', pp. 11918–11930.

[62] Song, Y. and Ermon, S. [2020], 'Improved techniques for training score-based generative models', *Advances in neural information processing systems* **33**, 12438–12448.

[63] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S. and Poole, B. [n.d.], Score-based generative modeling through stochastic differential equations, *in* 'International Conference on Learning Representations'.

[64] Todorov, E. [2008], General duality between optimal control and estimation, *in* '2008 47th IEEE Conference on Decision and Control', IEEE, pp. 4286–4292.

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

[66] Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B. and Pavone, M. [2022], 'Guided conditional diffusion for controllable traffic simulation', *arXiv preprint arXiv:2210.17366* .

[67] Ziebart, B. D. [2010], *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*, Carnegie Mellon University.

[68] Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K. et al. [2008], Maximum entropy inverse reinforcement learning., *in* 'Aaai', Vol. 8, Chicago, IL, USA, pp. 1433–1438.

A. Short Appendix to Mehtods

A.1. Short background on Diffusion Models and Guidance

Diffusion Models in Continuous Time. At a high level, diffusion models work by adding noise to data $\mathbf{x} \in \mathbb{R}^n$ (*forward process*), and then learning to denoise it (*backward process*).

The forward noising process in continuous time follows the Stochastic Differential Equation (SDE):

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}_t \tag{6}$$

where f is a function that is Lipschitz, $\mathbf{w}$ is a standard Brownian Motion [35] and g is a (continuous) noise schedule, which regulates the amount of noise added to the data during the forward process (c.f. B.2). Song et al. [63] then use a result of Anderson [4] to write the SDE satisfied by the *reverse process* of (6), denoted $\bar{\mathbf{x}}_t$, as:

$$d\bar{\mathbf{x}}_t = \left[\mathbf{f}(\bar{\mathbf{x}}_t, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\bar{\mathbf{x}}_t)\right] dt + g(t)\, d\bar{\mathbf{w}}_t \tag{7}$$

Here $p_t(\mathbf{x})$ denotes the marginal density function of the forward process $\mathbf{x}_t$, and $\bar{\mathbf{w}}$ is a reverse Brownian motion (see [4]). The diffusion model is a neural network $\mathbf{s}_\Theta(\mathbf{x}_t, t)$ with parameters Θ that is trained to approximate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t)$, called the *score* of the distribution of $\mathbf{x}_t$. The network $\mathbf{s}_\Theta(\mathbf{x}_t, t)$ can then be used to generate new samples from p_0 by taking $\bar{\mathbf{x}}_T \sim \mathcal{N}(0, I)$ for some $T > 0$, and simulating (7) *backwards in time* to arrive at $\bar{\mathbf{x}}_0 \approx \mathbf{x}_0$, with the score term substituted for the neural network $\mathbf{s}_\Theta$:

$$d\bar{\mathbf{x}}_t = \left[\mathbf{f}(\bar{\mathbf{x}}_t, t) - g(t)^2 \mathbf{s}_\Theta(\bar{\mathbf{x}}_t, t)\right] dt + g(t)\, d\bar{\mathbf{w}}_t \tag{8}$$

This formulation is essential for deriving existence results for the relative reward function of two diffusion models in Section 4, as it allows for conditional sampling. For example, to sample from $p(\mathbf{x}_0|y) \propto p(\mathbf{x}_0) \cdot p(y|\mathbf{x}_0)$, where $y \in \{0, 1\}$, the sampling procedure can be modified to use $\nabla_{\mathbf{x}} \log p(\mathbf{x}_t|y) \approx \mathbf{s}_\Theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}} \rho(\mathbf{x}, t)$ instead of $\mathbf{s}_\Theta(\bar{\mathbf{x}}_t, t)$. Here, $\rho(\mathbf{x}, t)$ is a neural network approximating $\log p(y|\mathbf{x}_t)$. The gradients of ρ are multiplied by a small constant ω, called the *guidance scale*. The resulting *guided* reverse SDE is as follows:

$$d\bar{\mathbf{x}}_t = \left[\mathbf{f}(\bar{\mathbf{x}}_t, t) - g(t)^2 [\mathbf{s}_\Theta(\bar{\mathbf{x}}_t, t) + \omega \nabla_{\mathbf{x}} \rho(\bar{\mathbf{x}}_t, t)]\right] dt + g(t)\, d\bar{\mathbf{w}}_t \tag{9}$$

Informally, this method, also called *classifier guidance* [59], allows for steering a diffusion model to produce samples with some property y by gradually pushing the samples in the direction that maximizes the output of a classifier predicting $p(y|x)$.

A.2. A Result on Existence and Uniqueness

We now provide a result saying that, once $\mathbf{s}_\phi^{(1)}$ and $\mathbf{s}_\Theta^{(2)}$ are fixed, we obtain a condition on the gradients of ρ that, if met, would allow us not only to match the *distributions* of $\mathbf{s}_\phi^{(1)}$ and $\mathbf{s}_\Theta^{(2)}$, but also their entire denoising processes, with probability 1 (i.e. *almost surely*, a.s.).

In the theorem, $\mathbf{h}$ plays the role of the gradients $\nabla_{\mathbf{x}} \rho(\mathbf{x}_t, t)$. Going from time $t = 0$ to $t = T$ in the theorem corresponds to solving the backward SDE (7) from $t = T$ to $t = 0$, and $\mathbf{f}$ in the theorem corresponds to the drift term (i.e. coefficient of dt) of (7). For proof, see Appendix D.

Theorem A.1 (Existence and Uniqueness). *Let $T > 0$. Let $\mathbf{f}^{(1)}$ and $\mathbf{f}^{(2)}$ be functions from $\mathbb{R}^n \times [0, T]$ to $\mathbb{R}^n$ that are Lipschitz, and $g : [0, T] \to \mathbb{R}_{\geq 0}$ be bounded and continuous with $g(0) > 0$. Fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and a standard $\mathbb{R}^n$-Brownian Motion $(\mathbf{w}_t)_{t \geq 0}$.*

Consider the Itô SDEs:

$$d\mathbf{x}_t^{(1)} = \mathbf{f}^{(1)}(\mathbf{x}_t^{(1)}, t)\, dt + g(t)\, d\mathbf{w}_t \tag{10}$$

$$d\mathbf{x}_t^{(2)} = \mathbf{f}^{(2)}(\mathbf{x}_t^{(2)}, t)\, dt + g(t)\, d\mathbf{w}_t \tag{11}$$

$$d\mathbf{x}_t = \left[\mathbf{f}^{(1)}(\mathbf{x}_t, t) + \mathbf{h}(\mathbf{x}_t, t)\right] dt + g(t)\, d\mathbf{w}_t, \text{ where } \mathbf{h} \text{ is Lipschitz} \tag{12}$$

and fix an initial condition $\mathbf{x}_0^{(1)} = \mathbf{x}_0^{(2)} = \mathbf{x}_0 = \mathbf{z}$, where $\mathbf{z}$ is a random variable with $\mathbb{E}[||\mathbf{z}||_2^2] < \infty$.

Then (10), (11), and (12) have almost surely (a.s.) unique solutions $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$ and $\mathbf{x}$ with a.s. continuous sample paths. Furthermore, there exists an a.s. unique choice of $\mathbf{h}$ such that $\mathbf{x}_t = \mathbf{x}_t^{(2)}$ for all $t \geq 0$, a.s., which is given by

$$\mathbf{h}(\mathbf{x}, t) = \mathbf{f}^{(2)}(\mathbf{x}, t) - \mathbf{f}^{(1)}(\mathbf{x}, t). \tag{13}$$

In all diffusion model methods we are aware of, the drift and noise terms of the backward process indeed satisfy the pre-conditions of the theorem, under reasonable assumptions on the data distribution (see D.3), and using a network with smooth activation functions like Mish [41] or GeLU [18].

Therefore, Theorem A.1 tells us that, if we were free to pick the gradients $\nabla_{\mathbf{x}}\rho(\mathbf{x}_t, t)$, setting them to $\mathbf{f}^{(2)}(\mathbf{x}_t, t) - \mathbf{f}^{(1)}(\mathbf{x}_t, t)$ would be the "best" choice: it is the only choice resulting in guided samples *exactly reproducing* the whole process $\mathbf{x}^{(2)}$ (and, in particular, the distribution of $\mathbf{x}_0^{(2)}$). We will now see that, in an idealized setting, there exists a unique function ρ satisfying this criterion. We start by recalling the concept of a *conservative vector field*, from multivariate calculus, and how it relates to gradients of continuously differentiable functions.

Definition A.2 (Conservative Vector Field, Definition 7.6 in [34]). We say that a vector field $\mathbf{f}$ is conservative if it is the gradient of a continuously differentiable function Φ,

$$\mathbf{f}(\mathbf{x}) = \nabla_{\mathbf{x}}\Phi(\mathbf{x})$$

for all $\mathbf{x}$ in the domain of $\mathbf{f}$. The function Φ is called a potential function of $\mathbf{f}$.

Suppose we had access to the ground-truth scores $\mathbf{s}_{\text{true}}^{(1)}(\mathbf{x}, t)$ and $\mathbf{s}_{\text{true}}^{(2)}(\mathbf{x}, t)$ of the forward processes for the base and expert models (i.e. no approximation error). Then they are equal to $\nabla_{\mathbf{x}}\log p_t^{(1)}(\mathbf{x})$ and $\nabla_{\mathbf{x}}\log p_t^{(2)}(\mathbf{x})$, respectively. If we also assume $p_t^{(1)}(\mathbf{x})$ and $p_t^{(2)}(\mathbf{x})$ are continuously differentiable, we have that, by Definition A.2, the diffusion models are conservative for each t. Thus, their difference is also conservative, i.e. the gradient of a continuously differentiable function.

Hence, by the Fundamental Theorem for Line Integrals (Th. 7.2 in [34]), there exists a unique ρ satisfying $\nabla_{\mathbf{x}}\rho(\mathbf{x}, t) = \mathbf{s}_{\text{true}}^{(2)}(\mathbf{x}, t) - \mathbf{s}_{\text{true}}^{(1)}(\mathbf{x}, t)$, up to an additive constant, given by the line integral

$$\rho(\mathbf{x}, t) = \int_{\mathbf{x}_{\text{ref}}}^{\mathbf{x}} [\mathbf{s}_{\text{true}}^{(2)}(\mathbf{x}', t) - \mathbf{s}_{\text{true}}^{(1)}(\mathbf{x}', t)] \cdot d\mathbf{x}', \tag{14}$$

where $\mathbf{x}_{\text{ref}}$ is some arbitrary reference point, and the line integral is path-independent.

In practice, however, we cannot guarantee the absence of approximation errors, nor that the diffusion models are conservative.

B. More details on Diffusion Models

B.1. Diffusion Models in Continuous Time

At a high level, diffusion models work by adding noise to data $\mathbf{x}$, and then learning to denoise it.

The seminal paper on diffusion models [59] and the more recent work of Ho et al. [20] describe this in discrete time. Song et al. [63] show a direct correspondence between the aforementioned formulations of denoising diffusion models and an existing line of work on score-based generative modeling [61], and propose a continuous-time framework leveraging (Itô) Stochastic Differential Equations [35, 44, 57] to unify both methods. We give an overview of the continuous-time formulation, and defer to Appendix B.2 an account of the discrete case.

The forward noising process in continuous time is given by:

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}_t \tag{15}$$

where f is a function that is Lipschitz, $\mathbf{w}$ is a standard Brownian Motion [35] and g is a "continuous noise schedule" (c.f. B.2). A continuous version of the processes in B.2 [59, 20] is recovered by setting $g(s) = \sqrt{\beta_s}$ and $\mathbf{f}(\mathbf{x}, s) = -\frac{1}{2}\beta_s\mathbf{x}$. The resulting SDE resembles an Ornstein-Uhlenbeck process, which is known to converge in geometric rate to a $\mathcal{N}(0, I)$ distribution. This justifies the choice of distribution of $\mathbf{x}_T$ in B.2.

Song et al. [63] then use a result of Anderson [4] to write the SDE satisfied by the *reverse process* of 6, denoted $\bar{\mathbf{x}}_t$, as:

$$d\bar{\mathbf{x}}_t = [\mathbf{f}(\bar{\mathbf{x}}_t, t) - g(t)^2\nabla_{\mathbf{x}}\log p_t(\bar{\mathbf{x}}_t)] dt + g(t) d\bar{\mathbf{w}}_t \tag{16}$$

Here $p_t(\mathbf{x})$ denotes the marginal density function of the forward process $\mathbf{x}_t$, and $\bar{\mathbf{w}}$ is a reverse Brownian motion (for further details on the latter, see [4]). The learning step is then to learn an approximation $\mathbf{s}_\Theta(\mathbf{x}_t, t)$ to $\nabla_{\mathbf{x}}\log p_t(\mathbf{x}_t)$ (the *score* of the distribution of $\mathbf{x}_t$). This is shown by Song et al. [63] to be equivalent to the noise model $\boldsymbol{\epsilon}_\Theta$ in B.2. The sampling step consists of taking $\bar{\mathbf{x}}_T \sim \mathcal{N}(0, I)$ for some $T > 0$, and simulating 16 *backwards in time* to arrive at $\bar{\mathbf{x}}_0 \approx \mathbf{x}_0$.

One of the advantages of this perspective is that the sampling loop can be offloaded to standard SDE solvers, which can more flexibly trade-off e.g. computation time for performance, compared to hand-designed discretizations [31].

B.2. Discrete-Time Diffusion Models

The seminal paper on diffusion models [59] and the more recent work of Ho et al. [20] formulate diffusion models in discrete time. The forward noising is a Markov Chain starting from an uncorrupted data point $\mathbf{x}_0$, and repeatedly applying a (variance-preserving) Gaussian kernel to the data:

$$q_t(\mathbf{x}_{t+1}|\mathbf{x}_t) \sim \mathcal{N}(\sqrt{1 - \beta_t}\mathbf{x}_t, \beta_t I)$$
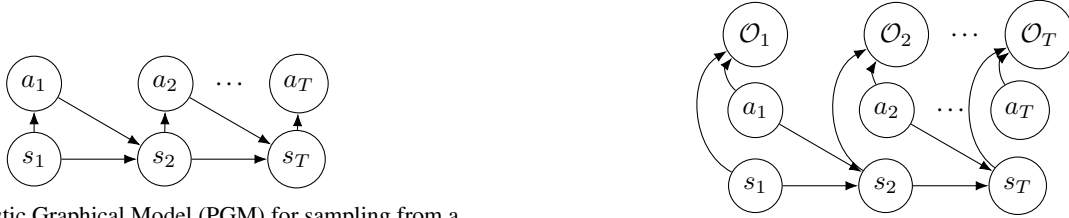
(a) Probabilistic Graphical Model (PGM) for sampling from a Markov Decision Process (MDP) with a fixed policy π.

(b) PGM for planning, with optimality variables $\mathcal{O}_{1:T}$.

Figure 5: Reinforcement Learning as Probabilistic Inference (see App. C for a detailed explanation).

The coefficients β_t are known as the *noise schedule* of the diffusion model, and can be chosen to regulate the speed with which noise is added to the data [31].

The modeling step then aims to *revert* the noising process by learning a backward kernel

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2) \tag{17}$$

The seminal paper of Sohl-Dickstein et al. [59] proposes a variational loss for training μ_θ, based on a maximum-likelihood framework. Subsequently, Ho et al. [20] proposed a new parametrization of μ_θ, as well as a much simpler loss function. Their loss function is

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right] \tag{18}$$

where $\alpha_t := 1 - \beta_t$, $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$ and $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$. Intuitively, this loss says that $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ should fit the noise $\boldsymbol{\epsilon}$ added to the initial datapoint $\mathbf{x}_0$ up to time t. For a full account of this derivation, we refer the reader to Ho et al. [20].

B.2.1. CLASSIFIER GUIDANCE

Sohl-Dickstein et al. [59] also propose a method for *steering* the sampling process of diffusion models. Intuitively, if the original diffusion model computes a distribution $p(\mathbf{x}_0)$, we would like to make it compute $p(\mathbf{x}_0|y)$, where we assume, for the sake of concreteness, that $y \in \{0, 1\}$. Applying Bayes's rule gives the factorization $p(\mathbf{x}_0|y) \propto p(\mathbf{x}_0) \cdot p(y|\mathbf{x}_0)$. The first term is the original distribution the diffusion model is trained on. The second term is a probabilistic classifier, inferring the probability that $y = 1$ for a given sample $\mathbf{x}_0$.

We now leverage the framework in B.1 to derive the classifier guidance procedure. Equation 16 shows we can sample from the backward process by modeling $\nabla_\mathbf{x}\log p_t(\mathbf{x}_t)$ using $\mathbf{s}_\Theta(\mathbf{x}_t, t)$. Thus, to model $p(\mathbf{x}_0|y)$, we may instead approximate $\nabla_\mathbf{x}\log p(\mathbf{x}_t|y)$ using:

$$\nabla_\mathbf{x}\log p(\mathbf{x}_t|y) = \nabla_\mathbf{x}\log \frac{p(\mathbf{x}_t)\,p(y|\mathbf{x}_t)}{p(y)} \tag{19}$$

$$= \nabla_\mathbf{x}\log p(\mathbf{x}_t) + \nabla_\mathbf{x}\log p(y|\mathbf{x}_t) \tag{20}$$

$$\approx \mathbf{s}_\Theta(\mathbf{x}_t, t) + \nabla_\mathbf{x}\rho(\mathbf{x}_t, t) \tag{21}$$

where $\rho(\mathbf{x}, t)$ is a neural network approximating $\log p(y|\mathbf{x}_t)$. In practice, we multiply the gradients of ρ by a small constant ω, called the *guidance scale*.

The *guided* reverse SDE hence becomes:

$$d\bar{\mathbf{x}}_t = \left[\mathbf{f}(\bar{\mathbf{x}}_t, t) - g(t)^2[\mathbf{s}_\Theta(\bar{\mathbf{x}}_t, t) + \omega\nabla_\mathbf{x}\rho(\bar{\mathbf{x}}_t, t)]\right] dt + g(t)\,d\bar{\mathbf{w}}_t \tag{22}$$

The main takeaway is, informally, that we can steer a diffusion model to produce samples with some property y by gradually pushing the samples in the direction that maximizes the output of a classifier predicting $p(y|\mathbf{x})$.

C. More details on Reinforcement Learning as Probabilistic Inference

Levine [36] provides an in-depth exposition on existing methods for approaching sequential decision-making from a probabilistic and causal angle, referencing the work of e.g. [64, 67, 68, 30, 29]. We refer to it as the Control as Inference Framework (CIF).

The starting point for the formulation is the classical notion of a Markov Decision Problem (MDP), a stochastic process given by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a (possibly stochastic) transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_\mathcal{S}$, and a reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathbb{R}_{\geq 0}}$. Here $\Delta_\mathcal{X}$ denotes the set of distributions over a set $\mathcal{X}$. In this context, a policy $\pi : \mathcal{S} \rightarrow \Delta_\mathcal{A}$ represents a (possibly stochastic) way of choosing an action given the current state.

The MDP starts at an initial state $s_0 \in \mathcal{S}$, and evolves by sampling $a_t \sim \pi(s_t)$, and then $s_{t+1} \sim \mathcal{T}(s_t, a_t)$ for $t \geq 0$. The reward received at time t is $r_t \sim r(s_t, a_t)$. We consider here an episodic setting, where the MDP stops at a fixed time $T > 0$. We call the sequence $((s_t, a_t))_{t=0}^T$ a *trajectory*. Sampling corresponds to the Probabilistic Graphical Model (PGM) in 5a, where each action depends on the current state, and the next state depends on the current state and the current action.

To encode the process of *choosing a policy to optimize a reward*, this PGM can be extended with *optimality variables*, as per Fig. 5b. They are defined as $\mathcal{O}_t \sim \mathrm{Ber}(e^{-r(s_t, a_t)})$, so that their *distribution* encodes the dependency of reward on current states and actions. The problem of optimizing the reward can then be recast as *sampling from* $p(\tau|\mathcal{O}_{1:T})$. Following [36], one can apply Bayes's Rule to obtain

$$p(\tau|\mathcal{O}_{1:T}) \propto p(\tau) \cdot p(\mathcal{O}_{1:T}|\tau) \tag{23}$$

which factorizes the distribution of optimal trajectories (up to a normalizing constant) as a *prior* $p(\tau)$ over trajectories and a *likelihood term* $p(\mathcal{O}_{1:T}|\tau)$. Observe that, from the definition of $\mathcal{O}_{1:T}$ and the PGM factorization in Fig. 5b, we have $p(\mathcal{O}_{1:T}|\tau) = e^{-\sum_t r(s_t, a_t)}$. Hence, in the CIF, $-\log p(\mathcal{O}_{1:T}|\tau)$ (a *negative log likelihood*) corresponds to the cumulative reward of a trajectory τ.

D. Details on Main Derivations

D.1. Existence and Uniqueness of Strong Solutions to Stochastic Differential Equations

Theorem D.1 (Existence and uniqueness theorem for SDEs, c.f. page 66 of Øksendal and Øksendal [44]). *Let $T > 0$ and $b(\cdot, \cdot)$: $[0, T] \times \mathbb{R}^n \to \mathbb{R}^n, \sigma(\cdot, \cdot) : [0, T] \times \mathbb{R}^n \to \mathbb{R}^{n \times m}$ be measurable functions satisfying*

$$|b(t, x)| + |\sigma(t, x)| \leq C(1 + |x|); \quad x \in \mathbb{R}^n, t \in [0, T]$$

for some constant C, where $|\sigma|^2 = \sum |\sigma_{ij}|^2$, and such that

$$|b(t, x) - b(t, y)| + |\sigma(t, x) - \sigma(t, y)| \leq D|x - y|; \quad x, y \in \mathbb{R}^n, t \in [0, T]$$

for some constant D. Let Z be a random variable which is independent of the σ-algebra $\mathcal{F}_\infty^{(m)}$ generated by $B_s(\cdot), s \geq 0$ and such that

$$\mathbb{E}\left[|Z|^2\right] < \infty$$

Then the stochastic differential equation

$$dX_t = b(t, X_t)\, dt + \sigma(t, X_t)\, dB_t, \quad 0 \leq t \leq T, X_0 = Z$$

has a unique t-continuous solution $X_t(\omega)$ with the property that $X_t(\omega)$ is adapted to the filtration $\mathcal{F}_t^Z$ generated by Z and $B_s(\cdot); s \leq t$ and

$$\mathbb{E}\left[\int_0^T |X_t|^2\, dt\right] < \infty.$$

Remark D.2. In the above, the symbol $|\cdot|$ is overloaded, and taken to mean the *norm* of its argument. As everything in sight is finite-dimensional, and as all norms in finite-dimensional normed vector spaces are topologically equivalent (Theorem 2.4-5 in Kreyszig [33]), the stated growth conditions do not depend on the particular norm chosen for $\mathbb{R}^n$.

Also, whenever we talk about a *solution to an SDE* with respect to e.g. the standard Brownian motion $\mathbf{w}$ and an initial condition $\mathbf{z}$, we are always referring only to $(\mathcal{F}_t^{\mathbf{z}})_{t \geq 0}$-adapted stochastic processes, as per Theorem D.1.

Corollary D.3. *Fix a Brownian Motion $\mathbf{w}$ and let $(\mathcal{F}_t)_{t \geq 0}$ be its natural filtration. Consider an SDE $d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t) + g(t)d\mathbf{w}_t$ with initial condition $\mathbf{z} \in \mathcal{L}^2$ independent of $\mathcal{F}_\infty = \cup_{t \geq 0}\mathcal{F}_t$. Suppose $\mathbf{f} : \mathbb{R}^n \times [0, T] \to \mathbb{R}^n$ is Lipschitz with respect to $(\mathbb{R}^n \times [0, T], \mathbb{R}^n)$ and $g : [0, T] \to \mathbb{R}_{\geq 0}$ is continuous and bounded.*

Then the conclusion of D.1 holds and there is an a.s. unique $(\mathcal{F}_t^{\mathbf{z}})_{t \geq 0}$-adapted solution $(\mathbf{x}_t)_{t \geq 0}$ having a.s. continuous paths, where $(\mathcal{F}_t^{\mathbf{z}})_{t \geq 0}$ is defined as in D.1.

Proof. Firstly, note that, as $\mathbf{f}$ and g are continuous, they are Lebesgue-measurable. It remains to check the growth conditions in D.1.

As $\mathbf{f}$ is Lipschitz, there exists C_1 (w.l.o.g. $C_1 > 1$) such that, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $t, s \in [0, T]$:

$$|\mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\mathbf{y}, s)| \leq C_1(|\mathbf{x} - \mathbf{y}| + |t - s|) \tag{24}$$
$$\leq C_1(|\mathbf{x} - \mathbf{y}| + T) \tag{25}$$

In particular, taking $\mathbf{y} = \mathbf{0}$ and $s = 0$, we obtain that $|\mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\mathbf{0}, 0)| \leq C_1(|\mathbf{x}| + T)$. Let $M = |\mathbf{f}(\mathbf{0}, 0)|/C_1$. Then, by the triangle inequality, $|\mathbf{f}(\mathbf{x}, t)| \leq C_1(|\mathbf{x}| + T + M)$.

As g is bounded, there exists $C_2 > 0$ such that $|g(t)| \leq C_2$, and so for any $t, s \in [0, T]$, we have:

$$|g(t) - g(s)| \leq 2C_2 \tag{26}$$

Hence we have, for $t \in [0, T]$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$:

$$|\mathbf{f}(\mathbf{x}, t)| + |g(t)| \leq ((T + M)C_1 + C_2)(1 + |\mathbf{x}|) \qquad\qquad |\mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\mathbf{y}, t)| \leq C_1 |\mathbf{x} - \mathbf{y}| \tag{27}$$

which yields the growth conditions required in D.1, and the conclusion follows. $\qquad\square$

D.2. Proof of Theorem A.1

Proof of Theorem A.1. We reference the fundamental result on the existence and uniqueness of strong solutions to SDEs, presented in Section 5.2.1 of [44] and reproduced above. In particular, Corollary D.3 shows that the Existence and Uniqueness Theorem D.1 applies for SDEs 10 and 11, by our assumptions on $\mathbf{f}^{(1)}$, $\mathbf{f}^{(2)}$ and g. As also we restrict our attention to $\mathbf{h}$ that are Lipschitz, and sums of Lipschitz functions are also Lipschitz, the corollary also applies to SDE 12. This establishes the existence of a.s. unique (adapted) solutions with a.s. continuous sample paths for these SDEs. Call them $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$ and $\mathbf{x}$, respectively.

Denote $\mathbf{h}_1(\mathbf{x}, t) = \mathbf{f}^{(2)}(\mathbf{x}, t) - \mathbf{f}^{(1)}(\mathbf{x}, t)$. The choice $\mathbf{h} = \mathbf{h}_1$ makes SDE 12 have the same drift and noise coefficients as SDE 11, and so, by the uniqueness of solutions we established, it follows that $\mathbf{x}_t^{(2)} = \mathbf{x}_t$ for all t a.s., with this choice of $\mathbf{h}$.

Now suppose we have a Lipschitz-continuous $\tilde{\mathbf{h}}$ which yields an a.s. unique, t-continuous solution $(\tilde{\mathbf{x}}_t)_{t \geq 0}$ that is indistinguishable from $\mathbf{x}^{(2)}$. (i.e. equal to $\mathbf{x}^{(2)}$ for all t, a.s.). We show that $\tilde{\mathbf{h}} = \mathbf{h}_1$.

As $\tilde{\mathbf{x}}$ and $\mathbf{x}^{(2)}$ are indistinguishable and satisfy SDEs with the same noise coefficient, we obtain a.s., for any $t \in [0, T]$:

$$0 = \tilde{\mathbf{x}}_t - \mathbf{x}_t^{(2)} \tag{28}$$

$$= \left(\int_0^t \mathbf{f}^{(1)}(\tilde{\mathbf{x}}_s, s) + \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_s, s)ds + \int_0^t g(s)\, d\mathbf{w}_s \right) - \left(\int_0^t \mathbf{f}^{(2)}(\mathbf{x}_s^{(2)}, s)ds + \int_0^t g(s)\, d\mathbf{w}_s \right) \tag{29}$$

$$= \int_0^t \mathbf{f}^{(1)}(\tilde{\mathbf{x}}_s, s) + \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_s, s)ds - \int_0^t \mathbf{f}^{(2)}(\mathbf{x}_s^{(2)}, s)ds \tag{30}$$

$$= \int_0^t \mathbf{f}^{(1)}(\mathbf{x}_s^{(2)}, s) + \tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s)ds - \int_0^t \mathbf{f}^{(2)}(\mathbf{x}_s^{(2)}, s)ds \tag{31}$$

$$= \int_0^t \tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s) - (\mathbf{f}^{(2)}(\mathbf{x}_s^{(2)}, s) - \mathbf{f}^{(1)}(\mathbf{x}_s^{(2)}, s))ds \tag{32}$$

$$= \int_0^t \tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s) - \mathbf{h}_1(\mathbf{x}_s^{(2)}, s)ds \tag{33}$$

where in 29 we substitute the integral forms of the SDEs satisfied by $\tilde{\mathbf{x}}$ and $\mathbf{x}^{(2)}$, and in 31 we use that the processes are indistinguishable to replace $\tilde{\mathbf{x}}$ by $\mathbf{x}^{(2)}$ in one of the integrals.

Hence, with probability 1, $\int_0^t \tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s)ds = \int_0^t \mathbf{h}_1(\mathbf{x}_s^{(2)}, s)ds$. As $\mathbf{x}^{(2)}$ is a.s. continuous, we also have that the mappings $s \mapsto \mathbf{h}_1(\mathbf{x}_s^{(2)}, s)$ and $s \mapsto \tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s)$ are continuous with probability 1. We may hence apply the Fundamental Theorem of Calculus and differentiate 33 to conclude that a.s. $\tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s) = \mathbf{h}_1(\mathbf{x}_s^{(2)}, s)$ for all s.

As $g(0) > 0$, $\mathbf{x}_t^{(2)}$ is supported on all of $\mathbb{R}^n$ for any $t > 0$. Therefore, the above implies that $\tilde{\mathbf{h}}(\mathbf{x}, s) = \mathbf{h}_1(\mathbf{x}, s)$ for all $s \in (0, T]$ *and* all $\mathbf{x} \in \mathbb{R}^n$.

More formally, let $\Delta(\mathbf{x}', s) = \tilde{\mathbf{h}}(\mathbf{x}', s) - \mathbf{h}_1(\mathbf{x}', s)$. Then $\Delta(\mathbf{x}_s^{(2)}, s) = 0$ for all s a.s.. Also, Δ is Lipschitz. Let C be its Lipschitz constant. Take $\mathbf{x} \in \mathbb{R}^n$ and fix $\varepsilon > 0$ and $t > 0$. Then $\mathbb{P}(||\mathbf{x}_t^{(2)} - \mathbf{x}||_2 < \frac{\varepsilon}{2C}) > 0$, as $\mathbf{x}_t^{(2)}$ is supported in all of $\mathbb{R}^n$. Hence, by the above, there exists $\mathbf{y}_\varepsilon \in B(\mathbf{x}, \frac{\varepsilon}{2C})$ such that $\Delta(\mathbf{y}_\varepsilon, t) = 0$. As $\tilde{\mathbf{h}}$ is Lipschitz, for any $\mathbf{x}' \in B(\mathbf{x}, \frac{\varepsilon}{2C})$, we have that

$$||\Delta(\mathbf{x}', t)|| \leq C||\mathbf{x}' - \mathbf{y}_\varepsilon|| \tag{34}$$

$$\leq C(||\mathbf{x}' - \mathbf{x}|| + ||\mathbf{x} - \mathbf{y}_\varepsilon||) \tag{35}$$

$$\leq C\left(\frac{\varepsilon}{2C} + \frac{\varepsilon}{2C}\right) \tag{36}$$

$$= \varepsilon \tag{37}$$

As ε was arbitrary, we have that $\Delta(\mathbf{x}', t) \to 0$ as $\mathbf{x}' \to \mathbf{x}$. As Δ is continuous (since it is Lipschitz), it follows that $\Delta(\mathbf{x}, t) = 0$. As $\mathbf{x}$ was also arbitrary, we have that $\tilde{\mathbf{h}}(\mathbf{x}, s) = \mathbf{h}_1(\mathbf{x}, s)$ for all $s \in (0, T]$ *and* all $\mathbf{x} \in \mathbb{R}^n$.

As $\mathbf{h}$ must be chosen to be (Lipschitz) continuous also with respect to t, for any $\mathbf{x}$ it must be that

$$\tilde{\mathbf{h}}(\mathbf{x}, 0) = \lim_{s \to 0^+} \tilde{\mathbf{h}}(\mathbf{x}, s) = \lim_{s \to 0^+} \mathbf{h}(\mathbf{x}, s) = \mathbf{h}_1(\mathbf{x}, 0) \tag{38}$$

This completes the proof of the uniqueness of the choice $\mathbf{h} = \mathbf{h}_1$. $\qquad\square$

D.3. Does the Backward Process of a Diffusion Model Fit in Theorem D.1?

In the forward process 6, $\mathbf{f}$ is taken to be Lipschitz, and g is always bounded, e.g. in [20, 60, 63]. However, in the backward process, one may be concerned about whether the score term $-\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t, t)$ is Lipschitz. We argue that it is reasonable to assume that it is.

We consider a setting where $g(t) = 1$, and $\mathbf{f}(\mathbf{x}, t) = -\lambda \mathbf{x}$. Then the forward process is the well-understood Ornstein-Uhlenbeck (or Langevin) process [44, p. 74]:

$$d\mathbf{x}_t = -\lambda \mathbf{x}_t \, dt + \, d\mathbf{w}_t \tag{39}$$

Consider the simplified case where the initial condition is deterministic: $\mathbf{x}_0 = \mathbf{y} \in \mathbb{R}^n$. Then $\mathbf{x}$ is a Gaussian process of co-variance function given by $K(s, t) = \text{cov}(X_s, X_t) = \frac{e^{-\lambda|t-s|} - e^{-\lambda(t+s)}}{2\lambda} I$, where I is the identity matrix, as per Le Gall [35], page 226. Its mean function is $\mathbf{m}_t = \mathbf{y} e^{-\lambda t}$. In particular, it has variance $\sigma_t^2 = \frac{1 - e^{-2\lambda t}}{2\lambda}$, and $\sigma_t^2 \to \frac{1}{2\lambda}$ as $t \to \infty$.

We follow the theoretical analysis of the convergence of diffusion models by De Bortoli [9] and assume the backward diffusion process finishes at a time $\varepsilon > 0$. Hence, the backward process $\bar{\mathbf{x}}_t$ is now supported in all of $\mathbb{R}^n$ for any $t \in [\epsilon, T]$. The original data need not be supported everywhere, and in fact is often assumed to not be, c.f. the manifold hypothesis in [9]. In our simplified case, it is supported at a point. This is why $t = 0$ needs to be excluded by the backward process finishing at some $\varepsilon > 0$.

From the aforementioned properties of the Ornstein-Uhlenbeck process, $p_t(\mathbf{x}_t) = C \exp\left(-\frac{||\mathbf{x}_t - \mathbf{m}_t||_2^2}{2\sigma_t^2}\right)$, where C is a constant independent of $\mathbf{x}$. But then $-\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t, t) = \frac{\mathbf{x}_t - \mathbf{m}_t}{\sigma_t^2}$. For any fixed t, we hence have that $-\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t, t)$ is $1/\sigma_t^2$-Lipschitz. Since the variance σ_t is bounded away from 0 for $t \in [\varepsilon, T]$, we may pick a Lipschitz constant that works uniformly across all $\mathbf{x}$ and t (for instance, $1/\sigma_\varepsilon^2$).

This can be extended to the case where the initial data is bounded, which is frequently assumed (e.g. by De Bortoli [9]). Informally, if the data is bounded, it cannot change the *tails* of the distribution of $\mathbf{x}_t$ significantly. We omit further details, as this discussion is beyond the scope of the paper.

In summary, the forward noising process is very similar to an Ornstein-Uhlenbeck process, so, for practical purposes, it is reasonable to assume the scores $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t, t)$ are Lipschitz.

D.4. Existence and Uniqueness of Minimizer of (2)

Definition D.4. $L^2(\mathbb{R}^n, \mathbb{R}^n)$ is the Hilbert space given by

$$L^2(\mathbb{R}^n, \mathbb{R}^n) = \left\{ \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n : \int_{\mathbb{R}^n} ||\mathbf{f}(\mathbf{x})||_2^2 d\mathbf{x} < \infty \right\} \tag{40}$$

Remark D.5. It is easy to show $L^2(\mathbb{R}^n, \mathbb{R}^n)$ is a Hilbert Space with norm given by

$$||\mathbf{f}||_2^2 = \int_{\mathbb{R}^n} ||\mathbf{f}(\mathbf{x})||_2^2 \, d\mathbf{x} \tag{41}$$

For instance, one can start from the standard result that $L^2(\mathbb{R}^n, \mathbb{R})$ is a Hilbert Space, and apply it to each coordinate of the output of $\mathbf{f}$.

Definition D.6. Denote by $\text{Cons}(\mathbb{R}^n)$ the space of the gradients of smooth, square-integrable potentials from $\mathbb{R}^n$ to $\mathbb{R}$ with square-integrable gradient, i.e.

$$\text{Cons}(\mathbb{R}^n) = \left\{ \nabla f : f \text{ is smooth}, \int_{\mathbb{R}^n} |f(\mathbf{x})|^2 d\mathbf{x} < \infty \text{ and } \int_{\mathbb{R}^n} ||\nabla f(\mathbf{x})||_2^2 d\mathbf{x} < \infty \right\} \tag{42}$$

Remark D.7. Clearly $\text{Cons}(\mathbb{R}^n) \subseteq L^2(\mathbb{R}^n, \mathbb{R}^n)$. The condition that "$f$ is square-integrable and has a square-integrable gradient" corresponds to f being in the Sobolev space $W^{1,2}(\mathbb{R}^n)$ (see Jost [28], Chapter 9).

Denote by $\overline{\text{Cons}(\mathbb{R}^n)}$ the **closure** of $\text{Cons}(\mathbb{R}^n)$, i.e.

$$\overline{\text{Cons}(\mathbb{R}^n)} = \{ \mathbf{f} : \text{there exists } (\nabla f_k)_{k \geq 0} \subseteq \text{Cons}(\mathbb{R}^n) \text{ such that } ||\nabla f_k - \mathbf{f}||_2^2 \to 0 \text{ as } k \to \infty \} \tag{43}$$

By construction, $\overline{\text{Cons}(\mathbb{R}^n)}$ is a vector subspace of $L^2(\mathbb{R}^n, \mathbb{R}^n)$, and is closed, i.e. stable under taking limits.

Theorem D.8 (Complementation in Hilbert Spaces, c.f. Kreyszig [33], Theorem 3.3-4)**.** *Let* $(\mathcal{H}, \langle \cdot, \cdot \rangle, ||\cdot||)$ *be a Hilbert space, and let* $Y \subseteq H$ *be a closed vector subspace. Then any* $x \in \mathcal{H}$ *can be written as* $x = y + z$, *where* $y \in Y$ *and* $z \in Y^{\perp}$.

Corollary D.9 (Uniqueness of Projection)**.** *Then the minimum of* $v \mapsto ||v - x||$ *over* $v \in Y$ *is attained at the (unique)* y *given in the theorem, as* $||v - x||^2 \geq |\langle v - x, z \rangle| = ||z||^2$, *and setting* $v = y$ *attains this bound.*

Proof of Proposition 4.1. Firstly note that we assumed $\mathbf{s}_{\Theta}^{(2)}(\cdot, t) - \mathbf{s}_{\phi}^{(1)}(\cdot, t)$ is in $L^2(\mathbb{R}^n, \mathbb{R}^n)$ for each individual t.

It follows directly from the above that, as $\overline{\mathrm{Cons}(\mathbb{R}^n)}$ is a closed subspace of $L^2(\mathbb{R}^n, \mathbb{R}^n)$, Corollary D.9 applies, and we have that there is a unique minimizer $\mathbf{h}_t \in \overline{\mathrm{Cons}(\mathbb{R}^n)}$ in Equation 2.

As $\overline{\mathrm{Cons}(\mathbb{R}^n)}$ consists of $L^2(\mathbb{R}^n, \mathbb{R}^n)$ limits of sequences in $\mathrm{Cons}(\mathbb{R}^n)$, there exists a sequence $(\nabla \Phi_k)_{k \geq 0}$ of gradients of $W^{1,2}$-potentials such that $||\nabla \Phi_k - \mathbf{h}_t||_2^2 \to 0$ as $k \to \infty$. From the definition of convergence, we get that, for any $\varepsilon > 0$, there is k large enough such that

$$\int_{\mathbb{R}^n} ||\nabla_{\mathbf{x}} \Phi_k(\mathbf{x}) - \mathbf{h}_t(\mathbf{x})||_2^2 \, d\mathbf{x} < \varepsilon \tag{44}$$

which completes the proof. $\square$

E. Relative Reward Learning Algorithms

The below algorithm was used for learning the relative reward functions for experiments with Stable Diffusion (see Section 5.3).

Algorithm 2: Relative reward function training with access only to diffusion models

---
Input: Base diffusion model $\mathbf{s}^{(1)}$ and expert diffusion model $\mathbf{s}^{(2)}$.
Output: Relative reward estimator ρ_θ.
```
// Dataset pre-generation using the diffusion models
```
$\mathcal{D}_1 \leftarrow \emptyset$
$\mathcal{D}_2 \leftarrow \emptyset$
for $m \in \{1, 2\}$*, K times* **do**
    $\mathbf{x}_T \sim \mathcal{N}(0, I)$
    for $t \in T - 1, ..., 0$ **do**
        $\mathbf{x}_t^{(1)} \leftarrow \mathbf{x}_{t+1}$ denoised by 1 more step using $\mathbf{s}^{(1)}$
        $\mathbf{x}_t^{(2)} \leftarrow \mathbf{x}_{t+1}$ denoised by 1 more step using $\mathbf{s}^{(2)}$
        $\mathbf{x}_t \leftarrow \mathbf{x}_t^{(m)}$
        Add $(t + 1, \mathbf{x}_{t+1}, \mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)})$ to $\mathcal{D}_m$
    end
end

```
// Training
```
$\mathcal{D} \leftarrow \mathcal{D}_1 \cup \mathcal{D}_2$
Initialize parameters θ
for $i \in 1...n\_train\_steps$ **do**
    ```// Using batch size of 1 for clarity```
    Sample $(t + 1, \mathbf{x}_{t+1}, \mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)})$ from $\mathcal{D}$
    ```// use pre-computed diffusion outputs to compute loss```
    $\hat{L}_{\mathrm{RRF}}(\theta) \leftarrow ||\nabla_{\mathbf{x}} \rho_\theta(\mathbf{x}_{t+1}, t + 1) - (\mathbf{x}_t^{(2)} - \mathbf{x}_t^{(1)})||_2^2$
    Take an Adam [32] optimization step on θ according to $\hat{L}_{\mathrm{IRL}}(\theta)$
end

Remark E.1. In our experiments with Stable Diffusion, the generation of the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ uses a *prompt dataset* for generating the images. For each prompt, we generate one image following the base model ($m = 1$), and one following the expert model ($m = 2$). The I2P prompt dataset [58] contains 4703 prompts.

F. Implementation details

Relative reward function architectures. The architectures and dimensions for all experiments are given in Table 2. In the Locomotion environments and in Stable Diffusion, we parameterize the relative reward functions using the encoder of a UNet, following the architecture

of Janner et al. [24] for parameterizing their value function networks. In the Maze environments, we use an MLP taking in a sequence of h state-action pairs. In the following, the *dimensions* refer to the number of channels after each downsampling step in the case of UNet architectures, and to layer dimensions in the case of MLP architectures. The *horizon* refers to the number of state-action pairs input to the reward function network. When it does not match the diffusion horizon H, we split trajectories into H/h segments, feed each into the reward network, and average the results.

Relative reward function training hyperparameters. In Table 3 we indicate the learning rate and batch size used for training the reward functions, as well as the number of denoising timesteps of the diffusion models they are trained against. We report the number of training steps for the models used to generate the plots and numerical results in the main paper. We use Adam [45] as an optimizer, without weight decay. The Diffusion models use ancestral sampling as in [24, 20], with a cosine noise schedule [43].

Conditional generation hyperparameters. In Locomotion environments, we find that the performance of the steered model is affected by the guidance scale ω and by `t_stopgrad`, a hyperparameter specifying the step after which we *stop* adding reward gradients to the diffusion model sample. For example, if `t_stopgrad = 5`, we only steer the generation up to timestep 5, and for steps 4 down to 0 no more steering is used.

In Tables 4 and 5 we indicate the respective values used for the results reported in Table 1 (see *Discriminator* and *Reward (Ours)*), as well as for the results presented in the later presented ablation study in Appendix G.0.2 (see *Reward (Ours, Ablation)*).

Using estimates of $\mathbb{E}[\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t]$ for gradient alignment. Diffusion models can be trained to output score estimates $\mathbf{s}_\theta(x_t, t) \approx \nabla_{\mathbf{x}}\log p_t(\mathbf{x}_t)$, but also estimates of a denoised sample: $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{s}_\theta(\mathbf{x}_t, t)\right) \approx \mathbb{E}[\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t]$.

For two diffusion models $\mathbf{s}_\phi^{(1)}$ and $\mathbf{s}_\Theta^{(2)}$, we have that $\boldsymbol{\mu}_\phi(\mathbf{x}_t, t) - \boldsymbol{\mu}_\Theta(\mathbf{x}_t, t) \propto \mathbf{s}_\Theta^{(2)}(\mathbf{x}_t, t) - \mathbf{s}_\phi^{(1)}(\mathbf{x}_t, t)$ (notice that the sign gets flipped). However, we find that aligning the reward function gradients to the difference in means $\boldsymbol{\mu}_\phi(\mathbf{x}_t, t) - \boldsymbol{\mu}_\Theta(\mathbf{x}_t, t)$ (instead of the difference in scores) leads to slightly better performance. Hence, in practice, we use the following reward function:

$$L_{\text{RRF}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,T], \mathbf{x}_t \sim p_t}\left[||\nabla_{\mathbf{x}}\rho_\theta(\mathbf{x}_t, t) - (\boldsymbol{\mu}_\phi(\mathbf{x}_t, t) - \boldsymbol{\mu}_\Theta(\mathbf{x}_t, t))||_2^2\right] \tag{45}$$

Notice that, since during sampling we multiply the outputs of ρ by a guidance scale $\omega > 0$ of our choice, this reward function is equivalent to the one in the main paper, up to a scale factor.

Trajectory normalization. Following the implementation of Janner et al. [24], we use min-max normalization for trajectories in Maze and Locomotion environments. The minima and maxima are computed for each coordinate of states and actions, across all transitions in the dataset. Importantly, they are computed without any noise being added to the transitions. For our experiments with Stable Diffusion, we use standard scaling (i.e. making the data be centered at $\mathbf{0}$ and have variance 1 on each coordinate). Since in this case a dataset is pre-generated containing all intermediate diffusion timesteps, the means and standard deviations are computed separately for each diffusion timestep, and for each coordinate of the CLIP latents.

Reward function horizon vs. Diffuser horizon. We consider different ways of picking the number of timesteps fed into our network for reward prediction. Note that the horizon of the reward model network must be compatible with the length of the Diffuser trajectories (256). To account for this, we choose horizons h that are powers of two and split a given diffusor trajectory of length 256 into $256/h$ sub-trajectories, for which we compute the reward separately. We then average the results of all windows (instead of summing them, so that the final numerical value does not scale inversely with the chosen horizon). The reward horizon h used for different experiments is indicated in Table 2.

Visualization details. To obtain visualizations of the reward function that are intuitively interpretable, we modified the D4RL [12] dataset in Maze2D to use start and goal positions uniformly distributed in space, as opposed to only being supported at integer coordinates. Note that we found that this modification had no influence on performance, but was only made to achieve visually interpretable reward functions (as they are shown in Figure 2).

Quantitative evaluation in Maze2D. We quantitatively evaluated the learned reward functions in Maze2D by comparing the position of the global maximum of the learned reward function to the true goal position. We first smoothed the reward functions slightly to remove artifacts and verified whether the position of the global reward maximum was within 1 grid cell of the correct goal position. We allow for

Table 2: Model architectures and dimensions of the relative reward functions.

Environment	Architecture	Dimensions	Diffusion Horizon	Reward Horizon
`maze2d-open-v0`	MLP	(32,)	256	64
`maze2d-umaze-v0`	MLP	(128,64,32)	256	4
`maze2d-medium-v1`	MLP	(128,64,32)	256	4
`maze2d-large-v1`	MLP	(128,64,32)	256	4
Halfcheetah	UNet Encoder	(64,64,128,128)	4	4
Hopper	UNet Encoder	(64,64,128,128)	32	32
Walker2D	UNet Encoder	(64,64,128,128)	32	32
Stable Diffusion	UNet Encoder	(32,64,128,256)	-	-

Table 3: Parameters for the training of the relative reward functions.

Environment	Learning Rate	Batch Size	Diffusion Timesteps	Training Steps
Maze	$5 \cdot 10^{-5}$	256	100	100000
Locomotion	$2 \cdot 10^{-4}$	128	20	50000
Stable Diffusion	$1 \cdot 10^{-4}$	64	50	6000

Table 4: Parameters for sampling (rollouts) in Locomotion – guidance scale ω.

Environment	Discriminator	Reward (Ours, Ablation)	Reward (Ours)
Halfcheetah	$\omega = 0.5$	$\omega = 0.1$	$\omega = 0.2$
Hopper	$\omega = 0.4$	$\omega = 0.1$	$\omega = 0.4$
Walker2d	$\omega = 0.2$	$\omega = 0.3$	$\omega = 0.3$

this margin as in the trajectory generation procedure the goal position is not given as an exact position but by a circle with a diameter of 50% of a cell's side length.

We train reward functions for 8 distinct goal positions in each of the four maze configurations and for 5 seeds each. The confidence intervals are computed as $\hat{p} \pm 1.96 \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$, where $\hat{p}$ is the fraction of correct goal predictions, and n is the total number of predictions (e.g. for the total accuracy estimate, $n = 4$ mazes $\cdot$ 8 goals $\cdot$ 5 seeds $= 160$).

Running time. The Diffusers [24] take approximately 24 hours to train for 1 million steps on an NVIDIA P40 GPU. Reward functions took around 2 hours to train for 100000 steps for Maze environments, and around 4.5 hours for 50000 steps in Locomotion environments, also on an NVIDIA Tesla P40 GPU. For the Stable Diffusion experiment, it took around 50 minutes to run 6000 training steps on an NVIDIA Tesla M40 GPU. For Locomotion environment evaluations, we ran 512 independent rollouts in parallel for each steered model. Running all 512 of them took around 1 hour, also on an NVIDIA Tesla P40.

G. Ablations

G.0.1. ABLATION STUDY ON DATASET SIZE IN MAZE2D

The main experiments in Maze2D were conducted with datasets of 10 million transitions. To evaluate the sensitivity of our method to dataset size, we conducted a small ablation study of 24 configurations with datasets that contained only 10 thousand transitions, hence on the order of tens of trajectories. The accuracy in this ablation was at 75.0% (as compared to 78.12% in the main experiments). Examples of visual results for the learned reward functions are shown in Figur 6. This ablation is of high relevance, as it indicates that our method can achieve good performance also from little data.
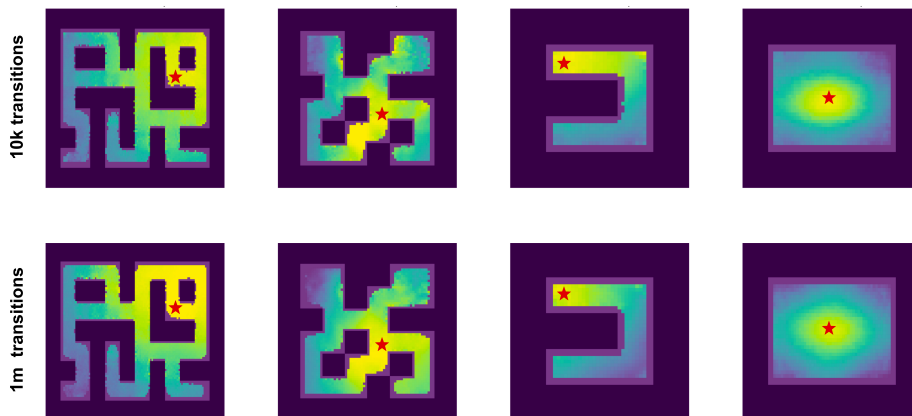

Figure 6: Heatmaps for rewards learned with 10 thousand and 1 million transition datasets. While there is slight performance degradation, the maxima of the reward still largely correspond to the ground-truth goals.

Table 5: Parameters for sampling (rollouts) in Locomotion – `t_stopgrad`.

Environment	Discriminator	Reward (Ours, Ablation)	Reward (Ours)
Halfcheetah	`t_stopgrad = 8`	`t_stopgrad = 0`	`t_stopgrad = 1`
Hopper	`t_stopgrad = 10`	`t_stopgrad = 2`	`t_stopgrad = 1`
Walker2d	`t_stopgrad = 1`	`t_stopgrad = 5`	`t_stopgrad = 2`

Table 6: Results for ablation on generalization in Locomotion.

Environment	Unsteered	Reward (Ours, Ablation)
Halfcheetah	31.74 ± 0.37	33.06 ± 0.31
Hopper	22.95 ± 0.81	25.03 ± 0.79
Walker2d	30.44 ± 1.01	42.40 ± 1.07
Mean	28.38	33.50

G.0.2. ABLATION STUDY ON GENERALIZATION IN LOCOMOTION

For the main steering experiments in Locomotion the reward functions are trained on the same base diffusion model that is then steered after. We conducted an ablation study to investigate whether the learned reward function generalizes to *other base* models, i.e. yields significant performance increases when used to steer a base model that was not part of the training process. We trained additional base models with new seeds and steered these base diffusion models with the previously learned reward function. We report results for this ablation in Table 6. We found that the relative increase in performance is similar to that reported in the main results in Table 1 and therefore conclude that our learned reward function generalizes to new base diffusion models.

H. Additional results for Maze2D