Extracting Reward Functions from Diffusion Models

Trinity Term, 2023

Third Year Project report

MCompSci Master of Computer Science

Felipe Pinto Coelho Nuti

Word Count: 4934

Abstract

Diffusion models are a class of generative models that have achieved remarkable results in image generation, protein modeling, 3D generation, and many other complex generation tasks. Recent work has established their applicability to sequential decision-making, exhibiting unusual properties. Most notably, it is possible to train a diffusion model on reward-agnostic demonstrations and produce plans optimizing a given reward function *without any need for retraining*, via *conditional sampling*. Meanwhile, the same result can be attained by directly training the diffusion model on already-optimal (expert) trajectories. The existence of these two ways of achieving the same goal motivate us to ask the question: can we go the other direction, and reconstruct the reward function from a reward-agnostic base model and an expert model? To answer this question, we first define the notion of *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 the relative reward function of two diffusion models by aligning the gradients of a neural network to the difference of their outputs. Our method empirically finds correct reward functions in different navigation environments. In addition, we verify that, in high-dimensional locomotion environments, our learned rewards can be plugged into lower-performance diffusion models to improve their performance through conditional sampling. Finally, we demonstrate that our approach generalizes beyond sequential decision-making by applying it to two large-scale image generation models of 859m parameters, and extracting a reward function that can distinguish harmful images from harmless ones.

1 Introduction

Recent work [9, 1] 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 (RL) as conditional sampling [16], Janner et al. [9] show that diffusion models can be used to parameterize a reward-agnostic prior distribution over trajectories, learned from offline demonstrations alone. Using classifier guidance [7], the diffusion model can then be steered with a neural network computing a (cumulative) reward function to generate near-optimal behaviors in various sequential decision-making tasks.

Diffusion models can hence achieve near-optimal behaviors in sequential decision-making tasks through either (a) training an *expert diffusion model* on a distribution of optimal trajectories, or (b) by training a *base diffusion model* on a distribution of 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 the base and expert diffusion models.

In this work, we introduce a method for extracting a reward function from pre-trained decision-making diffusion models. In contrast to prior work on reward learning [24, 18, 30, 4], 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 on their architecture.

We first derive a notion of a *relative reward function of two diffusion models*. We show that, under certain assumptions on the trajectory distribution and diffusion sampling process, our notion of reward exists and is unique, up to an additive constant. Further, the relative reward corresponds to the true reward under the probabilistic RL framework [16]. 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. Firstly, we show it is able to recover the goals of the expert model in a long-horizon navigation task. Secondly, we find that the learned rewards can improve the performance of low-reward models through conditional sampling. Finally, we apply our method to Stable Diffusion [22], a 859m-parameter image generation model, and show it is able to extract a reward function distinguishing harmful images from unharmful ones.

In light of the original theoretical and methodological contributions developed during this project, a version of this work has also been submitted to the 2023 Conference on Neural Information Processing Systems (NeurIPS 2023).

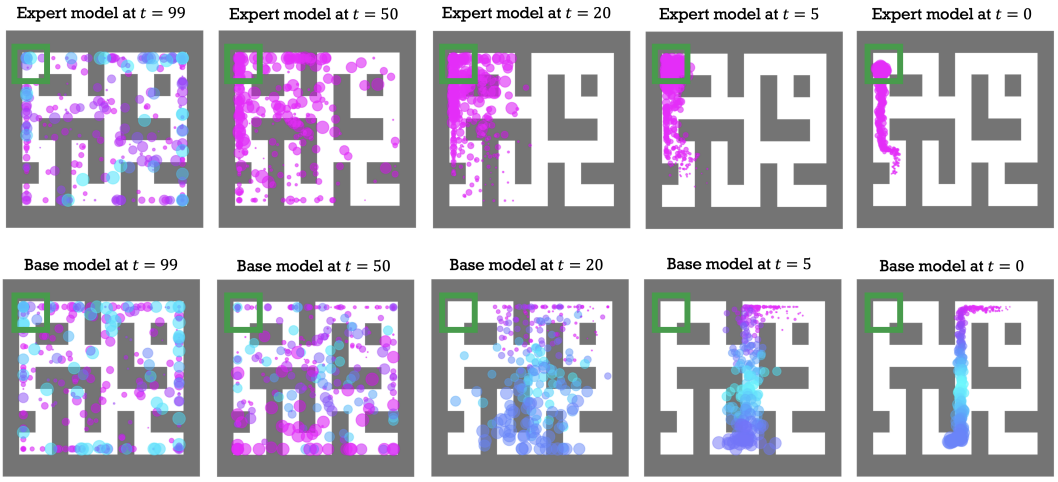


Figure 1: Our method learns to assign rewards to trajectories by comparing two diffusion models. In this example, blue indicates lower predicted reward, and pink indicates a higher predicted reward. We see that, at the start of generation, the assigned rewards are similar for the base and the expert model, whose goal is reaching the green square. As the trajectories become more and more denoised, however, our method assigns them very different rewards.

2 Background

Our method leverages mathematical properties of diffusion-based planners to extract reward functions from them. To put our contributions into context, we give a brief overview of the probabilistic formulation of Reinforcement Learning presented in [16], then of diffusion models, and finally of how they come together in decision-making diffusion models [9].

2.1 Reinforcement Learning as Probabilistic Inference

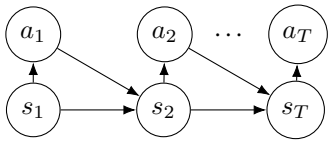
Levine [16] provides an in-depth exposition of existing methods for approaching sequential decision-making from a probabilistic and causal angle using Probabilistic Graphical Models (PGM) [10]. We review some essential notions to understand this perspective on RL.

Denote by $\Delta_{\mathcal{X}}$ the set of distributions over a set $\mathcal{X}$.

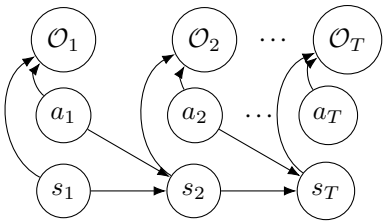
Markov Decision Process (MDP): a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, s_0, T \rangle$ consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathbb{R}_{\geq 0}}$, an initial state s_0, and an episode length T.

Evolution of an MDP: 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 $r_t \sim r(s_t, a_t)$. The episode ends at $t = T$. $\tau = ((s_t, a_t))_{t=0}^{T}$ is called a *trajectory*.

The framework in [16] recasts such an MDP as a PGM consisting 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.



(a) PGM for sampling from and MDP with a fixed policy π



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

Figure 2: Probabilistic Graphical Models for probabilistic Reinforcement Learning

Instead, it is encoded in the optimality variables via the relation: $\mathcal{O}_t \sim \text{Ber}(e^{-r(s_t,a_t)})$. See Figure 2b for an illustration. The problem of producing plans of high reward then reduces to *sampling from* $p(\tau|\mathcal{O}_{1:T})$.

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* $p(\mathcal{O}_{1:T}|\tau)$. Observe that, from the definition of $\mathcal{O}_{1:T}$ and the PGM factorization (Figure 2b), we have $p(\mathcal{O}_{1:T}|\tau) = e^{-\sum_t r(s_t,a_t)}$. Hence, $-\log p(\mathcal{O}_{1:T}|\tau)$ (a negative log-likelihood) corresponds to the cumulative reward of the trajectory τ:

$$-\log p(\mathcal{O}_{1:T}|\tau) = \sum_t r(s_t, a_t) \tag{2}$$

2.2 Overview of Diffusion Models

Janner et al. [9] leverage the factorization in (1) to sample from $p(\tau|\mathcal{O}_{1:T})$, using diffusion models [26] to parameterize the prior $p(\tau)$ over trajectories. We give a brief account of the continuous-time formulation of diffusion models, often used today in image generation [28, 27, 22, 25], and classifier guidance, a conditional sampling method.

2.3 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):

$$d\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}_t \tag{3}$$

where f is a function that is Lipschitz, $\mathbf{w}$ is a standard Brownian Motion [15] and g is a (continuous) noise schedule, which regulates the amount of noise added to the data during the forward process. Song et al. [28] then use a result of Anderson [2] to write the SDE satisfied by the *reverse process* of (3), denoted $\bar{\mathbf{x}}_t$, as:

$$d\bar{\mathbf{x}}_t = [\mathbf{f}(\bar{\mathbf{x}}_t, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\bar{\mathbf{x}}_t)]\, dt + g(t)\, d\bar{\mathbf{w}}_t \tag{4}$$

Here $p_t(\mathbf{x})$ denotes the marginal density function of the forward process $\mathbf{x}_t$ at time t, and $\bar{\mathbf{w}}$ is a reverse Brownian motion (see [2]). 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 (4) *backwards in time* to arrive at $\bar{\mathbf{x}}_0 \approx \mathbf{x}_0$, with the score term substituted for the neural network $\mathbf{s}_\Theta$:

$$d\bar{\mathbf{x}}_t = [\mathbf{f}(\bar{\mathbf{x}}_t, t) - g(t)^2 \mathbf{s}_\Theta(\bar{\mathbf{x}}_t, t)]\, dt + g(t)\, d\bar{\mathbf{w}}_t \tag{5}$$

This formulation is essential for deriving existence results for the relative reward function of two diffusion models in Section 3, 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:

$$d\bar{\mathbf{x}}_t = [\mathbf{f}(\bar{\mathbf{x}}_t, t) - g(t)^2 [\mathbf{s}_\Theta(\bar{\mathbf{x}}_t, t) + \omega \nabla_{\mathbf{x}} \rho(\bar{\mathbf{x}}_t, t)]]\, dt + g(t)\, d\bar{\mathbf{w}}_t \tag{6}$$

This method is called *classifier guidance*, and was introduced in [26]. Informally, it 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 $p(y|x)$.

2.4 Planning with Diffusion

The above shows how sequential decision-making can be framed as sampling from a posterior distribution $p(\tau|\mathcal{O}_{1:T})$ over trajectories. Section 2.3 shows how a diffusion model $p(\mathbf{x}_0)$ can be combined with a classifier to sample from a posterior $p(\mathbf{x}_0|y)$, without any re-training. These two observations point us to the approach in Diffuser, of Janner et al. [9]: 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 this diffusion model. Hence, steering decision-making diffusion models in this way allows for approximately sampling from $p(\tau|\mathcal{O}_{1:T})$, which produces (near-)optimal trajectories.

The policy produced by Diffuser denoises, at every timestep, fixed-length sequences of future states and actions using a diffusion model, taking only the first action of the sequence to act in the environment.

3 Methods

As we established in the previous section, in the probabilistic setting for RL, the (cumulative) return $\sum_t r(s_t, a_t)$ of a trajectory $\tau = ((s_t, a_t))_{t=1}^T$ corresponds directly with the output of the classifier $p(y|\mathbf{x})$ used for classifier guidance (Section 2.3). Hence, we consider the problem of finding $p(y|\mathbf{x})$ (i.e. $p(\mathcal{O}_{1:T}|\tau)$) in order to recover the cumulative reward for any given trajectory τ. Further, in Section 3.4, we show how the single-step reward can be computed by choosing a particular parametrization for $p(y|\mathbf{x})$.

3.1 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. 3.4 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 through Gradient Descent.

3.2 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 (4) from $t = T$ to $t = 0$, and $\mathbf{f}$ in the theorem corresponds to the drift term (i.e. coefficient of dt) of (4). For proof, see Appendix A.

Theorem 3.1 (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:

$$d\mathbf{x}_t^{(1)} = \mathbf{f}^{(1)}(\mathbf{x}_t^{(1)}, t)\, dt + g(t)\, d\mathbf{w}_t \tag{7}$$

$$d\mathbf{x}_t^{(2)} = \mathbf{f}^{(2)}(\mathbf{x}_t^{(2)}, t)\, dt + g(t)\, d\mathbf{w}_t \tag{8}$$

$$d\mathbf{x}_t \quad = [\mathbf{f}^{(1)}(\mathbf{x}_t, t) + \mathbf{h}(\mathbf{x}_t, t)]\, dt + g(t)\, d\mathbf{w}_t, \textit{where } \mathbf{h} \textit{ is Lipschitz} \tag{9}$$

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 (7), (8), and (9) 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

$$\mathbf{h}(\mathbf{x}, t) = \mathbf{f}^{(2)}(\mathbf{x}, t) - \mathbf{f}^{(1)}(\mathbf{x}, t). \tag{10}$$

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, and using a network with smooth activation functions like Mish [17] or GeLU [6].

Therefore, Theorem 3.1 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 3.2 (Conservative Vector Field, Definition 7.6 in [14])**.** 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 3.2, the diffusion models are conservative for each $t > 0$ (we exclude $t = 0$ to ensure p_t is supported in all of $\mathbb{R}^n$). 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 [14]), there exists a unique ρ satisfying $\nabla_{\mathbf{x}} \rho(\mathbf{x}, t) = \mathbf{s}^{(2)}_{\text{true}}(\mathbf{x}, t) - \mathbf{s}^{(1)}_{\text{true}}(\mathbf{x}, t)$, up to an additive constant, given by the line integral

$$\rho(\mathbf{x}, t) = \int_{\mathbf{x}_{\text{ref}}}^{\mathbf{x}} [\mathbf{s}^{(2)}_{\text{true}}(\mathbf{x}', t) - \mathbf{s}^{(1)}_{\text{true}}(\mathbf{x}', t)] \cdot d\mathbf{x}', \tag{11}$$

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.

3.3 Relative Reward Function of Two Diffusion Models

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. A.8), we obtain the following:

Proposition 3.3 (Optimal Relative Reward Gradient)**.** *Let $\mathbf{s}^{(2)}_\Theta$ and $\mathbf{s}^{(1)}_\phi$ be any two diffusion models and $t \in (0, T]$, with the assumption that $\mathbf{s}^{(2)}_\Theta(\cdot, t) - \mathbf{s}^{(1)}_\phi(\cdot, t)$ is square-integrable. Then there exists a unique vector field $\mathbf{h}_t$ given by*

$$\mathbf{h}_t = \underset{\mathbf{f} \in \overline{\text{Cons}(\mathbb{R}^n)}}{\text{argmin}} \int_{\mathbb{R}^n} ||\mathbf{f}(\mathbf{x}) - (\mathbf{s}^{(2)}_\Theta(\mathbf{x}, t) - \mathbf{s}^{(1)}_\phi(\mathbf{x}, t))||_2^2 \, d\mathbf{x} \tag{12}$$

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:

$$\int_{\mathbb{R}^n} ||\nabla_{\mathbf{x}} \Phi(\mathbf{x}) - \mathbf{h}_t(\mathbf{x})||_2^2 \, d\mathbf{x} < \varepsilon \tag{13}$$

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

For proof of Proposition 3.3 see Appendix A.3. It is important to note that for the projection to be well-defined, we required an assumption regarding the integrability of the difference of the diffusion models. Without this assumption, the integral would simply diverge.

The result in Proposition 3.3 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 3.4 (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 $\rho : \mathbb{R}^n \times [0, T] \to \mathbb{R}$ such that

$$\forall t \in (0, T] : \int_{\mathbb{R}^n} ||\nabla_{\mathbf{x}} \rho(\mathbf{x}, t) - \mathbf{h}_t(\mathbf{x})||_2^2 \, d\mathbf{x} < \varepsilon \tag{14}$$

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

3.4 Extracting Reward Functions

We now set out to actually approximate the relative reward function ρ. Definition 3.4 naturally translates into an L^2 training objective for learning ρ:

$$L_{\text{RRF}}(\theta) = \mathbb{E}_{t \sim \mathcal{U}[0,T], \mathbf{x}_t \sim p_t} \left[||\nabla_{\mathbf{x}} \rho_\theta(\mathbf{x}_t, t) - (\mathbf{s}_\Theta^{(2)}(\mathbf{x}_t, t) - \mathbf{s}_\phi^{(1)}(\mathbf{x}_t, t))||_2^2 \right] \tag{15}$$

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

We parameterize ρ using a neural network (more specifically, a UNet architecture [23]) and optimize this objective via Empirical Risk Minimization and Adam [12]. See Algorithm 1 for a version assuming access to the diffusion models and their training datasets, and Algorithm 2 in Appendix B 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 [20].

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 i^{th} state and action in τ_t. Then, $g_\theta(s, a, 0)$ predicts a reward for a state-action pair (s, a).

4 Experiments

In this section, we conduct empirical investigations to analyze the properties of relative reward functions in practice. Our experiments focus on three main properties we would expect from relative reward functions. Firstly, the learned rewards should be compatible with the goals of the expert agent. Secondly, it should be possible to steer the base model using the learned reward and make

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: $\mathbf{X}_\mathbf{t} \leftarrow [\mathbf{x}_{t_1}^{(1)}, ..., \mathbf{x}_{t_N}^{(N)}]$
  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

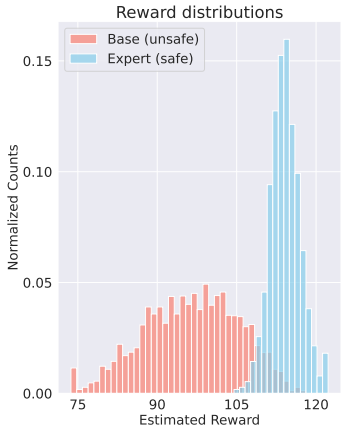


Figure 3: Learned rewards for base and expert diffusion models from Stable Diffusion (Sec. 4).

it achieve higher ground-truth rewards. Finally, it should be possible to learn relative rewards in domains beyond decision-making, as they are defined for any pair of diffusion models.

4.1 Learning Correct Reward Functions from Long-Horizon Plans

Maze2D [3] features various environments which involve controlling the acceleration of a ball to navigate it towards various goal positions in 2D mazes.

Implementation. To conduct our experiments, we generate multiple datasets of trajectories in each of the 2D environments, following the data generation procedure from D4RL [3], except sampling start and goal positions uniformly at random (as opposed to only at integer coordinates, as is done in the original library).

For four maze environments with different wall configurations (depicted in Figure 4), we first train a base diffusion model on a dataset of uniformly sampled start and goal positions, representing reward-agnostic 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 resulting 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. We find that the distributions are clearly separated.

To evaluate this quantitatively, we train a logistic regression classifier per expert model on a balanced dataset to label trajectories (as base or expert) using the predicted reward. We repeat this process

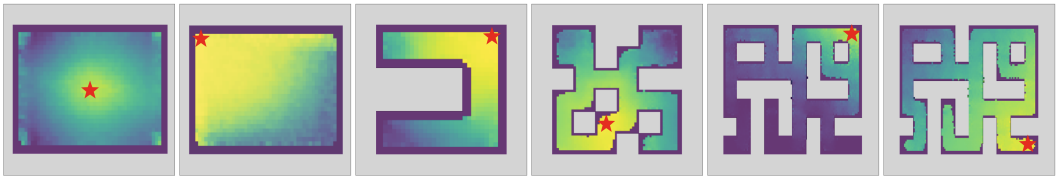


Figure 4: Maze2D learned reward heatmaps. ★ denotes the ground-truth goal position of the expert diffusion model. For more examples, see Appendix C.

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. 4 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% $\pm$ 8.96% of the cases for simpler mazes (`maze2d-open-v0` and `maze2d-umaze-v0`), and 77.50% $\pm$ 9.15% for more advanced mazes (`maze2d-medium-v1` and `maze2d-large-v1`). Overall, the network achieves an average success rate of 78.12% $\pm$ 6.40%. The aforementioned margins are 95% confidence intervals, computed across 5 random seeds 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.

4.2 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 [3]. These tasks involve controlling a multi-degree-of-freedom 2D robot to move forward at the highest possible 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 provides 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 base model of below-average performance towards producing

Table 1: Diffuser performance with different steering methods, on three locomotion 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**

Table 2: 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

better-performing trajectories. 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 use classifier guidance (Section 2.3) to steer each base model using the corresponding learned reward function.

Results. We sample 512 independent trajectories of the base diffusion model steered by the learned reward, using various guidance scales ω (Eq. 6). 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.

4.3 Learning a reward-like function for Stable Diffusion

The notion of a reward function is native to sequential decision-making problems. Still, our analysis in Section 3 shows a way in which it may be extended to more general domains through the concept of relative reward functions. We now set out to evaluate this empirically in one of the domains where diffusion models have displayed the most outstanding performance: image generation. We look at two 859m-parameter diffusion models widely used in practice: Stable Diffusion [22], a very popular general-purpose image generation model; and Safe Stable Diffusion [25], a modified version of Stable Diffusion tailored to preventing shocking images from being generated.

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 [8] during sampling and can be steered using natural language prompts, using CLIP embeddings [21] in the latent space. Specifically, Safe Stable Diffusion [25] introduces modifications to the *sampling loop* of the open-source model proposed by Rombach et al. [22], 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 [25, 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. [25]. 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 containing an equal number of base and expert samples is set aside for model 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 to do classification. The reward histogram is visualized in Figure 3.

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 5.

4.4 Ablations

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). This ablation indicates that our method can perform well even when given little data.

Generalization in Locomotion. 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 2, and find that relative improvements are comparable to

Figure 5: The 3 images with the highest learned reward in their batch (“safe”), and 3 with the lowest reward (“unsafe”, blurred), respectively.

those in Table 1 and therefore conclude that our learned reward function generalizes to new base diffusion models.

5 Limitations and Further Work

The main limitation of this work is the extent of the experiments. Further work could hence evaluate our method’s performance in a wider set of tasks and domains. In addition, we believe some of our theoretical results could be strengthened; Proposition 3.3 in particular. Finally, a deeper study of the implications of our work in AI Safety is left to further work.

6 Conclusion

To the best of our knowledge, our work introduces the first method for extracting relative reward functions from 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.

7 Acknowledgements

We use the following technologies and repositories as components in our code: PyTorch [20], NumPy [5], Diffuser [9], D4RL [3], HuggingFace Diffusers [29] and LucidRains’s Diffusion Models in Pytorch repository. In particular, we use the code from [9] as a starting point for our codebase.

References

[1] 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* .

[2] 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*

[3] 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* .

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

[5] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. [2020], 'Array programming with NumPy', *Nature* **585**(7825), 357–362.
URL: *https://doi.org/10.1038/s41586-020-2649-2*

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

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

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

[9] 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*

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

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

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

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

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

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

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

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

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

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

[20] 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*

[21] 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.

[22] 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.

[23] Ronneberger, O., Fischer, P. and Brox, T. [2015], 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', Springer, pp. 234–241.

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

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

[26] 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.

[27] 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*

[28] 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'.

[29] von Platen, P., Patil, S., Lozhkov, A., Cuenca, P., Lambert, N., Rasul, K., Davaadorj, M. and Wolf, T. [2022], 'Diffusers: State-of-the-art diffusion models', `https://github.com/huggingface/diffusers`.

[30] 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.

Appendix A Details on Main Derivation

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

Theorem A.1 (Existence and uniqueness theorem for SDEs, c.f. pp. 66 of Øksendal and Øksendal [19])**.** *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*

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

Then the stochastic differential equation

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

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*

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

Remark A.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 equivalent (Theorem 2.4-5 in Kreyszig [13]), the stated growth conditions do not depend on the particular norm chosen for $\mathbb{R}^n$.

Corollary A.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* $\mathcal{F}\infty$ *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 A.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 that is adapted to* $\mathcal{F}$*, where* $(\mathcal{F}_t^{\mathbf{z}})_{t\geq 0}$ *is defined as in A.1.*

Proof. Firstly, note that, as $\mathbf{f}$ and g are continuous, they are Lebesgue-measurable. It remains to check the growth conditions in A.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 $t, s \in [0,T]$:

$$|\mathbf{f}(\mathbf{x},t) - \mathbf{f}(\mathbf{y},s)| \leq C_1(|\mathbf{x}-\mathbf{y}|+|t-s|) \tag{16}$$

$$\leq C_1(|\mathbf{x}-\mathbf{y}|+T) \tag{17}$$

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:

$$|g(t) - g(s)| \leq 2C_2 \tag{18}$$

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

$$|\mathbf{f}(\mathbf{x}, t)| + |g(t)| \leq ((T + M)C_1 + C_2)(1 + |\mathbf{x}|) \qquad |\mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\mathbf{y}, t)| \leq C_1|\mathbf{x} - \mathbf{y}| \tag{19}$$

which yields the growth conditions required in A.1, and the conclusion follows. □

A.2 Proof of 3.1

Proof of Theorem 3.1. We reference the fundamental result on the existence and uniqueness of strong solutions to SDEs, presented in Section 5.2.1 of [19] and reproduced in Appendix A. In particular, Corollary A.3 shows that the Existence and Uniqueness Theorem A.1 applies for SDEs 7 and 8, 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 9. This establishes the existence of a.s. unique (adapted) a.s. continuous sample paths $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$ and $\mathbf{x}$.

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 9 have the same drift and noise coefficients as SDE 8, and so, by the uniqueness of solutions we established, it follows that $\mathbf{x}_t^{(2)} = \mathbf{x}_t$ for all t, a.s.

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 $x^{(2)}$. (i.e. equal 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 $t \in [0, T]$:

$$\begin{aligned}
0 &= \tilde{\mathbf{x}}_t - \mathbf{x}_t^{(2)} && (20)\\
&= \left(\int_0^t \mathbf{f}^{(1)}(\tilde{\mathbf{x}}_s, s) + \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_s, s) ds + \int_0^t g(s)\, d\mathbf{w}_s \right) - \left(\int_0^t \mathbf{f}^{(2)}(\mathbf{x}_s^{(2)}, s) ds + \int_0^t g(s)\, d\mathbf{w}_s \right) && (21)\\
&= \int_0^t \mathbf{f}^{(1)}(\tilde{\mathbf{x}}_s, s) + \tilde{\mathbf{h}}(\tilde{\mathbf{x}}_s, s) ds - \int_0^t \mathbf{f}^{(2)}(\mathbf{x}_s^{(2)}, s) ds && (22)\\
&= \int_0^t \mathbf{f}^{(1)}(\mathbf{x}_s^{(2)}, s) + \tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s) ds - \int_0^t \mathbf{f}^{(2)}(\mathbf{x}_s^{(2)}, s) ds && (23)\\
&= \int_0^t \tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s) - (\mathbf{f}^{(2)}(\mathbf{x}_s^{(2)}, s) - \mathbf{f}^{(1)}(\mathbf{x}_s^{(2)}, s)) ds && (24)\\
&= \int_0^t \tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s) - \mathbf{h}_1(\mathbf{x}_s^{(2)}, s) ds && (25)
\end{aligned}$$

where in 21 we substitute the integral forms of the SDEs satisfied by $\tilde{\mathbf{x}}$ and $\mathbf{x}^{(2)}$, and in 23 we use that the processes are indistinguishable and 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 25 to conclude that a.s. $\tilde{\mathbf{h}}(\mathbf{x}_s^{(2)}, s) = \mathbf{h}_1(\mathbf{x}_s^{(2)}, s)$ for all s.

Finally, 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

$$
\begin{aligned}
||\Delta(\mathbf{x}', t)|| &\leq C||\mathbf{x}' - \mathbf{y}_\varepsilon|| && (26)\\
&\leq C(||\mathbf{x}' - \mathbf{x}|| + ||\mathbf{x} - \mathbf{y}_\varepsilon||) && (27)\\
&\leq C(\frac{\varepsilon}{2C} + \frac{\varepsilon}{2C}) && (28)\\
&= \varepsilon && (29)
\end{aligned}
$$

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

$$\tilde{\mathbf{h}}(\mathbf{x}, 0) = \lim_{s \downarrow 0} \tilde{\mathbf{h}}(\mathbf{x}, s) = \lim_{s \downarrow 0} \mathbf{h}(\mathbf{x}, s) = \mathbf{h}_1(\mathbf{x}, 0) \tag{30}$$

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

A.3 Existence and Uniqueness of Minimizer of (12)

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

$$L^2(\mathbb{R}^n, \mathbb{R}^n) = \left\{ \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n : \int_{\mathbb{R}^n} ||\mathbf{f}(\mathbf{x})||_2^2 d\mathbf{x} < \infty \right\} \tag{31}$$

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

$$||\mathbf{f}||_2^2 = \int_{\mathbb{R}^n} ||\mathbf{f}(\mathbf{x})||_2^2 \, d\mathbf{x} \tag{32}$$

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 A.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.

$$\text{Cons}(\mathbb{R}^n) = \left\{ \nabla f : f \text{ is smooth, } \int_{\mathbb{R}^n} |f(\mathbf{x})|^2 d\mathbf{x} < \infty \text{ and } \int_{\mathbb{R}^n} ||\nabla f(\mathbf{x})||_2^2 d\mathbf{x} < \infty \right\} \tag{33}$$

Remark A.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 [11], Chapter 9).

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

$$\overline{\text{Cons}(\mathbb{R}^n)} = \{\mathbf{f} : \text{there exists } (\nabla f_k)_{k\geq 0} \subseteq \text{Cons}(\mathbb{R}^n) \text{ such that } ||\nabla f_k - \mathbf{f}||_2^2 \to 0 \text{ as } k \to \infty\} \tag{34}$$

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 A.8 (Complementation in Hilbert Spaces, c.f. Kreyszig [13], 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 A.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 3.3. 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 $\overline{\text{Cons}(\mathbb{R}^n)}$ is a closed subspace of $L^2(\mathbb{R}^n, \mathbb{R}^n)$, Corollary A.9 applies, and we have that there is a unique minimizer $\mathbf{h}_t \in \overline{\text{Cons}(\mathbb{R}^n)}$ in Equation 12.

As $\overline{\text{Cons}(\mathbb{R}^n)}$ consists of $L^2(\mathbb{R}^n, \mathbb{R}^n)$ limits of sequences in $\text{Cons}(\mathbb{R}^n)$, 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 that

$$\int_{\mathbb{R}^n} ||\nabla_{\mathbf{x}}\Phi_k(\mathbf{x}) - \mathbf{h}_t(\mathbf{x})||_2^2 \, d\mathbf{x} < \varepsilon \tag{35}$$

which completes the proof. □

Appendix B Relative Reward Learning Algorithms

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
  ```
  $\hat{L}_{\text{RRF}}(\theta) \leftarrow ||\nabla_{\mathbf{x}} \rho_\theta(\mathbf{x}_{t+1}, t+1) - (\mathbf{x}_t^{(2)} - \mathbf{x}_t^{(1)})||_2^2$
  Take an Adam [12] optimization step on θ according to $\hat{L}_{\text{IRL}}(\theta)$
end

Appendix C Further Maze2D Reward Heatmaps