Adversarial Diffusion for Robust Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Robustness to modeling errors and uncertainties remains a central challenge in reinforcement learning (RL). In this work, we address this challenge by leveraging diffusion models to train robust RL policies. Diffusion models have recently gained popularity in model-based RL due to their ability to generate full trajectories "all at once", mitigating the compounding errors typical of step-by-step transition models. Moreover, they can be conditioned to sample from specific distributions, making them highly flexible. We leverage conditional sampling to learn policies that are robust to uncertainty in environment dynamics. Building on the established connection between Conditional Value at Risk (CVaR) optimization and robust RL, we introduce Adversarial Diffusion for Robust Reinforcement Learning (AD-RRL). AD-RRL guides the diffusion process to generate worst-case trajectories during training, effectively optimizing the CVaR of the cumulative return. Empirical results across standard benchmarks show that AD-RRL achieves superior robustness and performance compared to existing robust RL methods.

1 Introduction

Reinforcement Learning (RL) has produced agents that surpass human-level performance in various domains [29, 55, 46, 45, 30]. However, the same policies are notoriously sensitive to small dynamics changes, sensor noise, or hardware mismatch, all of which can cause dramatic performance collapse. In safety–critical domains such as robotics, finance, or healthcare—where collecting new data is expensive, risky, or legally restricted—robustness to modeling errors is at least as important as maximizing nominal reward.

Model-based RL improves sample efficiency by learning a world-model and planning within it, but faces two key robustness obstacles: (i) compounding errors, which accumulate over long horizons [56, 8]; and (ii) the Sim2Real gap, where controllers that succeed in simulation fail after minor real-world deviations [41, 6]. Compounding error occurs in autoregressive models, where the model predicts one step ahead and then is fed its own prediction back: the state predicted at time t is used to predict the state at $t+1$. Small one-step errors accumulate, the trajectory moves away from reality, and performance degrades. The Sim2Real gap arises because, even with a highly accurate simulator that minimizes unrealistic artifacts, simulated physics can never perfectly replicate reality. This discrepancy leads to reduced real-world performance due to inherent modeling inaccuracies. To overcome these challenges, RL algorithms should be made more robust by optimizing not only the expected return but also the performance under adverse or uncertain dynamics.

Recently, diffusion models have been proposed to mitigate compounding errors by generating entire trajectories rather than predicting one step at a time [38, 17]. While this reduces error accumulation, diffusion models remain imperfect: the trajectories they generate may deviate from real-world dynamics. As a result, transferring policies learned in simulation to the real world remains challenging.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Despite recent progress, diffusion-based RL methods often struggle to maintain robustness when deployed in environments with unseen or perturbed dynamics.

Adversarial and risk-sensitive approaches have been explored to enhance robustness against model errors. These methods introduce worst-case perturbations during planning [35, 37], or optimize Conditional Value at Risk (CVaR) objectives, which have been shown to improve resilience to both reward variability and model inaccuracies [34, 5]. In this work, we show that diffusion models and CVaR-based approaches can be seamlessly integrated to complement each other. We propose Adversarial Diffusion for Robust Reinforcement Learning (AD-RRL), a novel algorithm that combines the strengths of diffusion models and CVaR-based robustness. By leveraging trajectory-level generation to mitigate compounding errors and incorporating risk-aware objectives, AD-RRL enhances the adaptability and robustness of RL agents to modeling mismatches and environmental uncertainty. More precisely, we make the following contributions.

(a) We present Adversarial Diffusion (AD), a guided diffusion model that for a given policy, generates trajectories that are challenging for the agent and result in relatively low rewards. These trajectories are either rare in the current environment or originate from unexplored regions of the domain. We show that by learning from such adversarial scenarios, the agent can improve its robustness to modeling errors. To generate these trajectories, we leverage the CVaR framework, applied to trajectory rewards, and demonstrate how guided diffusion can be used to efficiently implement this objective. This mechanism forms the foundation of AD.

(b) Building on this, we introduce AD-RRL, our RL algorithm that integrates AD within the Advantage Actor-Critic (A2C) framework. AD-RRL significantly enhances the agent's adaptability and robustness. We empirically evaluate AD-RRL across multiple environments from the Gym/MuJoCo suite, showing that it achieves superior robustness to modeling errors. In transfer scenarios involving variations in physics parameters, AD-RRL consistently outperforms state-of-the-art baselines.

2 Related Work

Model-Based Reinforcement Learning. In Model-Based Reinforcement Learning, the agent uses a model to generate new data, through which it is possible to plan further without interacting with the environment. This is essential for settings where collecting new data is impossible, illegal or dangerous. The parametric approach has received a lot of attention thanks to the constant improvements of function approximators, such as Deep Neural Networks [32, 7, 18, 16, 53]. Recently, Variational Auto Encoders [19] and Transformers [54] have seen many successful applications as powerful models for environment dynamics [28, 39, 42, 12, 11], leading to state-of-the-art methods in terms of sample efficiency and performance [13]. These methods rely on bootstrapping to generate trajectory samples. The state prediction generated by the model is fed again as input to the model to predict the next state. As a result, these methods introduce two sources of error: one coming from an imperfect model and one from the input of the model always being wrong, except for the first timestep. The sum of these errors is commonly known as the Compounding Error problem of Model-Based methods. Multi-step prediction solutions have been proven effective even before the introduction of Diffusion models, for example by learning H models to look H steps into the future [3]. It goes without saying that this approach results in a much higher learning complexity with respect to single-model approaches. Only recently, with Diffusion models becoming more popular, we have seen the rise of more efficient multi-step Model-Based RL methods [38, 17].

Diffusion Models in Reinforcement Learning. Diffusion models are inspired by non-equilibrium thermodynamics [47, 15], defining data generation as an iterative denoising process. Beyond being powerful function approximators, they also offer a natural way to condition the data generation process on labels [10]. Recently, diffusion models have gained significant attention in the RL community. They have been used to model system dynamics, generating trajectory segments by predicting either states [1, 58], actions [4, 22], or both [17, 24]. Guidance techniques can further refine trajectory generation by conditioning the process on value estimates, promoting high-expected-reward sequences. Additionally, diffusion models have been employed for policy modeling [56, 14] and value function approximation [26]. Most research on diffusion models in RL has focused on the offline setting. In this paper, we shift the focus to the online case, building on PolyGRAD—an online, Dyna-style Model-Based RL method that uses diffusion for modeling dynamics [38]. While prior work primarily uses conditioning to generate high-reward trajectories, we take the opposite approach.

Our goal is to generate challenging trajectories—those that are underexplored or unlikely—so the agent can learn a more robust policy, better suited to handling changes in dynamics and modeling errors.

Robust Reinforcement Learning. Classical RL can struggle to generalize when test environments deviate from training due to model errors or shifts. Robust RL addresses this by accounting for uncertainty in actions, states, and dynamics. One line of work regularizes transition probabilities within a defined uncertainty set [9, 20, 25]. This kind of methods, despite being theoretically sound and robust, do not scale well to more complex environment.

A well-known approach to tackling complex robust RL problems while maintaining theoretical guarantees is to frame the optimization problem as a two-player game [31]. In this framework, two players are trained iteratively to solve a maximin optimization problem: the primary agent aims to maximize the expected cumulative reward, while an adversarial agent attempts to minimize it by introducing disturbances. For instance, in Robust Adversarial Reinforcement Learning (RARL) [35], the adversary applies external forces to disturb the environment's dynamics. Max-min TD3 (M2TD3) [50] follows a similar strategy, solving a maximin problem to maximize the expected reward under worst-case scenarios within an uncertainty set. In Noisy Action Robust MDPs [51], the adversary perturbs the agent's actions, while in State Adversarial MDPs [48], the adversary introduces perturbations to the state, resulting in a Partially Observable MDP formulation.

Several Robust RL algorithms use CVaR to constrain their optimization problems [57]. For instance, CVaR-PPO is an extension of PPO [44] solving a risk-sensitive constrained optimization problem that constrains the CVaR to a given threshold.

Finally, we have algorithms using Domain Randomization (DR) [52], where the agent maximizes the expected return on average, over a predefined uncertainty set for some given environment parameters. These classes of methods have been proven very effective in domains such as robotics [23]. However, they do not aim to be robust to the worst-case scenarios, and might fall short when tested on environments outside of their training distribution.

3 Background and Problem Statement

3.1 Markov Decision Processes and Reinforcement Learning

Consider a Markov Decision Process (MDP) $M = \langle S, A, P, r, \gamma, \rho \rangle$, where S and A are the state and action spaces, respectively, $P(\cdot|s, a)$ is the transition probability function, $r(\cdot|s, a)$ is the reward function, γ is the discount factor and ρ is the initial state distribution. By interacting with the MDP, a Reinforcement Learning agent is able to collect sequences of states, actions and rewards, forming trajectories $\boldsymbol{\tau} = (s_0, a_0, r_0, \ldots, s_H, a_H, r_H)$. The objective of the Reinforcement Learning agent is to learn an optimal policy $\pi^\star$ maximizing the policy value $v_\pi(s) = \mathbb{E}_\pi[\sum_{i=0}^{\infty} \gamma^i r_{t+i+1} | s_t = s]$. In this paper, we consider a model-based RL setting, where we use a diffusion model to approximate the distribution of trajectories under a given policy. Specifically, if p^π denotes the true distribution of the trajectories $\boldsymbol{\tau}$ under policy π, the diffusion model samples trajectories with distribution $p_{\boldsymbol{\theta}}$ close to p^π. We adopt a Dyna-style approach [49], where the diffusion model and the policy are iteratively updated: the policy is improved using data collected from the model, while the model is improved using samples gathered from the target environment using the learned policy.

3.2 Robust RL through the Conditional Value at Risk.

We now discuss Conditional Value at Risk (CVaR) and its connection to Robust RL.

Conditional Value at Risk. When learning policies robust to modeling errors, a framework commonly used is the one of Conditional Value at Risk. We define the return of a trajectory $\boldsymbol{\tau}$ by $Z(\boldsymbol{\tau}) = \sum_{t=0}^{H} \gamma^t r_t$, where r_t is the reward obtained at time t in this trajectory[1]. Under a policy π, $Z(\boldsymbol{\tau})$ is a random variable with cdf F. The Value-at-Risk (VaR) of Z at confidence level $\alpha \in (0, 1)$ corresponds to its α quantile:

$$\mathrm{VaR}_\alpha^\pi(Z) = \max\{z | F(z) \leq \alpha\}. \quad (1)$$

[1]To avoid cluttering, we write Z instead of $Z(\boldsymbol{\tau})$ unless it is required to avoid misunderstandings.

The Conditional Value-at-Risk (CVaR) of Z is then defined as the expected value of Z on the lower α-portion of its distribution

$$\text{CVaR}_\alpha^\pi(Z) = \mathbb{E}_\pi[Z | Z \leq \text{VaR}_\alpha^\pi(Z)]. \tag{2}$$

CVaR dual formulation and its connection to robustness to modeling errors. An alternative way of defining CVaR stems from its dual formulation [2, 40]:

$$\text{CVaR}_\alpha^\pi(Z) = \min_{\xi \in \mathcal{U}_{CVaR}^{\alpha,\pi}} \mathbb{E}_{\boldsymbol{\tau} \sim p^\pi}[\xi(\boldsymbol{\tau}) Z(\boldsymbol{\tau})], \tag{3}$$

where ξ acts as a perturbation of the return Z. This perturbation belongs to the set $\mathcal{U}_{\text{CVaR}}^{\alpha,\pi}$, called the *risk envelope* and defined as

$$\mathcal{U}_{\text{CVaR}}^{\alpha,\pi} := \left\{ \xi : \forall \boldsymbol{\tau}, \xi(\boldsymbol{\tau}) \in \left[0, \frac{1}{\alpha}\right], \mathbb{E}_{\boldsymbol{\tau} \sim p^\pi}[\xi(\boldsymbol{\tau})] = 1 \right\}. \tag{4}$$

(3) states that the CVaR of Z can be defined as its expected value under a worst-case perturbed distribution.

In RL, optimizing a CVaR objective introduces robustness to model misspecification. This is exactly because of the dual form of CVaR, where the trajectory distribution is distorted by an adversarial density $\xi(\boldsymbol{\tau})$. CVaR optimization in this case equals maximizing the worst-case discounted reward when adversarial perturbations are budgeted over the whole trajectory rather than at each time step [5]. The connection between CVaR and robustness to modeling errors is well established in the RL field [34, 37, 35], and the dual formulation is at the core of our method, as explained in Section 4 and Appendix B.

3.3 Diffusion Models

In this work, we adopt a model-based approach to learn robust policies. To achieve this, we harness the efficiency of diffusion processes to learn a parameterized model $p_{\boldsymbol{\theta}}$ of the trajectory distribution. This model allows us to sample trajectories $\boldsymbol{\tau}$ as if they were generated by the true MDP, enabling policy training on these synthetic trajectories.

Diffusion models generate data by progressively refining noisy inputs through an iterative denoising process, $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i)$. This process reverses the forward diffusion, $q(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i-1})$, which gradually corrupts real data by adding random noise. Each step of the denoising process is typically parameterized as a Gaussian distribution

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i), \boldsymbol{\Sigma}_i), \tag{5}$$

with learned mean and fixed covariance matrices, both depending on the diffusion step i.

The denoising process is formulated as

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{0:N}) = p(\boldsymbol{\tau}_N) \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i), \tag{6}$$

where $p(\boldsymbol{\tau}_N) \approx \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\boldsymbol{\tau}_0$ is the real (i.e., noiseless) trajectory. The parameters $\boldsymbol{\theta}$ are learned by optimizing the variational lower bound on the negative log likelihood:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\tau}_0}[-\log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0)], \tag{7}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) = \int p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{0:N}) \mathrm{d}\boldsymbol{\tau}_{1:N}$.

Guided diffusion. A classifier $p(y|\boldsymbol{\tau}_0)$ adding information about the sample to be reconstructed (e.g., the optimality of the trajectory) can enhance the generative performance of the diffusion model [10]

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|y) \propto p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) p(y|\boldsymbol{\tau}_0). \tag{8}$$

By leveraging the classifier's gradient, we can guide the denoising process toward generating samples that align more closely with the classifier's predictions. This method, called Classifier-Guided Diffusion, generates trajectory samples according to

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i, y) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) + \boldsymbol{\Sigma}_i \boldsymbol{g}_i, \boldsymbol{\Sigma}_i) \tag{9}$$

where $\boldsymbol{g}_i = \nabla_{\boldsymbol{\tau}} \log p(y|\boldsymbol{\tau})|_{\boldsymbol{\tau} = \mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)}$.

3.4 Problem statement

We consider a model-based Reinforcement Learning setting. For a given policy π, we learn a model $p_{\boldsymbol{\theta}}$ of the distribution of the corresponding trajectories. Our goal is to use $p_{\boldsymbol{\theta}}$ to improve the policy and its robustness to modeling errors.

We formulate the problem of learning a robust policy using the following optimization problem:

$$\pi_\alpha^\star = \arg\max_\pi \mathrm{CVaR}_\alpha^\pi(Z) \tag{10}$$

$$= \arg\max_\pi \min_{\xi \in \mathcal{U}_{C\mathrm{VaR}}^{\alpha,\pi}} \mathbb{E}_{\boldsymbol{\tau}\sim p^\pi}[\xi(\boldsymbol{\tau})Z(\boldsymbol{\tau})]. \tag{11}$$

(10) describes the objective to obtain the policy that maximizes the return on the worst α-percentile of the trajectories, in terms of cumulative return. However, directly sampling trajectories from this worst α-percentile is challenging. In our approach, we leverage the dual definition of CVaR, solving instead the double optimization problem described in (11). The problem can be seen as a game where an adversarial agent ξ is perturbing the trajectories distribution under a given policy π. We model this distribution via a diffusion model $p_{\boldsymbol{\theta}}$, which allows us to leverage guiding techniques. We introduce *adversarial guiding*, a method that steers the diffusion process toward sampling trajectories that minimize the expected return for the agent. Because the adversarial guide actively seeks to reduce return, the generated trajectories naturally fall within the worst α-percentile. We formally demonstrate that the resulting adversarially guided diffusion models can be adapted to actually sample from the worst α-percentile. We also empirically validate our approach.

4 Adversarially Guided Diffusion Models

In this section, we consider a fixed policy π, and for notational convenience, we drop the corresponding superscripts. We explain below how to efficiently generate adversarial trajectories, sampled from the set of trajectories $C_\alpha := \{\boldsymbol{\tau} : Z(\boldsymbol{\tau}) \leq \mathrm{VaR}_\alpha(Z)\}$.

Sampling suboptimal trajectories. To steer the diffusion process towards the set C_α we need to define the proper guidance classifier, as in (8). We start from the definition of $\mathrm{CVaR}_{\alpha,p_{\boldsymbol{\theta}}}(Z)$ given in (2). The index $p_{\boldsymbol{\theta}}$ indicates that the trajectory $\boldsymbol{\tau}$ from which the return is computed is generated using the diffusion model $p_{\boldsymbol{\theta}}$. We have:

$$\mathrm{CVaR}_{\alpha,p_{\boldsymbol{\theta}}}(Z) = \mathbb{E}_{\tau_0\sim p_{\boldsymbol{\theta}}}[Z(\boldsymbol{\tau}_0)|\boldsymbol{\tau}_0 \in C_\alpha], \tag{12}$$

$$= \int Z(\boldsymbol{\tau}_0) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_0 \in C_\alpha) d\boldsymbol{\tau}_0, \tag{13}$$

$$= \min_{\xi \in \mathcal{U}_{C\mathrm{VaR}}^{\alpha,\pi}} \int Z(\boldsymbol{\tau}_0)\xi(\boldsymbol{\tau}_0) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) d\boldsymbol{\tau}_0, \tag{14}$$

where the last equality follows from the dual definition of CVaR presented in (3). To steer the generating process towards trajectories from C_α, we can use the classifier $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0 \in C_\alpha|\boldsymbol{\tau}_0)$, since $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_0 \in C_\alpha) \propto p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0 \in C_\alpha|\boldsymbol{\tau}_0)$.

Notice also that if we define $\xi^\star(\boldsymbol{\tau}_0)$ as the solution to the minimization problem in (14), i.e., the bounded trajectory perturbation minimizing the cumulative reward under the dynamics $p_{\boldsymbol{\theta}}$, we have

$$\xi^\star(\boldsymbol{\tau}_0) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) = p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_0 \in C_\alpha) \propto p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0 \in C_\alpha|\boldsymbol{\tau}_0).$$

In other words, weighting the distribution with the classifier $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0 \in C_\alpha \mid \boldsymbol{\tau}_0)$ is equivalent, up to a proportionality constant, to weighting $p_\theta(\tau_0)$ according to an adversarial perturbation $\xi^\star$. We can hence think of applying a guided diffusion to implement this perturbation. However, the set C_α is not known. To address this limitation, and following the approach of [21], we introduce a smooth approximation of $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0 \in C_\alpha \mid \boldsymbol{\tau}_0)$, namely $\exp(-c_0 \sum_{t=1}^H \gamma^t r_t)$ for some constant $c_0 > 0$. This approximation is intuitively reasonable, as it biases the generation process toward trajectories with lower cumulative rewards.

In the following two subsections, we describe how this guided diffusion can be implemented and how it influences the diffusion process. We also discuss how to tune the guided diffusion to ensure that the resulting adversarial perturbation ξ remains within the risk envelope $\mathcal{U}_{C\mathrm{VaR}}^{\alpha,\pi}$ defined in (4).

4.1 Perturbed diffusion model

We use the classifier $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i \in C_\alpha|\boldsymbol{\tau}_i)$ so that the trajectories $\boldsymbol{\tau}_i$ generated at every step i of the diffusion process belong to the set C_α. We assume that $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i \in C_\alpha|\boldsymbol{\tau}_i) \approx \exp\left(-c_i \sum_{t=1}^{H} \gamma^t r_t^{(i)}\right)$ for some value $c_i > 0$ as we did for τ_0. In the last approximation, the reward $r_t^{(i)}$ represents the reward collected at time t in trajectory $\boldsymbol{\tau}_i$ for the i-th step of the diffusion process.

As a slight extension of the guided diffusion principle presented in Section 3.3, we establish the following result (essentially obtained by applying (9) with $y = \{\boldsymbol{\tau}_i \in C_\alpha\}$).

Lemma 4.1. *Assume that the denoising process is Gaussian, that is (5) holds. Assume that for all $i \in [N]$, the approximation $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i \in C_\alpha|\boldsymbol{\tau}_i) = \exp\left(-c_i \sum_{t=1}^{H} \gamma^t r_t^{(i)}\right)$ holds. Then, we can sample trajectories from $p_\theta(\boldsymbol{\tau}_0|\boldsymbol{\tau}_0 \in C_\alpha)$ using diffusion steps of the form:*

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1} \in C_\alpha) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i, \boldsymbol{\Sigma}_i), \tag{15}$$

where $\boldsymbol{g}_i = \nabla_{\boldsymbol{\tau}} Z(\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))$ for $i \in [N]$.

The lemma is proved in Appendix A for completeness. The conditional sampling procedure induces the following perturbed model:

$$\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) = p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_0 \in C_\alpha) \propto \int \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1} \in C_\alpha) p(\boldsymbol{\tau}_N) \mathrm{d}\boldsymbol{\tau}_{1:N}. \tag{16}$$

We refer to this sampling procedure as an *Adversarially Guided Diffusion Model*.

4.2 Selecting $c_1, \ldots, c_N$

Note that the Adversarially Guided Diffusion Model depends on the constants $c_1, \ldots, c_N$, and recall that the resulting perturbation must lie within the risk envelope defined in (4). In the following, we establish conditions on these constants to ensure this requirement is satisfied. To that end, we first show in Appendix B that our model $\bar{p}_{\boldsymbol{\theta}}$ admits a product-form representation:

Lemma 4.2. *The Adversarially Guided Diffusion Model can be expressed as $\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) = \xi(\boldsymbol{\tau}_0) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0)$, where $\xi(\boldsymbol{\tau}_0) = \frac{\int \xi(\boldsymbol{\tau}_{0:N}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{0:N}) d\boldsymbol{\tau}_{1:N}}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0)}$ and where $\xi(\boldsymbol{\tau}_{0:N}) \coloneqq \prod_{i=1}^{N} \xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1})$ with*

$$\xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1}) \coloneqq \exp\left(-\frac{1}{2}(2c_i \boldsymbol{D}_i^T \boldsymbol{g}_i + c_i^2 \boldsymbol{g}_i^T \boldsymbol{\Sigma} \boldsymbol{g}_i)\right), \tag{17}$$

and $\boldsymbol{D}_i \coloneqq (\boldsymbol{\tau}_{i-1} - \mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))$.

Note that by definition (since $\bar{p}_{\boldsymbol{\theta}}$ is a distribution), we have that $\mathbb{E}_{\boldsymbol{\tau} \sim p_{\boldsymbol{\theta}}}[\xi(\boldsymbol{\tau}_0)] = 1$. Hence, we just need to verify that $\xi(\boldsymbol{\tau}_0) \leq 1/\alpha$ for all $\boldsymbol{\tau}_0$ to ensure that ξ belongs to the risk envelope. We define R such that the trajectories $\boldsymbol{\tau}_i$ lie in a bounded space $C = \{\boldsymbol{\tau}_i : ||\boldsymbol{\tau}_i||_\infty \leq R\}$ and such that $||\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty < R$. In the following proposition, proved in Appendix C, we provide conditions on $c_1, \ldots, c_N$ so that this holds.

Proposition 4.3. *For all $i \in [N]$, let $\eta_i(\alpha, N) \geq 0$ such that $\prod_{i=1}^{N} \eta_i(\alpha, N) = \frac{1}{\alpha}$.*
(a) When for all $i \in [N]$,

$$c_i \leq \min\left(\sqrt{\frac{2\log \eta_i(\alpha, N)}{\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i}}, \frac{R - ||\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty}{||\boldsymbol{\Sigma}_i \boldsymbol{g}_i||_\infty}\right), \tag{18}$$

then we have: for all $\boldsymbol{\tau}_0$, $\xi(\boldsymbol{\tau}_0) \leq 1/\alpha$.
(b) Let $i \in [N]$. Assume that $\boldsymbol{\Sigma}_i$ is a diagonal matrix with $(\boldsymbol{\Sigma}_i)_{jj} \in [0, 1)$. Assume $\eta_i(\alpha, N) = \left(\frac{1}{\alpha}\right)^{\frac{1}{N}}$. Then, for N large enough, (18) holds as soon as $c_i \leq \sqrt{\frac{2\log \eta_i(\alpha, N)}{\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i}}$.

Since our diffusion model uses a cosine noise schedule as in [33], we have that for all $i \in [N]$ $\boldsymbol{\Sigma}_i = \beta_i \boldsymbol{I}$ with $\beta_i \in [0, 1)$, so we can set $c_i = \sqrt{\frac{2\log \eta_i(\alpha, N)}{\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i}}$ to ensure that ξ belongs to the risk envelope.

Remark 4.4. Note that in our analysis, we have assumed for simplicity that the states and the actions were a one-dimensional vector, so that trajectories become Gaussian vectors. We can extend the analysis to the case where states and actions are multidimensional at the expense of considering trajectories as Gaussian matrices. Refer to Appendix D for details.

5 Algorithms

We now introduce Adversarial Diffusion for Robust Reinforcement Learning (AD-RRL), which alternates between model improvement and policy improvement steps (see Algorithm 1). AD-RRL leverages the adversarial conditional sampling discussed in the previous section to sample trajectories in the worst α-percentile in terms of return. Our approach is primarily inspired by PolyGRAD [38] and Diffuser [17].

We adopt the common assumption that the policy follows a Gaussian distribution over the action space, parameterized by $\mu_\omega(s)$ and $\sigma_\omega(s)$. The policy is deployed in the real environment to collect new data, which is then used to train both the dynamics model $\bar{p}_{\boldsymbol{\theta}}$ and the cumulative reward function $Z_{\boldsymbol{\phi}}$. Following the standard approach in Dyna-like algorithms [49], we generate synthetic trajectories using our learned models (Algorithm 2). These trajectories are then used to train the policy via an on-policy Reinforcement Learning algorithm.

Algorithm 1 Adversarial Diffusion for Robust Reinforcement Learning (AD-RRL)

```
Input: environment, E;
Initialize: policy, π_ω; adversarial denoising model, p̄_θ; cumulative reward function Z_φ;
    data buffer, D; training iterations M

for m = 1, ..., M do
    Sample τ ~ E using π_ω, add τ to D
    Improve p̄_θ, Z_φ on D            ▷ Algorithm 3
    Sample {τ̂} ~ p̄_θ                  ▷ Algorithm 2
    Improve π_ω on {τ̂} using RL
end for
```

Algorithm 2 Adversarial Diffusion Trajectory Sampling

```
Input: adversarial denoising model p̄_θ;
reward model Z_φ; buffer D; level α

τ̂_N ~ N(0, I)
s_0 ~ D
for i = N, ..., 1 do
    set ŝ_0 ← s_0 in τ̂_i
    c_i = sqrt(2 log η_i(α, N) / g_i^T Σ_i g_i)
    τ̂_{i-1} ~ N(μ_θ(τ̂_i, i) − c_i Σ_i g_i, Σ_i)
end for
return τ̂_0
```

Algorithm 3 illustrates the training procedure for both our diffusion models and is presented in Appendix E, alongside additional implementation details. The pseudocode provided is simplified. In reality, the diffusion model consists of a noise prediction function $\epsilon_{\boldsymbol{\theta}}(\hat{\boldsymbol{\tau}}_i, i)$ from which the mean is computed in closed form [15]. This model is trained using the following objective function (derived from (7))

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{i,\epsilon,\boldsymbol{\tau}_0}[||\epsilon - \epsilon_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||^2],$$

where $i \sim \mathcal{U}(\{1, \ldots, N\})$ is the diffusion process step, $\epsilon \sim \mathcal{N}(0, 1)$ is the target noise and $\boldsymbol{\tau}_i$ is the trajectory $\boldsymbol{\tau}_0 \sim \mathcal{D}$ after i steps of the *forward* diffusion process adding noise ϵ. We update $\boldsymbol{\theta}$ K times, each time by randomly sampling the step i. The model $Z_{\boldsymbol{\phi}}$ is trained to predict the cumulative reward of the trajectory samples $\boldsymbol{\tau}_i$.

Both the adversarial diffusion model $\bar{p}_{\boldsymbol{\theta}}$ and the cumulative reward function $Z_{\boldsymbol{\phi}}$ are used to sample adversarially generated trajectories in the worst $\alpha-$percentile. At every step of the diffusion process, we perform inpainting by substituting a real starting state $\boldsymbol{s}_0$ into the generated noisy trajectory $\hat{\boldsymbol{\tau}}_i$. We then proceed to compute c_i according to (18) and the gradient $\boldsymbol{g}_i = \nabla_{\boldsymbol{s}} Z_\phi$. Notice that the gradient is taken with respect to $\boldsymbol{s}$, so we only adversarially corrupt the states of the trajectory. To ensure that the generated actions are consistent with the generated states, we use the PolyGRAD diffusion guidance method [38], which generates a sequence of actions guided by the gradient of the policy π_ω.

6 Experiments

In this section, we empirically evaluate how robust our method is. During training, the agent interacts with a fixed instance of the environment. At test time, we alter key physics-related parameters and assess the agent's performance against both robust and non-robust baselines. Our experiments are conducted on several optimal control tasks from the MuJoCo suite: InvertedPendulum, Reacher, Hopper, HalfCheetah, and Walker. All agents are trained in the default MuJoCo/OpenAI Gym environment (fixed physics), for 1.5M steps. Additional results are provided in Appendix F.

Baseline methods. We evaluate AD-RRL against several state-of-the-art baselines for robust reinforcement learning:

(a) Domain Randomization (DR) [52], widely used in robotics [23, 27], improves policy generalization by maximizing expected return over a distribution of dynamics. However, it does not explicitly account for worst-case or lower-percentile outcomes. We implement DR using PPO and refer to the resulting method as DR-PPO.

(b) Max-Min TD3 (M2TD3) [50] frames robustness as a minimax optimization problem, training an actor-critic model to maximize performance under the worst-case dynamics sampled from a predefined uncertainty set.

(c) CVaR-PPO (CPPO) [57] augments Proximal Policy Optimization with a CVaR constraint, leading to a policy-gradient algorithm that explicitly controls the policy's risk.

Additionally, we compare AD-RRL to other baselines in RL.

(d) PolyGRAD [38], a diffusion-based model that our work builds upon, generates synthetic trajectories via policy-guided diffusion and trains policies in an online model-based setting. It improves sample efficiency but lacks explicit robustness to adverse dynamics.

(e) TRPO [43] and PPO [44], two strong model-free baselines, are also included for comparison. TRPO constrains policy updates using a KL-divergence trust region, while PPO employs a clipped surrogate objective for improved computational efficiency.

Robustness under varying physical parameters. To verify the robustness of AD-RRL, we vary several physical parameters of the environment at test time. For Hopper and Cheetah, we vary body mass, ground friction and environment gravity. For Walker, we modify friction and mass. For Reacher, we vary all the actuators' gears (i.e., the torque produced by the actions). For InvertedPendulum, we change the cart mass, the pole mass and environment gravity.

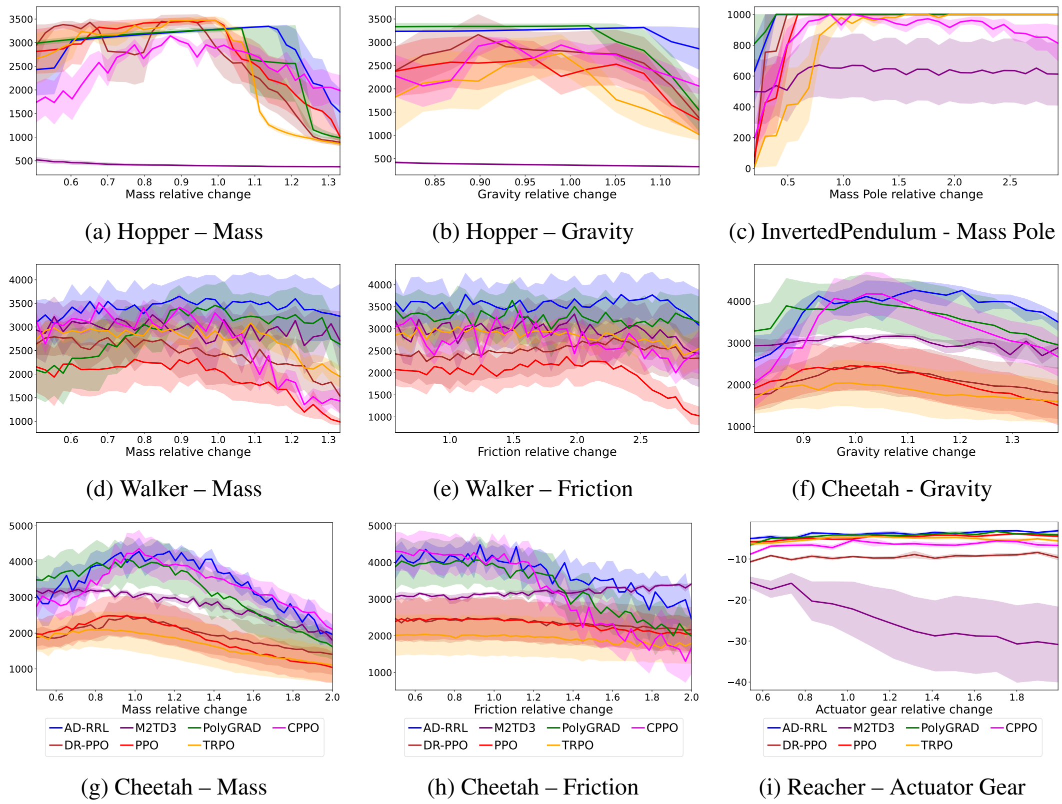
In Figure 1, we plot the return under the different algorithms and for selected environments and varying parameters. Additional plots are provided in Appendix F.2. In most environments, AD-RRL consistently outperforms both robust and non-robust baselines. PPO and TRPO appear surprisingly stable, which is likely a consequence of the well-tuned Stable-Baselines3 implementations—but are still matched or surpassed by AD-RRL. At the same time, AD-RRL consistently outperforms both DR-PPO and M2TD3, demonstrating greater stability and achieving higher cumulative rewards across all environments.

Furthermore, a direct comparison with PolyGRAD (the foundation of our algorithm) highlights that our modifications significantly improve performance under diverse test-time conditions, enhancing robustness to large parameter shifts and model misspecifications. This can be clearly seen in Figure 1d or Figure 1g. In the Reacher environment (Figure 1i) the difference in performance is less evident, but our model still performs consistently better or on par with the baselines. It is also clear from Figures 1a and 1b that while PolyGRAD achieves slightly better performance on the nominal environment (as observed in Table 1), it sacrifices robustness under perturbed conditions.

For some environments—see for example Figure 1a and Figure 3b (presented in Appendix F), AD-RRL performance degrades for extreme changes in the modified parameter (but it remains better than other algorithms). We hypothesize that this is because our model is generating challenging trajectories which are nonetheless plausible under the agent policy and environment dynamics. Extreme changes in the environment physics do not reflect these constraints, and relevant trajectories might not be generated often.

Performance on the nominal (training) environments. Table 1 reports the final episode returns (mean $\pm$ one standard error) for five MuJoCo continuous-control tasks. Best results are highlighted in bold. The results are obtained on the training environment, with the nominal physics parameters. AD-RRL attains the best mean return on four of the five domains, substantially outperforming the other baselines, showing that our risk-sensitive training does not trade nominal optimality for robustness. Only on Hopper, PolyGRAD performs better than AD-RRL, but the margin falls within overlapping confidence intervals. On the easy Inverted Pendulum task, multiple methods (AD-RRL, DR-PPO, PPO) reach the maximum score of 1000, as expected.

Sample efficiency. The learning curves in Appendix F.1 (see Appendix F) show that AD-RRL reaches higher or matching final performance with the same number of samples as the baselines,



(a) Hopper – Mass (b) Hopper – Gravity (c) InvertedPendulum - Mass Pole

(d) Walker – Mass (e) Walker – Friction (f) Cheetah - Gravity

(g) Cheetah – Mass (h) Cheetah – Friction (i) Reacher – Actuator Gear

Figure 1: Average return across variations in selected physics parameters. Shaded regions indicate ± one standard error.

	Hopper	Cheetah	Walker	Reacher	InvertedPendulum
AD–RRL	3280.23 ± 13.83	$\mathbf{4126.11 \pm 246.96}$	$\mathbf{4357.80 \pm 187.37}$	$\mathbf{-3.97 \pm 0.13}$	$\mathbf{1000.00 \pm 0.00}$
PolyGRAD	$\mathbf{3346.99 \pm 52.39}$	3879.16 ± 626.40	3489.48 ± 456.70	-4.48 ± 0.13	$\mathbf{1000.00 \pm 0.00}$
M2TD3	361.73 ± 13.71	3117.16 ± 55.34	2948.03 ± 598.77	-21.28 ± 5.75	634.76 ± 192.46
CPPO	2595.64 ± 298.35	2173.30 ± 422.97	2164.30 ± 510.60	-6.06 ± 0.32	979.85 ± 20.15
DR–PPO	2315.90 ± 482.17	2429.46 ± 558.93	2385.19 ± 589.49	-15.80 ± 1.44	$\mathbf{1000.00 \pm 0.00}$
PPO	2998.90 ± 432.28	2408.20 ± 546.33	1894.03 ± 349.06	-5.17 ± 0.57	$\mathbf{1000.00 \pm 0.00}$
TRPO	3270.27 ± 273.04	2014.91 ± 539.64	3090.80 ± 267.79	-6.22 ± 0.85	960.60 ± 39.40

Table 1: Return on the training environment (nominal physics parameters) for MuJoCo continuous-control tasks.

and in several cases converges faster. Hence, our adversarially guided diffusion not only preserves (or improves) performance on the training environment (with nominal physics parameters), but also matches the sample efficiency of state-of-the-art model-based and model-free alternatives.

7 Conclusion and Future Work

In this work we introduced AD-RRL, a novel approach to robust RL. AD-RRL is based on Adversarial Diffusion (AD), a diffusion model that can sample adversarial trajectories by leveraging the Conditional Value at Risk (CVaR) framework. AD enables agents to learn from adversarial scenarios that are either rare or unexplored in the environment. We demonstrated that AD-RRL, based on this diffusion model, significantly enhances the robustness of RL agents in the presence of modeling errors. Through empirical evaluation on multiple Gym/MuJoCo environments, we showed that AD-RRL outperforms current state-of-the-art robust RL methods.

AD relies on a specific strategy for guiding the diffusion process, and exploring alternative guidance methods presents a promising avenue for future work. Potential directions include (i) modifying the overall diffusion objective beyond the current CVaR framework, and (ii) enhancing the diffusion model architecture or algorithms to reduce computational overhead.

References

[1] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[2] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

[3] K. Asadi, D. Misra, S. Kim, and M. L. Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

[4] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[5] Y. Chow, A. Tamar, S. Mannor, and M. Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. *Advances in neural information processing systems*, 28, 2015.

[6] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

[7] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[8] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629. PMLR, 2018.

[9] E. Derman, M. Geist, and S. Mannor. Twice regularized mdps and the equivalence between robustness and regularization. *Advances in Neural Information Processing Systems*, 34:22274–22287, 2021.

[10] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[11] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[12] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering Atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[13] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

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

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

[16] T. Jafferjee, E. Imani, E. Talvitie, M. White, and M. Bowling. Hallucinating value: A pitfall of dyna-style planning with imperfect environment models. *arXiv preprint arXiv:2006.04363*, 2020.

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

[18] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al. Model-based reinforcement learning for Atari. *arXiv preprint arXiv:1903.00374*, 2019.

[19] D. P. Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] N. Kumar, K. Levy, K. Wang, and S. Mannor. Efficient policy iteration for robust markov decision processes via regularization. *arXiv preprint arXiv:2205.14327*, 2022.

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

[22] X. Li, V. Belagali, J. Shang, and M. S. Ryoo. Crossway diffusion: Improving diffusion-based visuomotor policy via self-supervised learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16841–16849. IEEE, 2024.

[23] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *The International Journal of Robotics Research*, page 02783649241285161, 2024.

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

[25] Z. Liu, Q. Bai, J. Blanchet, P. Dong, W. Xu, Z. Zhou, and Z. Zhou. Distributionally robust q-learning. In *International Conference on Machine Learning*, pages 13623–13643. PMLR, 2022.

[26] B. Mazoure, W. Talbott, M. A. Bautista, D. Hjelm, A. Toshev, and J. Susskind. Value function estimation using conditional diffusion models for control. *arXiv preprint arXiv:2306.07290*, 2023.

[27] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[28] V. Micheli, E. Alonso, and F. Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[29] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[31] J. Morimoto and K. Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

[32] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[33] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[34] T. Osogami. Robustness and risk-sensitivity in markov decision processes. *Advances in neural information processing systems*, 25, 2012.

[35] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning*, pages 2817–2826. PMLR, 2017.

[36] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[37] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[38] M. Rigter, J. Yamada, and I. Posner. World models via policy-guided trajectory diffusion. *arXiv preprint arXiv:2312.08533*, 2023.

[39] J. Robine, M. Höftmann, T. Uelwer, and S. Harmeling. Transformer-based world models are happy with 100k interactions. *arXiv preprint arXiv:2303.07109*, 2023.

[40] R. T. Rockafellar. Coherent approaches to risk in optimization under uncertainty. In *OR Tools and Applications: Glimpses of Future Technologies*, pages 38–61. Informs, 2007.

[41] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on robot learning*, pages 262–270. PMLR, 2017.

[42] I. Schubert, J. Zhang, J. Bruce, S. Bechtle, E. Parisotto, M. Riedmiller, J. T. Springenberg, A. Byravan, L. Hasenclever, and N. Heess. A generalist dynamics model for control. *arXiv preprint arXiv:2305.10912*, 2023.

[43] J. Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

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

[45] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[46] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[47] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[48] S. Stanton, R. Fakoor, J. Mueller, A. G. Wilson, and A. Smola. Robust reinforcement learning for shifting dynamics during deployment. 2021.

[49] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[50] T. Tanabe, R. Sato, K. Fukuchi, J. Sakuma, and Y. Akimoto. Max-min off-policy actor-critic method focusing on worst-case robustness to model misspecification. *Advances in Neural Information Processing Systems*, 35:6967–6981, 2022.

[51] C. Tessler, Y. Efroni, and S. Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

[52] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[53] E. van der Pol, T. Kipf, F. A. Oliehoek, and M. Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. *arXiv preprint arXiv:2002.11963*, 2020.

[54] A. Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[55] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

[56] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[57] C. Ying, X. Zhou, H. Su, D. Yan, N. Chen, and J. Zhu. Towards safe reinforcement learning via constraining conditional value-at-risk. *arXiv preprint arXiv:2206.04436*, 2022.

[58] Z. Zhu, M. Liu, L. Mao, B. Kang, M. Xu, Y. Yu, S. Ermon, and W. Zhang. Madiff: Offline multi-agent learning with diffusion models. *arXiv preprint arXiv:2305.17330*, 2023.

Contents

A Adversarially Guided Diffusion Models: Proofs of results from Section 4.1

Proof of Lemma 4.1. For a sufficiently smooth function r, the conditional distribution $\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1})$ can be approximated using a Gaussian. Following [47, Appendix C] (see also [10]) we know that

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1} \in C_\alpha) \approx \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) + \boldsymbol{\Sigma}_i \boldsymbol{h}_i, \boldsymbol{\Sigma}_i)$$

where $\boldsymbol{h}_i = \nabla_{\boldsymbol{\tau}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau} \in C_\alpha|\boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i,i)}$.

Since we assume that the approximation $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i \in C_\alpha|\boldsymbol{\tau}_i) = \exp\left(-c_i \sum_{t=1}^{H} \gamma^t r_t^{(i)}\right)$ holds, we get that

$$\begin{aligned}
\boldsymbol{h}_i &= \nabla_{\boldsymbol{\tau}} \log p(\boldsymbol{\tau} \in C_\alpha|\boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i,ti)} \\
&= -c_i \sum_{t=1}^{T} \nabla_{\boldsymbol{s_t},\boldsymbol{a_t}} r(s_t, a_t)|_{(s_t,a_t)=\mu^t_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i,i)} \\
&= -c_i \nabla_{\boldsymbol{\tau}} Z(\boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i,i)},
\end{aligned}$$

where $\mu^t_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)$ is the t-th state-action pair of $\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)$. Substituting, we get

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1} \in C_\alpha) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i, \boldsymbol{\Sigma}_i), \tag{19}$$

where $\boldsymbol{g}_i = \nabla_{\boldsymbol{\tau}} Z(\boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i,i)}$. $\square$

Proof of Equation (16). As mentioned earlier, to sample from $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_0 \in C_\alpha)$, we multiply each intermediate distribution in the diffusion process by $r_i(\boldsymbol{\tau}_i)$, with $r_i(\boldsymbol{\tau}_i) = \exp(-c_i \sum_{t=1}^{H} \gamma^t r_t^{(i)})$, where the notation $r_t^{(i)}$ refers to the t-th reward in $\boldsymbol{\tau}_i$ for the i-th diffusion step. This means that the corresponding modified distribution $\bar{p}_{\boldsymbol{\theta}}$ satisfies in the intermediate diffusion step i:

$$\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i) = \frac{1}{\bar{Z}_i} r_i(\boldsymbol{\tau}_i) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i), \tag{20}$$

where $\tilde{Z}_i$ is the normalizing constant. Next, we use the same strategy as that used in [47] to determine the diffusion process $\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1})$. Note first that:

$$\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i) = \int \bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1}) \bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i+1}) \mathrm{d}\boldsymbol{\tau}_{i+1}.$$

Plugging (20), the previous condition can be rewritten as

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i) = \int \bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1}) \frac{\bar{Z}_i}{\bar{Z}_{i+1}} \frac{r_{i+1}(\boldsymbol{\tau}_{i+1})}{r_i(\boldsymbol{\tau}_i)} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i+1}) \mathrm{d}\boldsymbol{\tau}_{i+1}. \tag{21}$$

However, we know that $p_{\boldsymbol{\theta}}$ also satisfies:

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i) = \int p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i+1}) \mathrm{d}\boldsymbol{\tau}_{i+1}.$$

This implies that (21) holds if:

$$\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1}) = p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1}) \frac{\bar{Z}_{i+1} r_i(\boldsymbol{\tau}_i)}{\bar{Z}_i r_{i+1}(\boldsymbol{\tau}_{i+1})}.$$

Now defining the normalization constant $\bar{Z}_i(\boldsymbol{\tau}_{i+1}) = \frac{\bar{Z}_{i+1}}{\bar{Z}_i r_{i+1}(\boldsymbol{\tau}_{i+1})}$, we get

$$\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1}) = \frac{1}{\bar{Z}_i(\boldsymbol{\tau}_{i+1})} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1}) r_i(\boldsymbol{\tau}_i).$$

We conclude that $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1}, \boldsymbol{\tau}_i \in C_\alpha) \propto \tilde{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i|\boldsymbol{\tau}_{i+1})$. Therefore, we have shown that:

$$\begin{aligned}
p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_0 \in C_\alpha) &= \bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0), \\
&= \int \bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_1) \bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_1) \mathrm{d}\boldsymbol{\tau}_1, \\
&= \int \bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_1) \cdots \bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{N-1}|\boldsymbol{\tau}_N) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_N) \mathrm{d}\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_N, \\
&\propto \int p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_1, \boldsymbol{\tau}_0 \in C_\alpha) \cdots p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{N-1}|\boldsymbol{\tau}_N, \boldsymbol{\tau}_{N-1} \in C_\alpha) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_N) \mathrm{d}\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_N.
\end{aligned}$$

$\square$

B Adversarial guide as a multiplicative perturbation: Proof of Lemma 4.2

Proof of Lemma 4.2. Let's define a denoising diffusion model $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i)$, and a perturbed denoising step of the form $p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1} \in C_\alpha) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i, \boldsymbol{\Sigma}_i)$. Since the two distributions are Gaussians with known mean and covariance matrices, we have

$$\begin{aligned}
p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1} \in C_\alpha) &= \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i, \boldsymbol{\Sigma}_i) \\
&= K \exp\left(-\frac{1}{2}(\boldsymbol{D}_i + c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i)^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{D}_i + c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i)\right) \\
&= K \exp\left(-\frac{1}{2}(\boldsymbol{D}_i^T \boldsymbol{\Sigma}_i \boldsymbol{D}_i + 2c_i \boldsymbol{D}_i^T \boldsymbol{g}_i + c_i^2 \boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i)\right) \\
&= K \exp\left(-\frac{1}{2}\boldsymbol{D}_i^T \boldsymbol{\Sigma}_i \boldsymbol{D}_i\right) \exp\left(-\frac{1}{2}(2c_i \boldsymbol{D}_i^T \boldsymbol{g}_i + c_i^2 \boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i)\right) \\
&= \mathcal{N}(\boldsymbol{\tau}_{i-1}|\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i), \boldsymbol{\Sigma}_i) \exp\left(-\frac{1}{2}(2c_i \boldsymbol{D}_i^T \boldsymbol{g}_i + c_i^2 \boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i)\right) \\
&= \xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i)
\end{aligned}$$

where $K = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_i|^{1/2}}$, $\boldsymbol{D}_i = (\boldsymbol{\tau}_{i-1} - \mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))$, and $\xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1}) = \exp\left(-\frac{1}{2}(2c_i \boldsymbol{D}_i^T \boldsymbol{g}_i + c_i^2 \boldsymbol{g}_i^T \boldsymbol{\Sigma} \boldsymbol{g}_i)\right)$. $\mathcal{N}(\boldsymbol{\tau}_{i-1}|\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i), \boldsymbol{\Sigma}_i)$ is the density of the Gaussian distribution of $\boldsymbol{\tau}_{i-1}$, with mean $\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)$ and covariance matrix $\boldsymbol{\Sigma}_i$.

If we define $\xi(\boldsymbol{\tau}_{0:N}) = \prod_{i=1}^{N} \xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1})$, we have

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{0:N}|\boldsymbol{\tau}_0 \in C_\alpha) = \xi(\boldsymbol{\tau}_{0:N}) p(\boldsymbol{\tau}_{0:N})$$

So we can define

$$\begin{aligned}
\bar{p}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0) = p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_0|\boldsymbol{\tau}_0 \in C_\alpha) &= \int p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{0:N}|\boldsymbol{\tau}_0 \in C_\alpha) = 1) \mathrm{d}\boldsymbol{\tau}_{1:N} \\
&= \int \xi(\boldsymbol{\tau}_{0:N}) p(\boldsymbol{\tau}_{0:N}) \mathrm{d}\boldsymbol{\tau}_{1:N} \\
&= \frac{\int \xi(\boldsymbol{\tau}_{0:N}) p(\boldsymbol{\tau}_{0:N}) \mathrm{d}\boldsymbol{\tau}_{1:N}}{P(\boldsymbol{\tau}_0)} P(\boldsymbol{\tau}_0) \\
&= \xi(\boldsymbol{\tau}_0) P(\boldsymbol{\tau}_0)
\end{aligned}$$

with $\xi(\boldsymbol{\tau}_0) = \frac{\int \xi(\boldsymbol{\tau}_{0:N}) p(\boldsymbol{\tau}_{0:N}) \mathrm{d}\boldsymbol{\tau}_{1:N}}{P(\boldsymbol{\tau}_0)}$. □

C Attaining the duality constraint on $\xi(\boldsymbol{\tau}_0)$: Proof of Proposition 4.3

Proof of Proposition 4.3.

Proof of (a): From Lemma 4.2 we know that $\xi(\boldsymbol{\tau}_0) = \frac{\int \xi(\boldsymbol{\tau}_{0:N}) p(\boldsymbol{\tau}_{0:N}) \mathrm{d}\boldsymbol{\tau}_{1:N}}{P(\boldsymbol{\tau}_0)}$. We want to have $\xi(\boldsymbol{\tau}_0) \leq \frac{1}{\alpha}$, this is equivalent to

$$\begin{aligned}
\frac{\int \xi(\boldsymbol{\tau}_{0:N}) p(\boldsymbol{\tau}_{0:N}) \mathrm{d}\boldsymbol{\tau}_{1:N}}{P(\boldsymbol{\tau}_0)} &\leq \frac{1}{\alpha}, \\
\int \xi(\boldsymbol{\tau}_{0:N}) p(\boldsymbol{\tau}_{0:N}) \mathrm{d}\boldsymbol{\tau}_{1:N} &\leq \frac{1}{\alpha} \int p(\boldsymbol{\tau}_{0:N}) \mathrm{d}\boldsymbol{\tau}_{1:N}.
\end{aligned}$$

One way to achieve this is to impose $\xi(\boldsymbol{\tau}_{0:N}) = \prod_{i=1}^{N} \xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1}) \leq \frac{1}{\alpha}$. This is satisfied also by constraining the single terms of the product using $\eta_i(\alpha, N)$ such that $\xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1}) \leq \eta_i(\alpha, N)$ and $\prod_{i=1}^{N} \eta_i(\alpha, N) = \frac{1}{\alpha}$.

However, $\boldsymbol{\tau}_{i-1}$ is a random quantity to which we do not have access at step i of the diffusion process. Therefore, to satisfy the constraints on the single terms we impose

$$\max_{\boldsymbol{\tau}_{i-1}} \xi(\boldsymbol{\tau}_{i-1}, \boldsymbol{\tau}_i) \leq \eta_i(\alpha, N).$$

$\xi(\boldsymbol{\tau}_{i-1}, \boldsymbol{\tau}_i)$ is maximized for $(\boldsymbol{\tau}_{i-1} - \mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))^T \boldsymbol{g} < 0$, so we want $(\boldsymbol{\tau}_{i-1} - \mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))^T$ to be a vector opposite to $\boldsymbol{g}$. We can take $\boldsymbol{\tau}_{i-1} = \mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i$.

We assume that the trajectories $\boldsymbol{\tau}_i$ lie in a bounded space $C = \{\boldsymbol{\tau}_i : ||\boldsymbol{\tau}_i||_\infty \leq R\}$, where $||\boldsymbol{\tau}_i||_\infty = \max_{\boldsymbol{s} \in \boldsymbol{\tau}_i} ||\boldsymbol{s}||_\infty$. From this assumption it follows that

$$\begin{aligned} ||\boldsymbol{\tau}_{i-1}||_\infty &\leq R \\ ||\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i||_\infty &\leq R \\ ||\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty + c_i ||\boldsymbol{\Sigma}_i \boldsymbol{g}_i||_\infty &\leq R \\ c_i &\leq \frac{(R - ||\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty)}{||\boldsymbol{\Sigma}_i \boldsymbol{g}_i||_\infty}. \end{aligned}$$

Substituting $\boldsymbol{\tau}_{i-1} = \mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i \boldsymbol{\Sigma}_i \boldsymbol{g}_i$ into $\xi(\boldsymbol{\tau}_{i-1}, \boldsymbol{\tau}_i) \leq \eta_i(\alpha, N)$ and developing we get

$$\begin{aligned} -\frac{1}{2}(-2c_i^2 \boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i + c_i^2 \boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i) &\leq \log \eta_i(\alpha, N) \\ c_i &\leq \sqrt{\frac{2 \log \eta_i(\alpha, N)}{\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i}}. \end{aligned}$$

So combining the two inequalities we can take

$$c_i \leq \min \left(\sqrt{\frac{2 \log \eta_i(\alpha, N)}{\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i}}, \frac{R - ||\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty}{||\boldsymbol{\Sigma}_i \boldsymbol{g}_i||_\infty} \right). \tag{22}$$

Proof of (b): To find the minimum between the terms in Equation (22), we analyze the denominators and numerators. For the denominators, $\sqrt{\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i}$ and $||\boldsymbol{\Sigma}_i \boldsymbol{g}_i||_\infty = \max_j |(\boldsymbol{\Sigma}_i \boldsymbol{g}_i)_j|$, it is equivalent to compare $\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i$ and $||\boldsymbol{\Sigma}_i \boldsymbol{g}_i||_\infty^2$.

Since our diffusion model adopts a cosine scheduler for the covariance matrix, Σ_i is diagonal with elements $(\boldsymbol{\Sigma}_i)_{jj} \in [0, 1)$. We can write the $j-$th element of $\boldsymbol{\Sigma}_i \boldsymbol{g}_i$ as $(\boldsymbol{\Sigma}_i \boldsymbol{g}_i)_j = \boldsymbol{e}_j^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i$, where $\boldsymbol{e}_j$ is a basis vector. Then using Cauchy-Schwarz inequality we get

$$\begin{aligned} |(\boldsymbol{\Sigma}_i \boldsymbol{g}_i)_j|^2 &= |\boldsymbol{e}_j^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i|^2 \\ &\leq (\boldsymbol{e}_j^T \boldsymbol{\Sigma}_i \boldsymbol{e}_j)(\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i), \end{aligned}$$

with $\boldsymbol{e}_j^T \boldsymbol{\Sigma}_i \boldsymbol{e}_j \leq 1$ by definition of $\boldsymbol{\Sigma}_i$. It follows that $|(\boldsymbol{\Sigma}_i \boldsymbol{g}_i)_j|^2 \leq \boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i$, and since this is true for all j we can conclude that

$$\begin{aligned} \max_j |(\boldsymbol{\Sigma}_i \boldsymbol{g}_i)_j|^2 &\leq \boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i \\ ||\boldsymbol{\Sigma}_i \boldsymbol{g}_i||_\infty &\leq \sqrt{\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i} \end{aligned}$$

When comparing the numerators of both terms in Equation (22), since $\log \eta_i(\alpha, N) = \frac{1}{N} \log\left(\frac{1}{\alpha}\right)$, for N large enough, $\sqrt{2 \log \eta_i(\alpha, N)} < R - ||\mu_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty$.

So $c_i \leq \sqrt{\frac{2 \log \eta_i(\alpha, N)}{\boldsymbol{g}_i^T \boldsymbol{\Sigma}_i \boldsymbol{g}_i}}$ satisfies the dual CVaR constraints. □

D Adaptation to Matrix Normal Distribution

Here we extend the analysis to the case where states and actions are multidimensional, and we consider trajectories as Gaussian matrices.

We define $p(\boldsymbol{\tau}_{i-1} | \boldsymbol{\tau}_i)$ as

$$p(\boldsymbol{\tau}_{i-1} | \boldsymbol{\tau}_i) = \mathcal{M}N(\boldsymbol{\tau}_{i-1} | M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i), \boldsymbol{U}_i, \boldsymbol{V}_i)$$

where $\mathcal{M}N(\boldsymbol{\tau}_{i-1}|M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i), \boldsymbol{U}_i, \boldsymbol{V}_i)$ is a Matrix Normal Distribution with mean $M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) \in \mathbb{R}^{n\times p}$, row and column covariances $\boldsymbol{U}_i \in \mathbb{R}^{n\times n}$ and $\boldsymbol{V}_i \in \mathbb{R}^{p\times p}$.

The probability density function of this Matrix Normal Distribution is defined as

$$\mathcal{M}N(\boldsymbol{\tau}_{i-1}|M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i), \boldsymbol{U}_i, \boldsymbol{V}_i) \coloneqq K_i \exp\left(-\frac{1}{2}\text{Tr}[\boldsymbol{V}_i^{-1}(\boldsymbol{\tau}_{i-1} - M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))^T\boldsymbol{U}_i^{-1}(\boldsymbol{\tau}_{i-1} - M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))]\right)$$

where Tr[·] is the trace operator, $K_i = \frac{1}{(2\pi)^{np/2}|\boldsymbol{V}_i|^{n/2}|\boldsymbol{U}_i|^{n/2}}$ and $|\cdot|$ is the determinant of a matrix.

Define $\boldsymbol{G}_i \in \mathbb{R}^{n\times p}$ as the gradient $\nabla_{\boldsymbol{\tau}} Z$ with respect to the second order tensor representing the trajectory $\boldsymbol{\tau} \in \mathbb{R}^{n\times p}$ evaluated at $M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)$. Also define $\boldsymbol{\Gamma}_i = \boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i$ for notation convenience. Consider the perturbed distribution with a mean $M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i\boldsymbol{\Gamma}_i$, we get

$$\begin{aligned}
&p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{i-1}|\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1} \in C_\alpha) \\
&= \exp\left(-\frac{1}{2}\text{Tr}[\boldsymbol{V}^{-1}(\boldsymbol{\tau}_{i-1} - M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) + c_i\boldsymbol{\Gamma}_i)^T\boldsymbol{U}^{-1}(\boldsymbol{\tau}_{i-1} - M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) + c_i\boldsymbol{\Gamma}_i)]\right) \\
&= K \exp\left(-\frac{1}{2}\text{Tr}[\boldsymbol{V}^{-1}(\boldsymbol{\tau}_{i-1} - M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))^T\boldsymbol{U}^{-1}(\boldsymbol{\tau}_{i-1} - M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))]\right)\xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1})
\end{aligned}$$

with

$$\xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1}) = \exp\left(-\frac{1}{2}\text{Tr}[\boldsymbol{V}_i^{-1}(2c_i(\boldsymbol{\tau}_{i-1} - M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))^T\boldsymbol{U}_i^{-1}\boldsymbol{\Gamma}_i + c_i^2\boldsymbol{\Gamma}_i^T\boldsymbol{U}_i^{-1}\boldsymbol{\Gamma}_i)]\right).$$

As we did in Appendix C, we take $\xi(\boldsymbol{\tau}_i, \boldsymbol{\tau}_{i-1}) \le \eta_i(\alpha, N)$. We can take $\boldsymbol{\tau}_{i-1} = M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i\boldsymbol{\Gamma}_i$ and get

$$\begin{aligned}
\exp\left(-\frac{1}{2}\text{Tr}[\boldsymbol{V}_i^{-1}(2c_i(\boldsymbol{\tau}_{i-1} - M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i))^T\boldsymbol{U}_i^{-1}\boldsymbol{\Gamma}_i + c_i^2\boldsymbol{\Gamma}_i^T\boldsymbol{U}_i^{-1}\boldsymbol{\Gamma}_i)]\right) &\le \eta_i(\alpha, N) \\
-\frac{1}{2}\text{Tr}[\boldsymbol{V}_i^{-1}(-2c_i^2\boldsymbol{\Gamma}_i^T\boldsymbol{U}_i^{-1}\boldsymbol{\Gamma}_i + c_i^2\boldsymbol{\Gamma}_i^T\boldsymbol{U}_i^{-1}\boldsymbol{\Gamma}_i)] &\le \log\eta_i(\alpha, N) \\
\text{Tr}[\boldsymbol{V}_i^{-1}(\boldsymbol{\Gamma}_i^T\boldsymbol{U}_i^{-1}\boldsymbol{\Gamma}_i)c_i^2)] &\le 2\log\eta_i(\alpha, N),
\end{aligned}$$

giving

$$c_i \le \sqrt{\frac{2\log\eta_i(\alpha, N)}{\text{Tr}[\boldsymbol{V}_i^{-1}(\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i)^T\boldsymbol{U}_i^{-1}(\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i))]}}.$$

Under the same assumptions of Appendix C, we get that

$$\begin{aligned}
||\boldsymbol{\tau}_{i-1}||_\infty &\le R \\
||M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i) - c_i\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i||_\infty &\le R \\
||M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty + c_i||\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i||_\infty &\le R \\
c_i &\le \frac{(R - ||M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty)}{||\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i||_\infty}.
\end{aligned}$$

So we can pick

$$c_i = \min\left(\sqrt{\frac{2\log\eta_i(\alpha, N)}{\text{Tr}[\boldsymbol{V}_i^{-1}(\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i)^T\boldsymbol{U}_i^{-1}(\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i))]}}, \frac{(R - ||M_{\boldsymbol{\theta}}(\boldsymbol{\tau}_i, i)||_\infty)}{||\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i||_\infty}\right)$$

Following the same reasoning as in Appendix C, if the covariance matrices $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ are diagonal with elements in $[0, 1)$ we can pick

$$c_i = \sqrt{\frac{2\log\eta_i(\alpha, N)}{\text{Tr}[\boldsymbol{V}_i^{-1}(\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i)^T\boldsymbol{U}_i^{-1}(\boldsymbol{U}_i\boldsymbol{G}_i\boldsymbol{V}_i))]}}$$

E Implementation details

Our method makes use of three MLPs: the policy $\pi_{\boldsymbol{\omega}}$, the adversarial denoising diffusion model $\bar{p}_{\boldsymbol{\theta}}$ and the learned cumulative reward function $Z_{\boldsymbol{\phi}}$.

Policy network and training. The policy $\pi_{\boldsymbol{\omega}}$ is parameterized in the same way as PolyGRAD [38]. We consider a Gaussian policy of the form $\pi_{\boldsymbol{\omega}} = \mathcal{N}(\mu_{\boldsymbol{\omega}}(s), \sigma_{\boldsymbol{\omega}})$, where $\boldsymbol{\omega}$ are the parameters of the MLP. The standard deviation of the policy is a single learnable parameter $\sigma_{\boldsymbol{\omega}}$, independent of the state.

The policy is trained using Advantage Actor Critic (A2C) with Generalised Advantage Estimation (GAE). The optimizer used is ADAM. The hyperparameters can be found in Table 2.

Parameter	**Value**
Batch size	512
Synthetic trajectory length	10
GAE λ	0.9
Critic learning rate	3e-4
Actor learning rate	3e-5
Discount factor, γ	0.99
Entropy bonus weight	1e-5

Table 2: Hyperparameters for A2C training.

Adversarial Diffusion and Cumulative Reward models. Our implementation builds directly on top of PolyGRAD. We follow the same training procedure, summarized in Algorithm 3. For the Diffusion Model, we use the same MLP architecture as PolyGRAD, trained by minimizing the L2 loss with ADAM optimizer. The MDP has skip connections at every layer, and features a learneable embedding of the diffusion step i, which is common for Diffusion Architectures [17]. The hyperparameters are summarized in Table 3.

Parameter	**Value**
Hidden size	1024
Length of generated trajectory	10
Batch size	256
Diffusion step embedding size	128
Number of layers	6
Learning rate	3e-4

Table 3: Hyperparameters for adversarial diffusion training.

When computing c_i according to (18), we chose $R = 3\sigma_i$, where σ_i is the standard deviation of the diffusion process[2] at step i. In our implementation we choose $\eta_i(\alpha, N) = \left(\frac{1}{\alpha}\right)^{\frac{1}{N}}$.

The cumulative reward model $Z_{\boldsymbol{\phi}}$ follows the same structure and hyperparameters of the Diffusion Model (also the step embedding), with a final linear layer producing a scalar output. It is optimized using the L2 loss and the ADAM optimizer.

[2] In Denoising Diffusion Probabilistic Models, the standard deviation is fixed at every step i according to a known scheduling rate [15].

Algorithm 3 Diffusion model training

Input: adversarial denoising model $\bar{p}_{\boldsymbol{\theta}}$; cumulative reward function $Z_{\boldsymbol{\phi}}$; data buffer, $\mathcal{D}$; diffusion steps N; training iterations K
for $k = 1, \ldots, K$ **do**
    Improve $\bar{p}_{\boldsymbol{\theta}}$ (7) using $\{\boldsymbol{\tau}_0\} \sim \mathcal{D}$.
    Train $Z_{\boldsymbol{\phi}}$ to predict the reward $Z(\boldsymbol{\tau}_0)$.
end for

Baselines implementation The implementation of PolyGRAD was taken from the respective github repository [38]. The same was done for CPPO and M2TD3: for M2TD3, we created a new config file for Reacher, where we set the range of the actuator gear to $[50.0, 500.0]$. For TRPO and PPO we used the implementation from Stable-Baselines3 [36]. We used Domain Randomization on top of the PPO baseline, training with values of the mass and parameters uniformly sampled according to the uncertainty intervals specified in Table 4.

Environment	Mass	Friction	Mass pole	Mass cart	Act. gear
Hopper	[0.5, 6.5]	[0.1, 3.0]	—	—	—
HalfCheetah	[3.5, 9.5]	[0.2, 0.8]	—	—	—
Walker	[0.5, 6.5]	[0.5, 2.0]	—	—	—
Cartpole	—	—	[2.5, 10.0]	[5.0, 20.0]	—
Reacher	—	—	—	—	[50.0, 500.0]

Table 4: Uncertainty sets used for domain randomization.

Computational resources The training of AD-RRL and Polygrad was performed on three different machines. On a cluster node with one A100 GPU, Icelake CPU and 256 GB of RAM.

The remaining model-free baselines were trained on a laptop with an Intel i7-1185G7 CPU, Mesa Intel Xe Graphics GPU and 32 GB of RAM.

In table 5 we report the wall-clock training time for each algorithm . As it is expected, the model-based algorithms (AD-RRL and PolyGRAD) are slower than the model-free ones. This is a well-known shortcoming of Model-Based RL methods, even more so when using Diffusion Models, known for their longer training times when compared to standard MLPs. AD-RRL is slower than PolyGRAD since it employs an additional Diffusion Model to approximate the cumulative reward of a trajectory.

Algorithm	Hopper	Halfcheetah	Walker	InvertedPendulum	Reacher
AD–RRL (ours)†	3-20-00	3-20-00	3-20-00	3-20-00	3-20-00
Polygrad†	2-14-00	2-14-00	2-14-00	2-14-00	2-14-00
M2TD3‡	0-02-00	0-03-30	0-02-45	0-02-45	0-02-45
CPPO‡	0-00-30	0-00-30	0-00-30	0-00-30	0-00-30
PPO‡	0-00-30	0-00-30	0-00-30	0-00-30	0-00-30
TRPO‡	0-00-30	0-00-30	0-00-30	0-00-30	0-00-30
DR–PPO‡	0-00-30	0-00-30	0-00-30	0-00-30	0-00-30

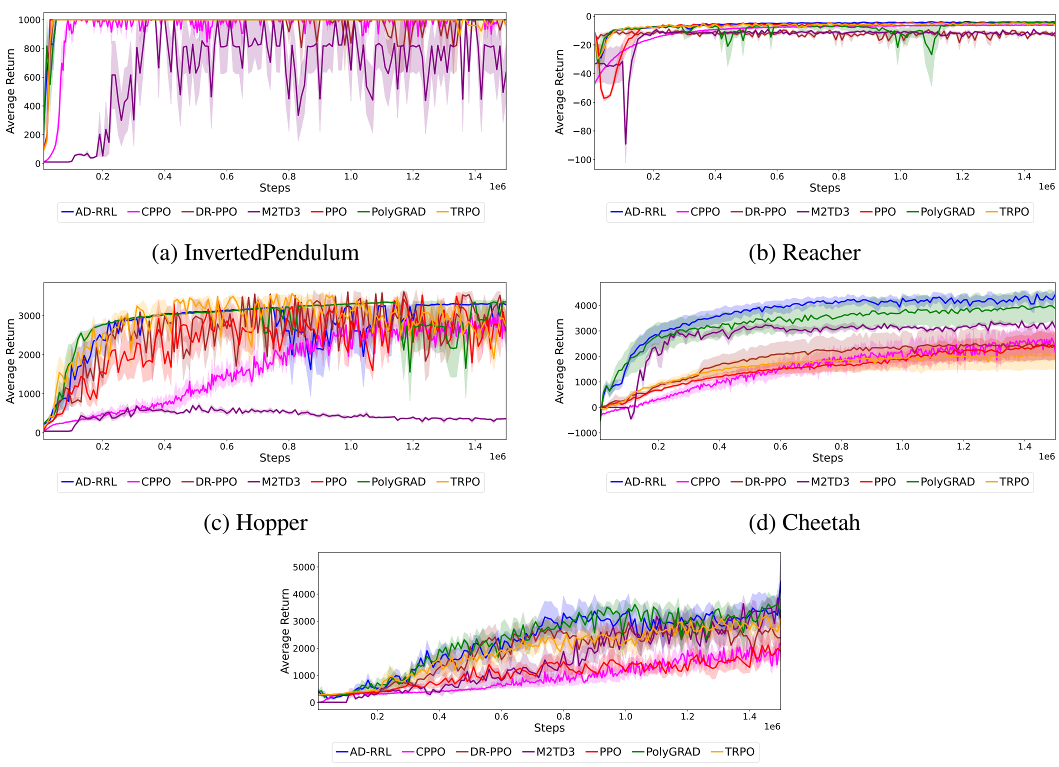
Table 5: Wall-clock training time (days–hours–minutes) needed to reach the reported performance on the MuJoCo tasks. Times are rounded up to the nearest quarter hour. †Trained on cluster node. ‡Trained on laptop.

F Additional Results

In this section, we provide additional plots to support our conclusions.

F.1 Training curves

Figure 2 shows the learning curves for AD-RRL and all the baselines for the considered MuJoCo tasks. Across seeds, AD-RRL reaches its final performance at least as quickly as the other methods. AD-RRL also achieves a final score matching or surpassing that of the baselines. This is particularly clear for the Cheetah and Walker environments, presented in Figures 2d and 2e. These results confirm that our method is more robust to modeling errors but does not sacrifice optimality in the training environment or learning speed.



(a) InvertedPendulum

(b) Reacher

(c) Hopper

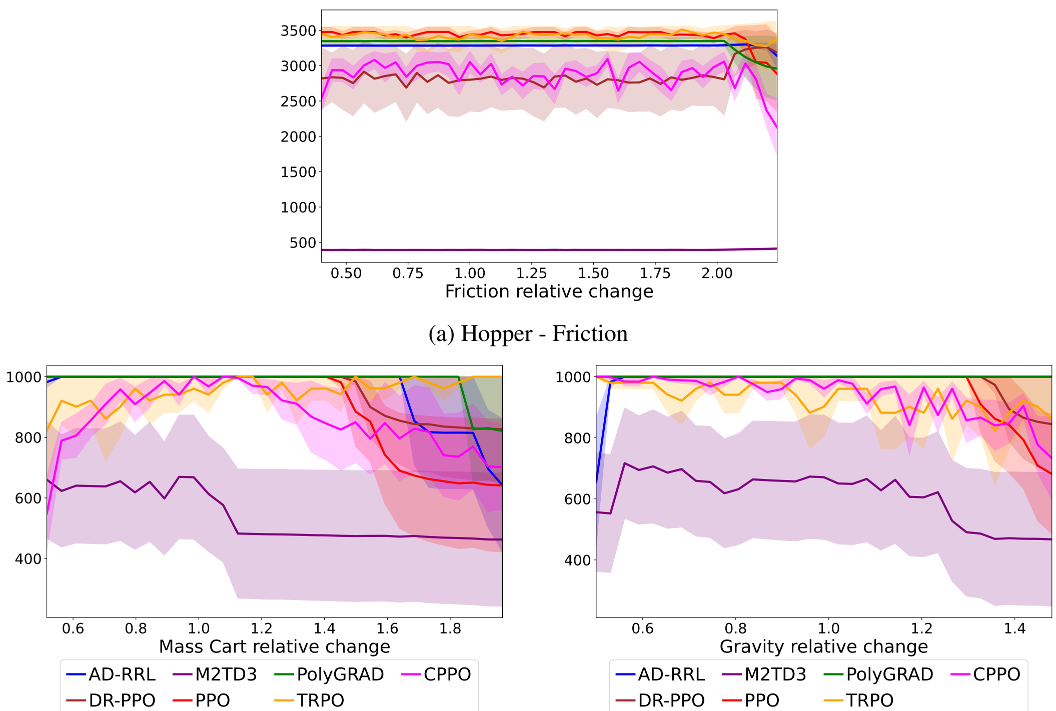
(d) Cheetah

(e) Walker

Figure 2: Training-return curves for five MuJoCo tasks. Shaded areas represent one standard error.

F.2 Varying parameters

In Figure 3 we present additional parameters variations for the InvertedPendulum and Cartpole environment. The pattern is consistent with the plots presented in Figure 1: AD-RRL achieves on par or higher returns than both robust and non-robust baselines as the dynamics deviate from nominal values. The only exception appears to be for higher variations of the cart mass, in the Inverted Pendulum environment (Figure 3b), where the additional inertia pushes most methods toward failure and AD-RRL similarly shows a performance decline.



(a) Hopper - Friction

(b) InvertedPendulum - Mass Cart

(c) InvertedPendulum - Gravity

Figure 3: Average Return for varying physical parameters. Shaded areas represent one standard error.

G Limitations and future work

Computation time. Guided diffusion requires dozens of reverse–diffusion steps for every synthetic trajectory and an extra gradient evaluation at each step. Consequently AD–RRL is slower than its precursor PolyGRAD, and both model–based methods need a higher training time (wall-clock) than the model-free baselines, even though they are more sample efficient. Improving the training time for diffusion models (e.g., fine-tuning the network size or the number of denoising steps) sounds like a natural next step.

Smooth–dynamics assumption. Our derivation employs a Gaussian approximation and the computation of gradients $\nabla_\tau Z(\tau)$. Both of these presuppose reasonably smooth rewards and state transitions. Some tasks might break this assumption, causing inaccurate guidance. Extending adversarial diffusion to domains with non-smooth dynamics is left for future work.

Scope of the evaluation. Our evaluation focused on simulated control tasks. A natural next step is a Sim2Real study. That is, AD–RRL is trained entirely in simulation and then deployed on real hardware, measuring how much the adversarial-diffusion training reduces the Sim2Real performance drop.

H Societal impact statement

Our contribution is methodological: we propose a technique for making model-based RL more robust to dynamics misspecification. Robustness is typically beneficial—e.g., safer robot control or fewer failures in medical-decision support—yet any improvement in sample efficiency or policy quality can also lower the barrier to deploying RL in high-stakes settings. In domains such as healthcare, finance, or autonomous driving, deployment must therefore be accompanied by domain-specific safety checks, bias audits, and human oversight. Our work does not introduce new data-collection practices, nor does it touch sensitive attributes, but it *could* be combined with decision pipelines that do. We encourage future users of this method to evaluate downstream ethical, legal, and safety implications before real-world deployment.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We specified all the claims and assumptions in the abstract and introduction. The claims are backed up by lemmata, propositions and empirical results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our algorithm in the Experiments and Conclusions and Future work sections.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: We provide the full set of assumptions in the main body of lemmata and propositions, while the complete proofs are provided in the appendix (and referenced in the main body).

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: In the appendix we provide the details regarding hyperparameters for our algorithm and baselines. We also provide pseudocode for our algorithm.

Guidelines:

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

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

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

Answer: [NA]

Justification: The code cannot be submitted in the workshop OpenReview form.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We provide hyperparameters in the appendix section.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Result tables and plots always include the standard error across all the simulations.

Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In the appendix we provide detailed information about the computing resources used to perform the experiments.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We reviewed the NeurIPS Code of Ethics and we do not use sensitive data.

   Guidelines:

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

10. **Broader impacts**

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

   Answer: [Yes]

   Justification: We discuss potential negative societal impacts in the appendix.

   Guidelines:

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

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: NA

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We credit all the authors of software used in the paper (like the experiments baselines).

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We provide details about our model (hyperparameters and pseudocode) and the code itself.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

    Justification: NA

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.