DARE: The Deep Adaptive Regulator for Control of Uncertain Continuous-Time Systems

Anonymous Author(s)
Affiliation
Address
email

Abstract

A fundamental challenge in continuous-time optimal control (OC) is the efficient computation of adaptive policies when agents act in unknown, uncertain environments. Traditional OC methods, such as dynamic programming, face challenges in scalability and adaptability due to the curse-of-dimensionality and the reliance on fixed models of the environment. One approach to address these issues is Model Predictive Control (MPC), which iteratively computes open-loop controls over a receding horizon. However, classical MPC algorithms typically also assume a fixed environment. Another approach is Reinforcement Learning (RL) which scales well to high-dimensional setups but is often sample inefficent. Certain RL methods can also be unreliable in highly stochastic continuous-time setups and may be unable to generalize to unseen environments. This paper presents the **D**eep **A**daptive **Re**gulator (DARE) which uses physics-informed neural network based approximations to the agent's value function and policy which are trained online to adapt to unknown environments. To manage uncertainty of the environment, DARE optimizes an augmented reward objective which dynamically trades off exploration with exploitation. We show that our method effectively adapts to unseen environments in settings where "classical" RL fails and is suited for online adaptive decision-making in environments that change in real time.

1 Introduction

Many real-world decision making problems in fields such as biology [24] and algorithmic trading [11], require agents to act at high-frequency in noisy systems, and hence can be modeled as continuous-time stochastic optimal control (OC) problems. Moreover, in many of these areas, the agent is uncertain of the system's true dynamics, which may be non-stationary, and the agent must learn these dynamics in real-time through interacting with the environment [17]. However, classical continuous-time OC methods for generating optimal policies, such as solving the standard Hamilton-Jacobi-Bellman (HJB) equation, generally assume known, fixed environments, limiting their wider application [55]. To address this challenge, *adaptive* control methods enable agents to optimize their performance by continuously learning and adapting to the unknown and evolving dynamics of their environment [8]. However, these methods typically rely on explicit parametric models of the environment [12], and the numerical methods required to solve these problems are often intractable [10].

One common approach to controlling *unknown* systems is Reinforcement Learning (RL), in which agents compute (near) optimal policies iteratively through trial and error [48]. However, without certain ad-hoc techniques such as action-repetition, certain RL methods can perform poorly in highly stochastic and nearly continuous-time setups, even when the environment is stationary; see [49, 54, 25]. Moreover, model-free RL methods suffer from sample inefficiency [3, 13] which can limit their applicability to real-world problems, whereas model-based methods that aim to mitigate

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

this drawback can require expert data that is not always accessible and are likely to lead to poor generalization [39].

Another control methodology for the control of uncertain systems is model predictive control (MPC). In MPC, agents optimize control inputs by solving a sequence of *open-loop* optimization problems based on a predictive model over a receding time horizon, allowing for real-time adjustments of the agent's policy; see [19]. While MPC is generally more computationally efficient than standard dynamic programming methods, which often require compute-intensive finite-difference schemes, standard methods of solving MPC problems still face computational bottlenecks, hindering their application to high-frequency environments [51]. For a more detailed review of relevant literature, see Appendix A. Hence, flexible methods that compute optimal policies efficiently in uncertain, non-stationary environments are desirable.

In this paper, we propose the Deep Adaptive Regulator (`DARE`). `DARE` is a deep learning-based method for solving stochastic continuous-time adaptive control problems. Our method consists of two distinct phases: offline and online. In both phases, physics-informed neural networks (PINNs) [42] parameterize the agent's value function and policy. During both phases, we optimize the value function and policy networks to satisfy the HJB equation resulting from the agent's estimate of the dynamics of the system and costs.

In the *offline* phase, the agent is endowed with an initial (potentially misspecified) estimate of the environment, which in practice may be learned from historical data. With this initial estimate, the agent constructs and solve an approximate OC problem which yields an initial policy. To accelerate training, we make use of two inductive biases: 1) we let the value function network learn the discrepancy between the true value function and the terminal condition of the HJB equation, and 2) we let the policy network learn the discrepancy between the true policy and a locally optimal policy generated by the Iterative Linear-Quadratic Gaussian (ILQG) method [51]. In the *online* phase, the agent implements their policy in the true environment and updates their estimate of the environment based on (potentially noisy) observations of the system and running costs. At each observation time, the agent updates their value function and policy by solving an updated OC problem over a receding horizon, similar to MPC. In contrast to MPC, which iteratively computes a sequence of open-loop controls to attain closed-loop feedback, `DARE` approximates the optimal closed-loop policy directly and updates this approximation in real-time. Moreover, to account for the agent's uncertainty about the environment, we consider a modification of the agent's objective function which dynamically balances exploration and exploitation.

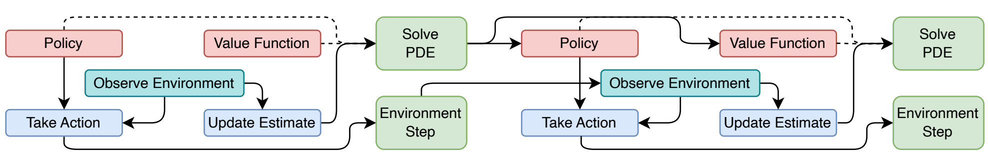

Figure 1: A schematic of `DARE`.

To benchmark `DARE`, we study its performance in three adaptive OC problems: (i) a one-dimensional Linear-Quadratic-Gaussian (LQG) Regulator problem in which the agent is uncertain of the drift of the system, (ii) an adaptive nonlinear MPC problem in which the agent is uncertain of the running cost and models it in real time with a Gaussian Process (GP), and (iii) a realistic high-dimensional nonlinear MPC problem motivated by algorithmic trading in finance [11]. While these problems are relatively simple, we demonstrate that in the LQG problem, a suite of classical RL methods including PPO [44], A2C [35], and SAC [20] are unable to learn robust policies when the time between actions is small, for a variety of action-repetition parameters. Moreover, we demonstrate that when RL methods are able to learn a robust solution, they require substantially more training time than `DARE`. In contrast, we show that `DARE` can learn accurate solutions and adapt to non-stationary environments in each problem.

In summary, this paper: (i) proposes the PINN-based method `DARE` to compute adaptive control policies in uncertain, continuous-time environments (ii) proposes an OC problem formulation that explicitly trades off exploration and exploitation which aids adaptation in non-stationary systems, (iii) demonstrates experimentally that classical RL methods fail on relatively simple continuous-time adaptive control tasks when time between actions is small, and (iv) proposes a inductive bias based on ILQG and the structure of the HJB which significantly accelerates offline training and improves speed of adaptation in non-stationary environments, outperforming classical RL.

2 Problem Formulation

Let $X_t \in \mathbb{R}^{d_X}$ be a stochastic system evolving continuously in time. We consider an agent who controls X with a policy $u_t \in \mathbb{R}^{d_u}$ over a fixed time horizon $T > 0$ to maximize a terminal reward $g : \mathbb{R}^{d_X} \to \mathbb{R}$. The agent's actions u_t on the system X_t incur a penalty modelled by a function $f : \mathbb{R}^{d_X} \times \mathbb{R}^{d_u} \to \mathbb{R}$, and their impact on the system dynamics is modelled by a drift function $h : \mathbb{R}^{d_X} \times \mathbb{R}^{d_u}$. The system evolves according to the dynamics

$$dX_t = h(X_t, u_t)\,\mathrm{d}t + \tilde{\Sigma}\,\mathrm{d}W_t\,, X_0 \in \mathbb{R}^{d_X}\,, \tag{1}$$

where W is a d_X-dimensional Brownian motion and $\tilde{\Sigma} \in \mathbb{R}^{d_X \times d_X}$ is a covariance matrix. We assume $\tilde{\Sigma}, T$ and g are fixed and known to the agent, and $\mathfrak{p} := (h, f)$ represents the modelling assumptions of the agent over the environment. We refer to $\mathfrak{p}$ as the *OC pair*.

Classical OC approaches assume a fixed and known pair $\mathfrak{p}$ to compute an optimal policy. In practice, the agent uses an uncertain estimate $\widehat{\mathfrak{p}}$ of the true environment. To account for this uncertainty, `DARE` solves the decision-making problem in two phases: offline and online. In the offline phase, the agent solves an OC problem according to an initial estimate of the environment $\widehat{\mathfrak{p}}_0$. In the online phase, the agent implements the policy from the offline phase in the true environment, receives noisy samples of the true OC pair, and updates their estimate of the environment. Then, the agent updates their control policy to be the solution of an updated OC problem according to the agent's new estimate of the environment. The agent is uncertain of their estimate, so it may be profitable to explore unknown areas of the system for potentially higher rewards. Hence, a balance must be struck between exploring new information and exploiting existing knowledge.

Offline phase. At time $t = 0$, the agent assumes an initial estimate $\widehat{\mathfrak{p}}_0 = (\widehat{h}_0, \widehat{f}_0)$ of the OC pair. To explicitly account for the exploration-exploitation trade-off, the agent seeks an optimal policy u^* which maximizes the following performance criterion:

$$J(s, x; u) = \mathbb{E}\Big[g(X_T) - \int_s^T \mathbb{E}\left[\widehat{f}_0\right](X_r, u_r)\,\mathrm{d}r + \phi \int_s^T \mathrm{Var}\left[\widehat{\mathfrak{p}}_0\right](X_r, u(r, X_r))\,\mathrm{d}r \;\Big|\; \mathcal{G}_s\Big]\,, \tag{2}$$

for all $s \in [0, T]$, where $\mathcal{G}_s$ is the information known to the agent at time s, $\mathbb{E}\left[\widehat{f}_0\right]$ denotes the mean prediction of the estimate $\widehat{f}_0$ and $\mathrm{Var}\left[\widehat{\mathfrak{p}}_0\right]$ is the sum of the variance of each each estimator in $\widehat{\mathfrak{p}}_0$, and we assume that X_s follows the dynamics

$$dX_s = \widehat{h}_0(X_s, u_s)\,\mathrm{d}s + \tilde{\Sigma}\,\mathrm{d}W_s, \quad X_0 \in \mathbb{R}^{d_X}\,. \tag{3}$$

The objective optimized by the agent in (2) is an adjusted formulation of the agent's true objective, in which the agent explicitly rewards or penalizes uncertainty on their estimate of the environment. More precisely, When $\phi > 0$ (resp. < 0) the agent rewards (resp. penalizes) exploration, i.e., the agent is encouraged to visit areas of the environment with higher (resp. lower) uncertainty. We show in Section 4.5 that the exploration parameter ϕ is key to the performance of decision-making problems in noisy and non-stationary environments.

The incorporation of the variance of the environment estimation in (2) is similar in spirit to the variance adjusted objective common in bandit algorithms such as [46]. However a notable difference is that the uncertainty in (2) is incorporated *dynamically* as opposed to myopically in the case of bandits.

To solve the problem (2), the agent defines the value function

$$V(s, x) = \sup_u J(s, x; u)\,. \tag{4}$$

We assume that the dynamic programming principle holds for $\mathbb{E}\left[\widehat{f}_0\right]$ and $\mathrm{Var}\left[\widehat{\mathfrak{p}}_0\right]$, so V solves the HJB equation:

$$0 = V_t + \frac{1}{2}\mathrm{Tr}(\Sigma\,\nabla_{xx} V) + \sup_{u \in \mathbb{R}^{d_u}} H(x, u, \nabla_x V(t, x); \widehat{\mathfrak{p}}_0)\,, \tag{5}$$

subject to terminal condition $V(T,x)=g(x)$, where $\Sigma=\tilde{\Sigma}\,\tilde{\Sigma}^{\intercal}$. For $\ell\in\mathbb{R}^{d_X}$, the Hamiltonian H in (5) is defined as

$$H(x,u,\ell;\widehat{\mathfrak{p}}_0)=\widehat{h}_0(x,u)^{\intercal}\ell+\mathbb{E}\left[\widehat{f}_0\right](x,u)-\phi\,\mathrm{Var}\left[\widehat{\mathfrak{p}}_0\right](x,u(t,x))\,. \tag{6}$$

The policy u^* which maximizes (2) can be computed using the following first order conditions for $s\in[0,T]$:

$$u^*(s,x;\widehat{\mathfrak{p}}_0)=\arg\max_{u\in\mathbb{R}} H(x,u,V_x(s,x);\widehat{\mathfrak{p}}_0)\,, \tag{7}$$

where $V(t,x)$ solves the nonlinear PDE (5). In contrast to several approaches in RL which address the exploration-exploitation trade-off through penalization or reward of *random* control processes (see [53] in a continuous-time setup), our method learns a control policy that is a *deterministic* function of the environment and which explores domain regions in which the agent is uncertain of their estimates of the OC pair.

Online Problem. At each time $t\in(0,T]$, the agent takes an action u_t, observes a noisy sample of the true environment $\mathfrak{p}(u_t,X_t)+\epsilon_t$ for some i.i.d. noise $\{\epsilon_t\}$, and updates their estimate $\widehat{\mathfrak{p}}_t$ accordingly.[1] Conditionally on the new estimate, the policy computed during the offline phase is not optimal. To adapt the optimal policy, the agent computes a new policy which maximizes the updated objective, for $s\in[t,T]$:

$$J(s,x;u)=\mathbb{E}\left[g(X_T)-\int_s^T\mathbb{E}\left[\widehat{f}_t\right](X_r,u_r)\,\mathrm{d}r+\phi\int_s^T\mathrm{Var}\left[\widehat{\mathfrak{p}}_t\right](X_r,u(r,X_r))\,\mathrm{d}r\,\middle|\,\mathcal{G}_s\right]. \tag{8}$$

This new control policy is then implented in the true system for $s\in[t,T]$ until the agent observes the system again, at which point the procedure is repeated. When the uncertainty $\mathrm{Var}(\widehat{\mathfrak{p}}_t)$ is high and the agent rewards exploration, i.e., $\phi<0$, the `DARE` policy focuses on improving the agent's estimate of their environment. As the estimation accuracy increases, the variance of the estimator and hence its contribution to the objective decreases. Hence, the `DARE` policy naturally balances the exploration-exploitation trade-off throughout the online phase. We demonstrate the benefit of the exploration term to overcome misspecified priors or nonstationary environments in Section 4.1. We note that simply optimizing the offline variance-adjusted objective (2) without updating the system and re-solving for a new optimal control would likely lead to sub-optimal solutions, as the variance penalty would be fixed for the entire time horizon. As far as we are aware, incorporation of model uncertainty into continuous-time stochastic control problems in this fashion is new to the literature.

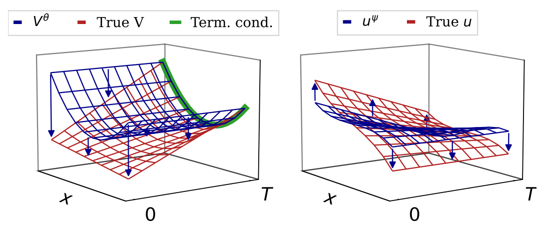

Figure 2: Illustration of the initialization of V^{θ_0}, u^{ψ_0} in the offline phase. The value function network is initialized around the terminal condition g in (2), and the control policy network is initialized around an affine approximation to the true optimal control.

3 The Deep Adaptive Regulator

A clear difficulty faced in the procedure above is the computation of the updated policy in the online phase. Even in low-dimensional settings, compute-intensive finite-difference schemes are required to solve the resulting HJBs [18]. Hence, we present `DARE`, which addresses this issue with PINN models of both teh value function and control policy.

Offline Phase. To obtain the initial policy corresponding to the pair $\widehat{\mathfrak{p}}_0$, we use PINN approximations $V^{\boldsymbol{\theta}}$ and $u^{\boldsymbol{\psi}}$ of the value function V and the optimal control u that solve the HJB (5). We

[1] We do not assume a particular estimation procedure, but this can be achieved with function approximators suitable for online learning, e.g., Gaussian Processes or Bayesian NNs as in [16].

initialize $V^{\boldsymbol{\theta}}$ and $u^{\boldsymbol{\psi}}$, for $s \in [0,T]$, as follows:

$$\begin{cases} V^{\boldsymbol{\theta}}(s,x) &= g(x) + \mathfrak{X}^{\boldsymbol{\theta}}(s,x), \\ u^{\boldsymbol{\psi}}(s,x) &= \widehat{u}_{X_0}(s,x) + \mathfrak{X}^{\boldsymbol{\psi}}(s,x), \end{cases} \tag{9}$$

where g is the terminal reward function and $\widehat{u}_{X_0}$ is a locally optimal linear approximation of the true optimal control. We use the ILQG method of [51] to compute $\widehat{u}_{X_0}$ starting from X_0 and we set $\mathfrak{X}^{\boldsymbol{\theta}}$ and $\mathfrak{X}^{\boldsymbol{\psi}}$ to be fully-connected feedforward networks; see Figure 2.

To train $V^{\boldsymbol{\theta}}$ and $u^{\boldsymbol{\psi}}$, we devise a multi-objective PINN loss which considers (i) the HJB (5), (ii) the Hamiltonian (6) satisfying first-order conditions, and (iii) the terminal condition. We use Monte Carlo integration to minimize the loss function on a compact domain $K \subset \mathbb{R}^{d_X}$. We let $\|\cdot\| = \|\cdot\|_{L^2([0,T]\times K)}$ and we reformulate (5) in a variational form to define the loss:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}; \widehat{\mathfrak{p}}) = \mathcal{L}_{\text{HJB}} + \mathcal{L}_{\text{hamiltonian}} + \mathcal{L}_{\text{terminal}}\,, \tag{10}$$

where

$$\begin{cases} \mathcal{L}_{\text{HJB}} = \|V_t^{\boldsymbol{\theta}} + \frac{1}{2}\text{Tr}(\Sigma\, \nabla_{xx} V^{\boldsymbol{\theta}}) + H(\cdot, u^{\boldsymbol{\psi}}(\cdot,\cdot), V_x^{\boldsymbol{\theta}}(\cdot,\cdot); \widehat{\mathfrak{p}})\|\,, \\ \mathcal{L}_{\text{hamiltonian}} = \|\partial_u H(\cdot, u^{\boldsymbol{\psi}}(\cdot,\cdot), V_x^{\boldsymbol{\theta}}(\cdot,\cdot); \widehat{\mathfrak{p}})\|\,, \\ \mathcal{L}_{\text{terminal}} = \|V^{\boldsymbol{\theta}}(T,\cdot) - g\|\,. \end{cases} \tag{11}$$

The loss (10) is similar to that in [1], however, here we use a first-order condition for the Hamiltonian component, which we found to improve training performance when H is concave. Otherwise, we set

$$\mathcal{L}_{\text{hamiltonian}} = -\|H(\cdot,\cdot,u^{\boldsymbol{\psi}}; \widehat{\mathfrak{p}})\|\,. \tag{12}$$

We summarize the procedure in Algorithm B.

Online Phase. Let $\mathcal{T} = \{t_0, \ldots, t_n\} \subset [0,T]$ be a set of potentially irregularly spaced times which are unknown to the agent at time $t = 0$. In the online phase, the agent uses new estimates of the environment to update their control policy as follows. Suppose the agent calculated $V^{\boldsymbol{\theta}_{t_{k-1}}}(\cdot,\cdot;\widehat{p}_{t_{k-1}})$ and $u^{\boldsymbol{\psi}_{t_{k-1}}}(\cdot,\cdot;\widehat{p}_{t_{k-1}})$ at time t_k, where $\boldsymbol{\theta}_{t_{k-1}}$ and $\boldsymbol{\psi}_{t_{k-1}}$ minimize the loss $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}, \widehat{p}_{t_{k-1}})$. At time t_k, the agent (i) takes the action $u^{\boldsymbol{\psi}_{t_{k-1}}}(t_k, X_{t_k}; \widehat{p}_{t_{k-1}})$ and (ii) computes the new estimate $\widehat{p}_{t_k}$. Then, over the period $[t_k, t_{k+1})$, the agent minimizes $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\psi}, \widehat{p}_{t_k})$ to compute the parameters $(\boldsymbol{\theta}_{t_k}, \boldsymbol{\psi}_{t_k})$. This loss minimization uses a gradient-based method (e.g., ADAM), with $\big(\boldsymbol{\theta}_{t_{k-1}}, \boldsymbol{\psi}_{t_{k-1}}\big)$ as a warm start for the neural networks; our method is outlined in Figure 1 and Algorithm B.

4 Numerical Experiments

This section investigates the performance of `DARE` in both the offline and online phases. Three control tasks are considered, including Linear-Quadratic Gaussian (LQG) control, an augmentation of the LQG problem with uncertain running costs, and a high-dimensional control task motivated by algorithmic trading in finance. First, we demonstrate the ability of `DARE` to learn accurate solutions in the offline phase. We then present a failure mode of classical RL methods in continuous-time control tasks, demonstrating degradation of performance in the simple LQG control problem. We also investigate the impact of the variance-adjusted objective in the MPC problem when running cost observations are corrupted by noise. Finally, we present a comparison of the sample efficiency of `DARE` compared to RL methods, demonstrating vastly improved training times.

4.1 Description of Control Tasks

Linear-Quadratic-Gaussian. Consider system dynamics

$$\mathrm{d}X_t = (b + c\, u_t)\,\mathrm{d}t + \tilde{\Sigma}\,\mathrm{d}W_t\,, \quad X_0 \in \mathbb{R}, \tag{13}$$

where b is a constant drift, $c > 0$ scales the linear impact of an agent on the system, and $\tilde{\Sigma} > 0$ is the variance of the observation noise. The agent maximizes the LQ criterion

$$\mathbb{E}\left[X_T - \alpha\, X_T^2 - \phi \int_0^T u_t^2\, \mathrm{d}t\right]\,, \tag{14}$$

where $\phi > 0$ scales the running quadratic penalty and $\alpha > 0$ scales the terminal quadratic penalty.

Model Predictive Control. We consider an augmentation of LQG in which the agent is subject unknown running cost and must model this cost in real-time. In particular, the dynamics of the system evolves according to (13), and the true running cost of the system is quadratic as in (14). The agent uses a Gaussian Process (GP) $\widehat{f}$ to model the true running cost, and updates this model in real time using noisy observations of accumulated running costs. Hence, in both the offline and online phase of training, the DARE optimizes the objective

$$\mathbb{E}\left[X_T - \alpha X_T^2 - \phi \int_0^T \mathbb{E}[\widehat{f}](u_t)\,\mathrm{d}t - \varphi \int_0^T \mathrm{Var}[\widehat{f}](u_t)\mathrm{d}t\right]. \tag{15}$$

High-Dimensional Control. For a high-dimensional control example, we consider an example from finance. In recent years, regulators have urged financial institutions to manage the risk of their trading activity within very large portfolios called central risk books. The aggregated trading activity of large institutions is often conducted at very high frequency and can be modeled as an OC problem. The controlled system is described by the agent's inventory $Q_t \in \mathbb{R}^d$, the asset prices $S_t \in \mathbb{R}^d$, and running wealth $X_t \in \mathbb{R}$, with dynamics:

$$\mathrm{d}Q_t = u_t\,\mathrm{d}t\,,\ \mathrm{d}S_t = \tilde{\Sigma}\,\mathrm{d}W_t\,,\ \mathrm{d}X_t = -u_t^{\intercal} S_t\,\mathrm{d}t - f(u_t)\,\mathrm{d}t\,,$$

where $u_t \in \mathbb{R}^d$ denotes the trader's speed of trading. The agent incurs transaction costs according to some unknown function of the trading speed $f(u_t) \in \mathbb{R}^d$, and maximizes the exponential utility of their terminal wealth for some estimate $\widehat{f}$ of the true transaction costs

$$\sup_v \mathbb{E}\left[-\exp\left(-\gamma\left(X_T + Q_T^{\intercal} S_T - Q_T^{\intercal}\,\Gamma\, Q_T\right)\right)\right],$$

In this example, we set $d = 5$. In Appendix H we record the model parameters η and $\tilde{\Sigma}$ and include a detailed motivation for this problem.

4.2 Offline Performance

First, we investigate the performance of DARE in the offline phase. To do so, we fix the agent's prior, and demonstrate the ability of DARE to learn accurate solutions to the HJB equation posed with this prior. In particular, we investigate the impact of the inductive biases used to augment the value function and control policy defined in (9). We compare DARE to PINNs without such bias, that is, we consider two methods, MLP and DGM with initializations

$$V^{\boldsymbol{\theta}}(t,x) = \mathfrak{X}^{\boldsymbol{\theta}}(t,x) \quad \text{and} \quad u^{\boldsymbol{\psi}}(t,x) = \mathfrak{X}^{\boldsymbol{\psi}}(t,x)\,,$$

where $\mathfrak{X}^{\boldsymbol{\theta}}$ and $\mathfrak{X}^{\boldsymbol{\psi}}$ are feedforward DNNs with Xavier initialization in MLP and LSTM-like networks as in the Deep Galerkin Method [45] in DGM.

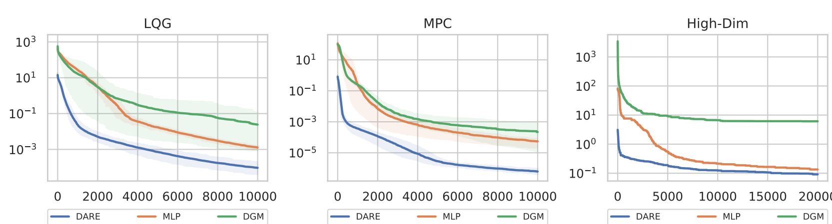

Figure 3: We plot the training loss trajectory in the offline phase for DARE, MLP, and DGM. In the MPC problem, we set $b = 0$, and other model parameters are given in Appendix C. In the LQG and MPC figures, we take the average of 100 seeds.

In each of problems, the MLPs of both DARE and MLP, there are 2 layers and 20 hidden units. In DGM, there are two hidden LSTM-like layers between two single layer feedforward neural networks of width 20. Additional details about the parameters for each problem in this experiment are found in Appendix C. Figure 3 shows the loss (10) of the three methods throughout offline training when the agent's prior is fixed. On average, we observe that DARE substantially outperforms other architectures in convergence speed in all three problems, achieving lower loss with fewer iterations. While existing work [45] emphasizes the importance of network architecture for performance, our findings indicate that, at least for simple problems, inductive biases may hold greater importance.

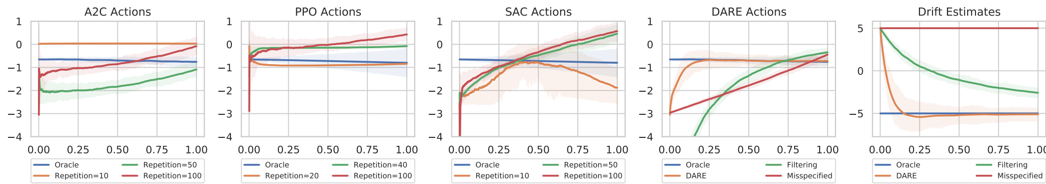

Figure 5: We plot the actions each of algorithm in the online phase, along with the drift estimation used by the RL agents and DARE. For each RL method, we plot actions corresponding to the top three performing methods in Figure 4. We also include continuous-time Kalman filtering using a prior $b \sim \mathcal{N}(5, 3)$, which is calculated using techniques in Appendix G.

4.3 Failure Mode of Reinforcement Learning in Continuous-Time Systems

In [49], the authors find that Q-learning based RL methods struggle in near continuous-time environments, as the Q-value contribution of a single action vanishes with a shrinking discretization timestep. Here, we demonstrate that a suite of RL methods, including Proximal Policy Optimization (PPO) [44], Advantage Actor Critic (A2C) [35], and Soft Actor Critic (SAC) [20] struggle in even the most simple continuous-time stochastic control setting, the LQG problem, when the time discretization is very small.

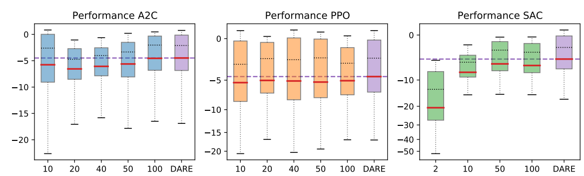

Figure 4: Average total reward of PPO, A2C, SAC, and DARE in the LQG problem across 1000 rollouts. The oracle average reward is denoted by the dashed line and is calculated using techniques outlined in Appendix G. Each agent is trained offline using a misspecified prior of $b_0 = 5$ in the case of DARE and $b \sim \mathcal{N}(b_0, \Pi_0)$ for the RL agents, and upon rollout, uses an exponential moving average to estimate the true drift $b \sim \mathcal{N}(-5, 3)$ of the system.

In the LQG problem the drift b is unknown to the agent. To train the RL agent, we simulate trajectories of the state process in (13) using samples of the drift b from some prior distribution. We then let the drift estimation be a state variable, that is, we let the policy of each agent $u^\psi = u^\psi(t, x; b)$. To estimate the drift, at each observation time t in the online phase, each agent uses an exponential moving average with smoothing $\lambda = 0.95$ to determine an estimate b_t. To test each agent's ability to adapt, we consider the case of a misspecified prior distribution where the drift is sampled from $b \sim \mathcal{N}(b_0, \Pi_0)$, where $b_0 = 5$ and $\Pi_0 = 3$. To implement DARE in this environment, we use the same drift estimation, and after each observation, we use 10 ADAM steps to update the DARE policy. Additional details regarding our implementation and hyperparameters used for A2C, PPO, and SAC are included in Appendix J.

In Figure 4, we plot the performance of each RL agent when the system is simulated with the true drift $b = 5$ and the agent must estimate the drift in run time. We discretize the problem horizon into 1000 steps and investigate the effect of varying number of times actions are repeated. When actions are repeated less frequently, we observe that the performance of each RL algorithm steadily degrades. DARE, however, does not require action-repetition during training, and achieves similar performance to the best performing RL method, which required extensive hyperparameter tuning.

In Figure 5, we picked the top three performing action-repetition hyperparameters for each RL method and plotted the actions of these agents in the online phase. Note that while A2C with 100 repeated actions is the top performing method, the actions produced by this method are quite far from the oracle. On the other hand, DARE learns the oracle control policy quickly after only a few environment steps.

4.4 Real Time Adaptation in Nonlinear Environments

Here, we investigate the performance of the online phase of DARE in the MPC and high-dimensional control problems. To test the ability of DARE to adapt in these settings, we first train DARE to a misspecified prior in the offline phase. Then, we suppose the agent learns the true environment parameters, and we test how many iterations are required to adjust the misspecified policy to the optimal policy of the true environment.

For the MPC problem, We define two running penalty functions $f_i = |u|^{1+\gamma_i}$ for $i \in \{0, 1\}$, where $\gamma_0 = 1.3$ and $\gamma_1 = 1$. We consider agents who use a Gaussian Process (GP) $\widehat{f}$ as a predictive model for the running penalty; see Appendix I for details on GPs. First, the agents fit two Gaussian Process $\widehat{f}_i$ for $i \in \{0, 1\}$ to ten noisy, random samples of the running penalty f_i. Next, we use DARE, MLP, and DGM to solve for solutions $V^{\boldsymbol{\theta}_0}, u^{\boldsymbol{\psi}_0}$ relative to $\widehat{f}_0$. Once all methods have converged, we change the agents' estimate of the running penalty to $\widehat{f}_1$ and re-train $V^{\boldsymbol{\theta}_0}, u^{\boldsymbol{\psi}_0}$ with this updated penalty function. For the high-dimensional control problem, we first train each algorithm using a running cost of $f(u) u^{\gamma \intercal} \eta u^{\gamma}$ where the exponent $\gamma = 1.3$ is applied element-wise, and then we change the exponent to $\gamma = 1$.

In Figure 6, we plot the training loss after adjusting the running costs. In the MPC problem, all methods learn the new policy with comparable precision after a few hundred iterations. In the high-dimensional problem, DGM is unable to adapt to the new environment. DARE, however, is able to adapt to the new environment faster than the other methods, requiring less than 20 ADAM steps to achieve satisfactory precision. Each iteration lasts 0.00446 seconds on average in our experiments so DARE is suited for online problems with near continuous observations in nonstationary environments.

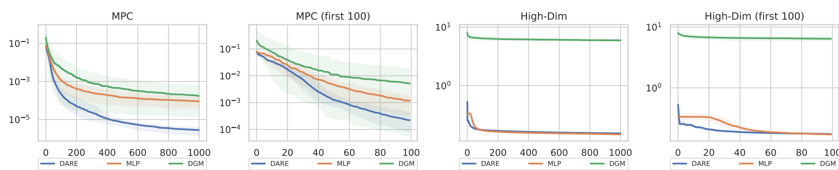

Figure 6: We plot the training loss trajectory in the online phase DARE, MLP, and DGM when the prior in each case is $\gamma = 1.3$ and the true environment is $\gamma = 1$.

4.5 Online Performance: Exploration-Exploitation in Non-Stationary Environments

We now focus on the MPC problem above. We consider the online phase and investigate when the agent's real-time observations of the running penalty are corrupted by noise. That is, the agent observes $|u^{\boldsymbol{\psi}}(t, X_t)|^2 + \epsilon_t$ for $\epsilon_t \sim \mathcal{N}(0, .02)$. In this setting, we investigate whether it is beneficial for the agent to explore to ensure they have an accurate model of the running penalty function. We examine the effect of varying the variance penalty weight φ in (15). In particular, we test DARE with $\varphi = 0$ and $\varphi < 0$, which corresponds to an agent which is indifferent to exploration and encourages exploration, respectively.

Figure 7 shows that in the presence of noise, encouraging exploration in the objective enables the agent to learn the true policy far faster than being indifferent to exploration. Often, the absence of exploration in noisy environments leads to local optima in the value function and control policy network parameters, because a (wrong) mean prediction leads to a specific policy which prevents accurate learning of the cost function in the whole domain of controls.

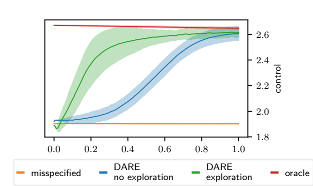

Figure 7: Mean and std dev of policy for `oracle`, `misspecified`, and DARE when $\varphi = 0$ (no exploration), and when $\varphi = 5 \cdot 10^{-3}$ (exploration).

Finally, we consider a simulation setup where the true form of the cost function randomly switches between that of $\gamma_1^* = 1.3$ and $\gamma_2^* = 1$ according to a Poisson process with intensity 0.005, i.e., with 1.5 switches, on average, per simulation. We consider an environment which starts with the cost functional $\gamma_1^* = 1.3$, and the observations of the running penalty $|u^{\boldsymbol{\psi}}(t, X_t)|^{1+\gamma^*} + \epsilon_t$ are corrupted with noise $\epsilon_t \sim \mathcal{N}(0, .1)$. Similar to the previous experiment, the agent uses a GP to model the running penalty. In Figure 8 one sees that DARE quickly adapts to the new environment after each jump in the running cost exponent.

We note that our decision to exclude an RL benchmark in this case was informed by two factors. First, we demonstrate above that the RL algorithms considered struggle even in the LQG setting, suggesting poor performance in nonlinear problems. Second, in the MPC problem, the agent uses a GP to model uncertainty. In the LQG problem, uncertainty about the drift was parametric and could be incorporated as a state variable for the RL agent. Hence, the RL agent could be trained on samples of different drifts and learn the optimal policy for all drifts in the sampled region. In the MPC problem, the GP estimate could not be incorporated directly as a state variable for the RL agent, because the GP is a non-parametric function approximator. Tailoring an RL algorithm to optimize over a set of non-parametric cost functions was out of the scope of this paper.

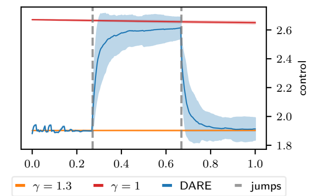

Figure 8: Mean and std dev of policy for `DARE` when the true value of γ jumps between 1.3 and 1.

4.6 Sample Efficiency

Finally, we show that `DARE` is more computationally efficient than the SOTA RL algorithms tested. In Table 1, we report the wall-clock training times of the offline phase each of the best performing RL methods and `DARE` in the LQG task. Clearly, `DARE` learns an optimal solution much faster than each of the RL methods. As the RL methods tested here are model-free, this is expected, as each algorithm must also learn from scratch the dynamics corresponding to the agent's prior over the system. We leave the comparison of `DARE` to more model-based RL methods for future work.

Table 1: Average Wall-Clock Runtime Comparison, see Appendix J.

Algorithm	Wall-clock time in sec.	Device
A2C	451	NVIDIA A40
PPO	550	NVIDIA A40
SAC	1629	NVIDIA A40
`DARE`	22	Apple M1

5 Conclusions, Limitations, and Future Work

This paper presents `DARE`, a PINN-based methodology for solving OC problems in noisy and non-stationary environments. We demonstrate that when the time between actions is small, classical RL methods struggle in continuous-time control settings. Moreover, these methods are unstable with respect to action-repetition. In contrast, `DARE` does not require a fixed time discretization scheme and is shown to learn accurate solutions in stationary environments. Additionally, with the incorporation of a variance-adjusted objective which explicitly trades off exploration and exploitation, we show that `DARE` can efficiently adapt to non-stationary environments in real time.

A limitation of this work is the restriction of our experiments to simple control problems. Justification for our choice of problem settings is twofold. First, we sought to demonstrate that even simple continuous-time control problems prove difficult for classical RL methods without ad-hoc techniques. With these techniques, the RL methods considered were still unable to learn the true optimal control policy. Second, we sought OC problems for which classical solutions were available in order to verify that `DARE` indeed learns the true solution. We also wish to highlight that while the MPC environment appears simple, generating solution via classical methods is by no means trivial, as the system does not admit a dimensionality reducing ansatz. We acknowledge the need for further experimentation of `DARE` in physical systems and intend to pursue this in future work.

Also left for future work is a more unified methodology for system identification. In the LQG problem, the dynamics of the system could be learned quickly because of the problem's simple linear structure. For more nonlinear problems, this estimation can be far more difficult. In future work, we intend to integrate `DARE` with more flexible methods for learning system dynamics.

References

[1] A. Al-Aradi, A. Correia, G. Jardim, D. de Freitas Naiff, and Y. Saporito. Extensions of the deep galerkin method. *Applied Mathematics and Computation*, 430:127287, 2022.

[2] F. Allgower, R. Findeisen, Z. K. Nagy, et al. Nonlinear model predictive control: From theory to application. *Journal-Chinese Institute Of Chemical Engineers*, 35(3):299–316, 2004.

[3] A. S. Anand, J. E. Kveen, F. Abu-Dakka, E. I. Grøtli, and J. T. Gravdahl. Addressing sample efficiency and model-bias in model-based reinforcement learning. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1–6. IEEE, 2022.

[4] A. C. Aristotelous, E. C. Mitchell, and V. Maroulas. Adlgm: An efficient adaptive sampling deep learning galerkin method. *Journal of Computational Physics*, 477:111944, 2023.

[5] A. Bachouch, C. Huré, N. Langrené, and H. Pham. Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications. *Methodology and Computing in Applied Probability*, 24(1):143–178, 2022.

[6] M. Basei, X. Guo, A. Hu, and Y. Zhang. Logarithmic regret for episodic continuous-time linear-quadratic reinforcement learning over a finite-time horizon. *The Journal of Machine Learning Research*, 23(1):8015–8048, 2022.

[7] A. Bensoussan. Perturbation methods in optimal control. *(No Title)*, 1988.

[8] T. Bhudisaksang and A. Cartea. Adaptive robust control in continuous time. *SIAM Journal on Control and Optimization*, 59(5):3912–3945, 2021.

[9] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz. Deep model predictive flow control with limited sensor data and online learning. *Theoretical and computational fluid dynamics*, 34:577–591, 2020.

[10] T. Björk, M. H. Davis, and C. Landén. Optimal investment under partial information. *Mathematical Methods of Operations Research*, 71:371–399, 2010.

[11] Á. Cartea, S. Jaimungal, and J. Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.

[12] S. N. Cohen, C. Knochenhauer, and A. Merkel. Optimal adaptive control with separable drift uncertainty. *arXiv preprint arXiv:2309.07091*, 2023.

[13] F. E. Dorner. Measuring progress in deep reinforcement learning sample efficiency. *arXiv preprint arXiv:2102.04881*, 2021.

[14] K. Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1): 219–245, 2000.

[15] F. Drissi. Solvability of differential riccati equations and applications to algorithmic trading with signals. *Applied Mathematical Finance*, 29(6):457–493, 2022. doi: 10.1080/1350486X.2023.2241130. URL `https://doi.org/10.1080/1350486X.2023.2241130`.

[16] G. Duran-Martin, A. Kara, and K. Murphy. Efficient online bayesian inference for neural bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 6002–6021. PMLR, 2022.

[17] D. Evangelista and Y. Thamsten. Approximately optimal trade execution strategies under fast mean-reversion. *arXiv preprint arXiv:2307.07024*, 2023.

[18] P. A. Forsyth and G. Labahn. Numerical methods for controlled hamilton-jacobi-bellman pdes in finance. *Journal of Computational Finance*, 11(2):1, 2007.

[19] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

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

[22] M. Hoglund, E. Ferrucci, C. Hernandez, A. M. Gonzalez, C. Salvi, L. Sanchez-Betancourt, and Y. Zhang. A neural rde approach for continuous-time non-markovian stochastic control problems, 2023.

[23] C. Huré, H. Pham, A. Bachouch, and N. Langrené. Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis. *SIAM Journal on Numerical Analysis*, 59(1):525–557, 2021.

[24] P. A. Iglesias and B. P. Ingalls. *Control theory and systems biology*. MIT press, 2010.

[25] Y. Jia and X. Y. Zhou. Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms. *The Journal of Machine Learning Research*, 23(1):12603–12652, 2022.

[26] Y. Jia and X. Y. Zhou. q-learning in continuous time. *Journal of Machine Learning Research*, 24(161):1–61, 2023.

[27] D. Jiang, J. Sirignano, and S. N. Cohen. Global convergence of deep galerkin and pinns methods for solving partial differential equations. *arXiv preprint arXiv:2305.06000*, 2023.

[28] K. Kunisch and D. Walter. Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation. *ESAIM: Control, Optimisation and Calculus of Variations*, 27:16, 2021.

[29] I. Lenz, R. A. Knepper, and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, volume 10, page 25. Rome, Italy, 2015.

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

[31] A. Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[32] H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*, 8(1):164–177, 1996.

[33] P. K. Mishra, M. V. Gasparino, A. E. B. Velasquez, and G. Chowdhary. Deep model predictive control, 2023.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[36] 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.

[37] S. Niu, Y. Liu, J. Wang, and H. Song. A decade survey of transfer learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, 1(2):151–166, 2020.

[38] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto. A neural network approach for high-dimensional optimal control applied to multiagent path finding. *IEEE Transactions on Control Systems Technology*, 31(1):235–251, 2022.

[39] E. C. Ozcan, V. Giammarino, J. Queeney, and I. C. Paschalidis. A model-based approach for improving reinforcement learning efficiency leveraging expert observations. *arXiv preprint arXiv:2402.18836*, 2024.

[40] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[41] 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. URL `http://jmlr.org/papers/v22/20-1364.html`.

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

[43] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll. Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, apr 2023. doi: 10.1109/lra.2023.3246839. URL `https://doi.org/10.1109%2Flra.2023.3246839`.

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

[45] J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[46] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[47] P. M. Suder, J. Xu, and D. B. Dunson. Bayesian transfer learning. *arXiv preprint arXiv:2312.13484*, 2023.

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

[49] C. Tallec, L. Blier, and Y. Ollivier. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, pages 6096–6104. PMLR, 2019.

[50] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pages 270–279. Springer, 2018.

[51] E. Todorov and W. Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306. IEEE, 2005.

[52] R. van der Meer, C. W. Oosterlee, and A. Borovykh. Optimally weighted loss functions for solving pdes with neural networks. *Journal of Computational and Applied Mathematics*, 405: 113887, 2022.

[53] H. Wang, T. Zariphopoulou, and X. Y. Zhou. Reinforcement learning in continuous time and space: A stochastic control approach. *The Journal of Machine Learning Research*, 21(1): 8145–8178, 2020.

[54] C. Yildiz, M. Heinonen, and H. Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, pages 12009–12018. PMLR, 2021.

[55] J. Yong and X. Y. Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*, volume 43. Springer Science & Business Media, 1999.

[56] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A Related Work

Deep Learning Methods for Control. Deep learning is extensively used to solve HJB equations because of its flexibility and scalability to high dimensions [23, 5, 38, 28]. Among earlier examples, [21, 45] provide two contrasting approaches. The former proposes the Deep Galerkin Method, which uses Monte Carlo integration to minimize a variational form of the HJB in a mesh-free manner, while the latter proposes the Deep BSDE method, which reformulates the PDE as a backward stochastic differential equation and approximates the gradient of the solution with a neural network. Global convergence of DGM was recently established in [27]. There are numerous extensions to their method, including adaptive Monte Carlo sampling in [4], augmented loss function for non-parametric running penalties and drifts in [1], and optimally weighted loss objectives in [52].

Model Predictive Control. Classical methods in MPC are the foundation of many online control optimization methods in both the deterministic and stochastic settings [2, 31]. Recently, deep learning was integrated with MPC, with applications in controlling uncertain nonlinear systems such as unsteady fluid flow and high-performance autonomous systems [29, 33, 9, 43, 36]. These approaches leverage neural networks to enhance dynamic modeling capacity and real-world control performance.

Continuous-Time Reinforcement Learning. Methods in deep RL are highly effective in several complex decision-making problems [34, 30, 44]. Continuous-time environments pose significant challenges to RL methods [49, 54]. In particular many RL methods optimize incorrect objectives [see 25] when environments are noisy, e.g., temporal difference (TD) learning [14]. Recently, [53] study the exploration-exploitation trade-off in stochastic and continuous-time RL, and prove that the optimal exploration policy is Gaussian in a Linear-Quadratic setting. Subsequent work [25, 26, 22, 6] extend RL methods to stochastic and continuous-time environments.

B Algorithms

Algorithm 1 : `DARE` - Offline Phase

Inputs:
- Initial OC pair $\widehat{\mathfrak{p}}_0 = (\widehat{h}_0, \widehat{f}_0)$
- Weights θ_0, ψ_0
- Area of integration $K \subset \mathbb{R}$
- Number of training iterations $N \in \mathbb{N}$
- Initial state X_0

$\widehat{u}_{X_0} \leftarrow \text{ILQG}(X_0, \widehat{\mathfrak{p}}_0)$
$V^{\theta_0} \leftarrow \widehat{g}_0 + \mathfrak{X}^{\theta_0}$
$u^{\psi_0} \leftarrow \widehat{u}_{X_0} + \mathfrak{X}^{\psi_0}$
for $n = 1, ..., N$ **do**
  $\mathcal{D} \leftarrow$ *Batch of uniform samples of* $[0, T) \times K$
  $\ell \leftarrow \frac{1}{|\mathcal{D}|} \sum_{(t,x)\in\mathcal{D}} \mathcal{L}(t, x, V^{\theta_0}, u^{\psi_0}, \widehat{\mathfrak{p}}_0)$
  $\theta_n, \psi_n \leftarrow \text{ADAM}(\theta_{n-1}, \psi_{n-1}, \textit{loss} = \ell)$
end for

Algorithm 2 : `DARE` - Online Phase

Input:
- OC pair $\widehat{\mathfrak{p}}_0 = (\widehat{h}_0, \widehat{f}_0)$
- Initial weights θ, ψ
- Area of integration $K \subset \mathbb{R}$
- Number of ADAM updates in offline step $N_{off} \in \mathbb{N}$
- Number of ADAM updates per online step $N_{on} \in \mathbb{N}$
- Time discretization $\mathcal{T} \subset (0, T]$
- Initial State X_0

$V^{\theta_0}, u^{\psi_0} \leftarrow \texttt{DARE}(\widehat{\mathfrak{p}}_0, \theta, \psi, K, N_{off}, X_0)$
$t_{prev} \leftarrow 0$
$\widehat{\mathfrak{p}}_{prev} \leftarrow \widehat{\mathfrak{p}}_0$
for $t \in \mathcal{T}$ **do**
  $X_t \leftarrow \text{System}(t, X_{t_{prev}}, u^{\psi_{t_{prev}}}(t, X_{t_{prev}}))$
  $\widehat{\mathfrak{p}}_t \leftarrow \text{Approx}(\widehat{\mathfrak{p}}_{prev}, \widehat{\mathfrak{p}}_{prev}(X_t, u^{\boldsymbol{\psi}}_{t_{prev}}(t, X_t)) + \epsilon)$
  $V^{\theta_t}, u^{\psi_t} \leftarrow \texttt{DARE}(\widehat{\mathfrak{p}}_t, \theta_{t_{prev}}, \psi_{t_{prev}}, t, N_{on})$
  $t_{prev} \leftarrow t$
end for

C Experimental Parameters

Default parameters for the LQG and MPC problems are reported here.

Table 2: Default parameter values for the LQG problem.

PARAM.	b	c	σ	ϕ	α	x_0	T
VALUE	-5	1	1	1	0.3	10	1

Table 3: Default parameters values for the MPC problem.

Param.	b	c	σ	ϕ	φ	α	x_0	T
Value	0	1	1	0.15	0.1	0.05	100	1

D ILQG

We provide a brief overview of the ILQG method from [51] that we use to initialize the control policy. Let the system X_t evolve as:

$$\mathrm{d}X_t = h(X_t, u_t)\mathrm{d}t + \Sigma(X_t, u_t)\mathrm{d}W_t$$

and let the performance criterion be

$$J(t, x; u) = \mathbb{E}\left[g(X_T) + \int_t^T f(\tau, X_\tau, u_\tau)\mathrm{d}\tau\right] .$$

In this section, we assume that the agent seeks to *minimize* $J(t, x; u)$. Let $\overline{u}_t$ be a random *open-loop* control policy, and consider

$$\mathrm{d}\overline{X}_t = f(\overline{X}_t, \overline{u}_t) .$$

Next, we linearize the original system around $\overline{X}_t, \overline{u}_t$ and discretize time $k = \{0, \dots, K\}$ with $\Delta t = \frac{T}{K-1}$ and $t_k = k\delta t$.

Define the discrepancies $\delta X_t = X_t - \overline{X}_t, \delta u_t = u_t - \overline{u}_t$, which evolve (approximately) as

$$\begin{aligned}
\delta X_{k+1} &= A_k \delta X_k + B_k \delta u_k + \mathcal{C}_k(\delta u_k)\xi_k \\
\mathcal{C}_k &= c_{1,k} + C_{1,k}\delta u_k + \cdots + C_{d,d_u} \\
\mathrm{cost}_k &= q_k + \delta X_k^\intercal \boldsymbol{q}_k + \frac{1}{2}\delta X_k^\intercal Q_k \delta X_k \\
&\quad + \delta u_k^\intercal \boldsymbol{r}_k + \frac{1}{2}\delta u_k^\intercal R_k \delta u_k + \delta u_k^\intercal P_k \delta X_k ,
\end{aligned}$$

where $\delta X_0 = 0$, $\xi_k \sim N(0, I_{d_X})$,

$$\begin{aligned}
A_k &= I_{d_X} + \Delta t\, h_x & \boldsymbol{q}_k &= \Delta t\, f_x \\
B_k &= \Delta t\, h_u & Q_k &= \Delta_t\, f_{xx} \\
c_{i,k} &= \sqrt{\Delta t}\, \Sigma^i & \boldsymbol{r}_k &= \Delta t\, f_u \\
C_{i,k} &= \sqrt{\Delta t}\, \Sigma_u^i & R_k &= \Delta t\, f_{uu} \\
q_k &= \Delta t\, f & P_k &= \Delta t\, f_{ux} ,
\end{aligned}$$

and $q_K = g$, $\boldsymbol{q}_K = g_x$, and $Q_K = g_{xx}$.

Above, all functions are evaluated at $\overline{X}_k, \overline{u}_k$, and Σ^i denotes the i-th row of Σ. It is shown in [51] that the optimal control to the linearized system δu^* is affine, with

$$\delta u^*(\delta X) = l_k + L_k\, \delta X . \tag{16}$$

When δu takes the form (16), the value function is quadratic and we write

$$V_k(\delta X) = s_k + \delta X_k^\intercal \boldsymbol{s}_k + \frac{1}{2}\delta X_k^\intercal S_k \delta X_k \, .$$

On can obtain an explicit representation of $S_k, \boldsymbol{s}_k, s_k$ by first defining

$$\begin{aligned} \boldsymbol{g}_k &= \boldsymbol{r}_k + B_k^\intercal \boldsymbol{s}_k + \sum_i C_{i,k}^\intercal S_{k+1} c_{i,k} \\ G_k &= P_k + B_k^\intercal S_{k+1} A_k \\ H_k &= R_k + B_K^\intercal B_k + \sum_i C_{i,k}^\intercal S_{k+1} C_{i,k} \, , \end{aligned}$$

which leads to the following equalities

$$\begin{aligned} S_k &= Q_k + A_k^\intercal S_{k+1} A_k - L_k^\intercal H_k L_k + L_k^\intercal G_k + G_k^\intercal L_k \\ \boldsymbol{s}_k &= \boldsymbol{q}_k + A_k^\intercal \boldsymbol{s}_{k+1} + L_k^\intercal H_k l_k + L_k^\intercal \boldsymbol{g}_k + G_k^\intercal l_k \\ s_k &= q_k + s_{k+1} + \frac{1}{2}\sum_i c_{i,k} S_{k+1} c_{i,k} + \frac{1}{2} l_k^\intercal H_k l_k + l_k^\intercal \boldsymbol{g}_k \, , \end{aligned}$$

where $S_K = Q_K, \boldsymbol{s}_K = \boldsymbol{q}_K, s_K = q_K$. Consequently, we obtain

$$\begin{aligned} l_k &= -H_k^{-1} \boldsymbol{g}_k \\ L_k &= -H_k^{-1} G_k \, . \end{aligned}$$

When f or g are not convex, H may have negative eigenvalues. This generally causes numerical issues due to the the minimization problem being unbounded. In this case, we use the Levenberg-Marquardt method to achieve an approximate inverse, by forcing all negative eigenvalues of H to be equal to some $\lambda > 0$.

E Transfer Learning in Neural Adaptive Control

One way of interpreting the ability of DARE to adapt to unseen environments in real time is through transfer Learning (TL). TL encompasses methods in which knowledge acquired from an initial source task is used to improve performance on a related target task; see [40, 56, 37, 47, 50] for an overview of transfer learning. One can study the efficiency of DARE in the online phase from the perspective of TL because its performance hinges on successive transferring of knowledge (parameters) between DNNs corresponding to the solutions to "similar" OC problems; see Figure 1. In this section, we provide a theoretical justification for our method. More precisely, we analyze the smoothness of OC problems with respect to the OC pair describing the environment and the resulting smoothness of DNN parameters.

To provide a theoretical foundation to this claim, we use the tools of regular perturbation in OC and a notion of continuity of the DARE network parameters. Later, Section 4.2 explores specific examples and quantifies empirically the improvement achieved from TL in the online phase of DARE. In particular, we use the number of iterations required to attain, on average, a prespecified loss in the target task to measure the *strength of transfer*.

To streamline our analysis, assume $d_X = d_u = 1$ and consider an agent who receives observations of the OC pair and updates their estimate $\widehat{\mathfrak{p}}_t$, accordingly.[2] In practice, between two sufficiently close observation times $r, s \in [0, T]$ with $r < s$, we assume that the estimate $\widehat{\mathfrak{p}}_r$ at time r remains close to the estimate $\widehat{\mathfrak{p}}_s$ at time s. Hence, we write $\widehat{\mathfrak{p}}_s$ as a perturbation of $\widehat{\mathfrak{p}}_r$. This is formalized in the following assumption.

Assumption E.1. For any ϵ, there are suitable perturbation functions p^f and p^h such that

$$\widehat{f}_s = \widehat{f}_r + \epsilon \, p^f \quad \text{and} \quad \widehat{h}_r = \widehat{h}_r + \epsilon \, p^h \, . \tag{17}$$

[2]It is straightforward to generalize to multi-dimensional setups.

Fix $t \in [0,T]$ and consider the value and control functions associated to $\widehat{\mathfrak{p}}_r$ and $\widehat{\mathfrak{p}}_s$ on $[t,T]$. That is, for $\rho \in \{r,s\}$, let

$$V^\rho(t,x) = \sup_u \mathbb{E}\left[\widehat{g}(X_T^\rho) + \int_t^T \widehat{f}_\rho(X_\tau^\rho, u_\tau)\,\mathrm{d}\tau \,\middle|\, X_t^\rho = x\right], \tag{18}$$

where

$$\mathrm{d}X_\tau^\rho = \widehat{h}_\rho(X_\tau^\rho, u_\tau)\,\mathrm{d}\tau + \tilde{\Sigma}\,\mathrm{d}W_\tau\,.$$

Theorem E.2 first shows that small perturbations in the OC pair lead to small perturbations in the optimal policy and the value function. Next, the result examines the continuity of the parameters of the DNNs approximating the value and control functions. This continuity is considered with respect to the function space in which the functions are defined. Intuitively, when a DNN is trained to a particular function, one expects that marginal changes to this function will result in marginal changes to the network parameters. Providing such a result in a general setting poses an intricate challenge. Thus, we simplify the setting by reducing the class of DNNs to that of single-layer perceptrons. However, our empirical findings suggest that it generalizes to more general cases.

Theorem E.2. *Suppose that $\widehat{f}^r$, $\widehat{f}^s$, $\widehat{h}^r$, $\widehat{h}^s$, p^f, $p^h \in C_b^{1,2}([0,T];K)$ and ϵ is defined as in* (17).[3] *Moreover, assume that solutions $(V^r, u^{r,\star})$ and $(V^s, u^{s,\star})$ to* (18) *exist and are unique. For a value of ϵ which is sufficiently small, then there exists $L > 0$ such that for any $\gamma > 0$ and single-layer perceptron approximations $(V^{\boldsymbol{\theta}_r}, u^{\boldsymbol{\psi}_r})$ and $(V^{\boldsymbol{\theta}_s}, u^{\boldsymbol{\psi}_s})$ of (V^r, u^r) and (V^s, u^s), respectively, with precision γ, such that*

$$\|\boldsymbol{\theta}^r - \boldsymbol{\theta}^s\| + \|\boldsymbol{\psi}^r - \boldsymbol{\psi}^s\| \le L\epsilon^2\,. \tag{19}$$

The proof of Theorem E.2 is given below. Although the above result justifies the performance of `DARE` for two consecutive policy updates with two fixed OC pairs, the results extend to the case of a dynamic estimate of the environment. That is, suppose $(h,f) = (h_t, f_t)$ evolves throughout $t \in [0,T]$. Then, if (h_t, f_t) changes smoothly, we expect the DNNs parameterizing the corresponding solutions to vary smoothly. Finally, our numerical results indicate that `DARE` also adapts efficiently to large and abrupt changes in the environment.

E.1 Definitions

First, we introduce the notation used throughout the section.

Definition E.3 (Single-Layer Perceptron (SLP))**.** Denote $d_\mathrm{i}, d_\mathrm{h}, d_\mathrm{o} \in \mathbb{N}$, $\sigma : \mathbb{R} \to \mathbb{R}$. A *single-layer perceptron* is defined as

$$F: \begin{array}{ccc} \mathbb{R}^{d_\mathrm{i}} & \longrightarrow & \mathbb{R}^{d_\mathrm{o}} \\ x & \longmapsto & \sum_{i=1}^{d_\mathrm{h}} (C^\intercal)_i \phi \bullet (A_i x + b_i) \end{array}$$

with $A_i \in \mathbb{R}^{d_\mathrm{i}}$, $b_i \in \mathbb{R}$, $(C^\intercal)_i \in \mathbb{R}^{d_\mathrm{o}}$ for $i \in \{1,\ldots,d_\mathrm{h}\}$ and $\bullet$ denotes the component-wise application. We denote with $\theta := (A,b,C) \in \mathbb{R}^d$ the parameters of this SLP, with $d = d_\mathrm{i} d_\mathrm{h} + d_\mathrm{h} + d_\mathrm{o} d_\mathrm{h}$, and F_θ is an SLP with parameter θ.

E.2 Proof of Theorem E.2

We split the proof of Theorem E.2 into two results. Proposition E.4 shows that perturbations in the environment lead to perturbations of similar scale in the value function and the optimal policy of OC problems. Next, Proposition E.5 shows that for small perturbations of the value function and optimal policy, the parameters of the networks used to approximate these functions are continuous.

Proposition E.4. *There is a constant C such that*

$$|V^r(t,x) - V^s(t,x)| \le C\,\epsilon^2\,, \tag{20}$$

$$|u^{r,\star} - u^{s,\star}| \le C\,\epsilon\,. \tag{21}$$

[3] $C_b^{1,2}([0,T];K)$ denotes the set of functions defined on $[0,T] \times K$ with continuous first derivative in t and continuous and bounded second derivatives in x.

Proof First, note that the assumption of Theorem E.2 ensure that the functional J is well defined. Observe that the OC problem V^r is a perturbation of V^s but also V^s is a perturbation of V^r. In particular, first write

$$J^r(t,x,u) = \mathbb{E}\left[\widehat{g}(X_T^s) + \int_t^T \widehat{f}_s(X_\tau^s, u_\tau)\,\mathrm{d}\tau \,\middle|\, X_t^s = x\right] \tag{22}$$

$$J^r(t,x,u) = \mathbb{E}\left[\widehat{g}(X_T^r) + \int_t^T \widehat{f}_s(X_\tau^r, u_\tau)\,\mathrm{d}\tau \,\middle|\, X_t^r = x\right]. \tag{23}$$

Next, use Theorem 2.1 of Chapter III and Theorem 2.1 of Chapter IV in [7] to write

$$\begin{cases} |V^r(t,x) - J^r(t,x,u^{s,\star})| & \le C_0\,\epsilon^2, \\ |V^s(t,x) - J^s(t,x,u^{r,\star})| & \le C_1\,\epsilon^2\,, \end{cases}$$

for suitable constants C_0 and C_1. Finally, use the asymptotic expansions (Section 2.3, Chapter III, in [7]) to write

$$\begin{cases} |V^r(t,x) - J^s(t,x,u^{s,\star})| & \le L_0\,\epsilon^2, \\ |V^s(t,x) - J^r(t,x,u^{r,\star})| & \le L_1\,\epsilon^2\,, \end{cases}$$

for suitable constants L_0 and L_1. □

Proposition E.5. *Let $K \subseteq \mathbb{R}^{d_\mathrm{i}}$ be compact; $f \in C_b^2(K;\mathbb{R})$. There exists $\delta, L > 0$ such that for every $f' : K \to \mathbb{R}^{d_\mathrm{o}}$ with $\|f - f'\| < \delta$ and every $\gamma > 0$, there exists parameters $\theta, \theta' \in \mathbb{R}^d$ such that*

$$\|F_\theta - f\|_{C_b^2(K;\mathbb{R})} \le \gamma, \tag{24}$$

$$\|F_{\theta'} - f'\|_{C_b^2(K;\mathbb{R})} \le \gamma, \tag{25}$$

and

$$\|\theta' - \theta\| < L\|f - f'\|_{C_b^2(K;\mathbb{R})}\,, \tag{26}$$

where F_θ is a single-layer perceptron with ReLU activation of width d_h.

Proof Without loss of generality, assume that K includes an open set around 0, i.e. there exists $0 < \delta < \epsilon$ s.t. $\mathcal{D}^\delta - f \subseteq C_b^2(K;\mathbb{R})$, where

$$\mathcal{D}^\delta := \left\{ f' \in C_b^2(K;\mathbb{R}) \middle| \|f - f'\|_{C_b^2(K;\mathbb{R})} < \delta \right\}.$$

Fix an arbitrary $\gamma > 0$. According to [32], Theorem 2.1, we can find a hidden dimension $d_\mathrm{h} \in \mathbb{N}$, a matrix $A \in \mathbb{R}^{d_\mathrm{h} \times d_\mathrm{i}}$, a vector $b \in \mathbb{R}^{d_\mathrm{h}}$, and a continuous linear functional $\mathcal{C} : C_b^2(K;\mathbb{R}) \to \mathbb{R}^{d_\mathrm{h}}$ such that

$$\|f' - F_{(A,b,\mathcal{C}(f'))}\|_p \le \gamma, \quad f \in \mathcal{D}^\delta.$$

Since for $f' \in \mathcal{D}^\delta$ holds

$$F_{(A,b,\mathcal{C}(f'))} = F_{(A,b,\mathcal{C}(f'-f))} + F_{(A,b,\mathcal{C}(f))},$$

due to linearity of $\mathcal{C}$ and Definition E.3, we have

$$\|\theta' - \theta\| = \|\mathcal{C}(f' - f)\| \le L\|f - f'\|_{C_b^2(K;\mathbb{R})}$$

where we used the fact that the operator norm $\|\mathcal{C}\|_T$ of a continuous operator is finite. We conclude $\|\theta' - \theta\| < \epsilon$ and finish the proof. □

F Performance

We report training performance of all the methods tested in Section 4.1 in Table 4. All tests have been conducted on a standard MacBook Pro M1 and a single NVIDIA A40 for training each of the RL algorithms. We make our code public in the following (anonymized) repo: `https://anonymous.4open.science/r/dare-7136/README.md`

Table 4: Run time of different algorithms in the experiments.

Section	Test	Algorithm	Run time in seconds per 1000 iteration
4.2 Training performance	Offline LQG	DGM	4.26
		MLP	1.95
		DARE	2.24
	Offline MPC	DGM	6.02
		MLP	2.28
		DARE	2.34
4.4 LQG	Online phase	DARE	7.47
4.5 MPC	Exploration-Exploitation	DARE	2.46
	Non-stationary	DARE	2.52
4.1 High-dimensional	Offline phase	DGM	6.01
		MLP	2.41
		DARE	4.01
	Online phase	DGM	6.64
		MLP	3.26
		DARE	3.92

G Filtering mathematics

G.1 Perfect knowledge of the drift

This section solves the OC problem (14) when the agent fixed the value of the drift and does not update their belief throughout the time window.

When the drift is known and fixed, the OC problem (14) can be solved with standard methods [55], and is

$$u^{\star} = \frac{c}{2\,\phi}\,\left(2\,A\,(t)\;x + B\,(t) + 1\right)\,, \tag{27}$$

where A and B solve the ODE system

$$\begin{cases} -A^{'}\,(t) = \frac{c\,A(t)^2}{2\,\phi} \\ -B^{'}\,(t) = 2\,\mu\,A\,(t) + \frac{c^2\,A(t)\,(B(t)+1)}{\phi}\,. \end{cases} \tag{28}$$

G.2 Bayesian filtering of the Gaussian drift

This section solves the OC problem when the agent uses a Gaussian prior to continuously update their estimation of the drift throughout the time window of the OC problem.

Consider the control problem in (14). When the agent uses a Gaussian prior $\mathcal{N}\,(b_t, \Pi_0)$ for μ then it can be shown that the dynamics of x can be written

$$dx_t = \;\beta_t\,\mathrm{d}t + c\,u_t\,\mathrm{d}t + \sigma\,d\widehat{W}_t$$

in a different filtration in which $\widehat{W}$ is a Gaussian process. $\beta_t = \mathbb{E}\,[\mu|\mathcal{F}_t]$ is the best estimate of μ at time t and can be obtained analytically as

$$\beta_t = -\frac{\mathit{\Pi}\,(t)}{\sigma}\left(x_0 - \frac{\sigma\,b_0}{\Pi_0} - x_t + q_t\right)$$

where $\mathit{\Pi}(t) = \left(\mathit{\Pi}_0^{-1} + \frac{t}{\sigma}\right)^{-1}$ and $q_t = \int_0^t c\,u_t\,\mathrm{d}t\,.$

Using the learning dynamics above to solve the control problem (see [15] for details) gives the optimal control $\tilde{u}^{\star}$ given by

$$\tilde{u}^{\star} = \frac{c}{2\,\phi}\,\left(2\,A\,(t) + B\,(t)\right)\,x + \left(2\,C\,(t) + B\,(t)\right)\,q \tag{29}$$

$$+ \left(1 + D\left(t\right) + \ E\left(t\right)\right) \ , \tag{30}$$

where A, B, C solve the Riccati equation in

$$P\left(t\right) = \begin{pmatrix} A\left(t\right) & \frac{1}{2}B\left(t\right) \\ \frac{1}{2}B\left(t\right)^{\intercal} & C\left(t\right) \end{pmatrix}$$

$$0 = P'\left(t\right) + Y\left(t\right)^{\intercal} \ P\left(t\right) + P\left(t\right) \ Y\left(t\right) + P\left(t\right) \ U \, P\left(t\right) \ ,$$

$$Y\left(t\right) = \begin{pmatrix} \frac{\Pi(t)}{\sigma} & \frac{\Pi(t)}{\sigma} \\ 0 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} \frac{c^2}{\phi} & \frac{c^2}{\phi} \\ \frac{c^2}{\phi} & \frac{c^2}{\phi} \end{pmatrix},$$

$$P\left(T\right) = \begin{pmatrix} -\alpha & 0 \\ 0 & 0 \end{pmatrix},$$

and D and E solve the ODE system

$$\begin{cases} 0 = & D^{'}\left(t\right) + 2\,\overline{\Pi}\,\Pi\left(t\right) \ A\left(t\right) - \Pi\left(t\right)\left(1 + D\left(t\right)\right) \\ & + \frac{c^2}{4\,\phi} \ \left(2\,A\left(t\right) + B\left(t\right)\right)^2 \\ 0 = & E^{'}\left(t\right) + \overline{\Pi}\,\Pi^{\intercal}\,B\left(t\right) + \Pi^{\intercal} \ \left(1 + D\left(t\right)\right) \\ & + \frac{c^2}{4\,\phi} \ \left(2\,C\left(t\right) + B\left(t\right)\right)^2 \ , \end{cases}$$

with terminal conditions $D(T) = E(T) = 0$.

H Algorithmic trading in high dimension

We motivate the multidimensional setup in our experiments of Section 4.1. Consider the case of the trading desk of a large bank that must execute a number $d \in \mathbb{N}^*$ of large transactions in d correlated financial assets throughout a trading window $[0, T]$. The trading desk must minimize their trading costs while minimizing the risk of their positions. Throughout this section, we consider a filtered probability space $\left(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}\right)$, with $T > 0$, satisfying the usual conditions and supporting all the processes we introduce.

Let $Q_0 \in \mathbb{R}^d$ represent the transaction sizes in every asset. The inventory of the agent is modeled by $(Q_t)_{t\in[0,T]} = \left(Q_t^1, \ldots, Q_t^d\right)^{\intercal}_{t\in[0,T]}$ and it evolves with the trading speed $(u_t)_{t\in[0,T]} = (u_t^1, \ldots, u_t^d)^{\intercal}_{t\in[0,T]}$ in each asset:[4]

$$dQ_t = u_t \, \mathrm{d}t.$$

The prices $(S_t)_{t\in[0,T]} = \left(S_t^1, \ldots, S_t^d\right)^{\intercal}_{t\in[0,T]}$ of the d assets are modeled as correlated Brownians with dynamics

$$dS_t = \tilde{\Sigma}\,\mathrm{d}W_t\,,$$

where $W = \left(W^1, \ldots, W^d\right)$ is a d-dimensional standard Brownian motion and $S_0 \in \mathbb{R}^d$ is known. The matrix $\tilde{\Sigma} \in \mathcal{M}_d(\mathbb{R})$ measures the correlation of the prices and we define the covariance matrix $\Sigma = \tilde{\Sigma}\,\tilde{\Sigma}^{\intercal} \in S_d^{++}(\mathbb{R})$.[5]

Trading activity of the agent generates transaction costs, driven by some function of the trading speed $f(u_t)$ so the cash from their trading activity evolves as

$$\mathrm{d}X_t = -u_t^{\intercal} S_t \,\mathrm{d}t - f(u_t)\,\mathrm{d}t, \quad X_0 = 0.$$

The agent maximizes the exponential utility of their terminal wealth so their objective is

$$V(t, x, q, s) = \sup_v \mathbb{E}\Big[- \exp\Big(- \gamma\big(Q_T^{\intercal}\,S_T - Q_T^{\intercal}\,\Gamma\,Q_T \tag{31}$$

$$- \int_t^T u_s^{\intercal}\,S_s\,\mathrm{d}s - \int_t^T f(u_s)\,\mathrm{d}s\big)\Big)\Big], \tag{32}$$

[4] The superscript $^{\intercal}$ is the transpose operator.

[5] $\mathcal{M}_d(\mathbb{R}) := \mathcal{M}_{d,d}(\mathbb{R})$ is the set of $d \times d$ real square matrices, $\mathcal{S}_d(\mathbb{R})$ is the set of real symmetric $d \times d$ matrices, and $\mathcal{S}_d^{++}(\mathbb{R})$ is the set of positive matrices.

for values $Q_t = q$, $X_t = x$, and $S_t = s$ at time t.

The dynamic programming principle holds and the HJB equation associated with the problem

$$
\begin{aligned}
0 = \partial_t V &+ \frac{1}{2}\mathrm{Tr}\left(\Sigma\, D^2_{SS} V\right) \\
&+ \sup_{u\in\mathbb{R}^d} \left(-(u^\intercal s + f(u))\partial_x V + v^\intercal \nabla_q V\right), \qquad (33)
\end{aligned}
$$

with terminal condition

$$
V(T,x,q,s) = -\exp\left(-\gamma\left(q^\intercal s - q^\intercal \Gamma q\right)\right). \qquad (34)
$$

In the experiment of Section 4.1, we solve the HJB (33)-(34) using `DARE` to obtain the optimal policy of the trading agent.

When all the parameters of the problem are known and fixed, i.e., the agent does not adapt to new information, the problem described above admits an analytical solution which we use to study the performance of `DARE`.

To solve the problem semi-analytically, the function f must be a quadratic form, that is, there is some $\eta \in S_d^{++}(\mathbb{R}^{d\times d})$ with

$$
f(u) = u^\intercal\, \eta\, u\,. \qquad (35)
$$

We follow the standard steps in linear-exponential quadratic Gaussian (LEQG) control and we propose the following form for the value function

$$
\begin{aligned}
&V(t,x,q,s) = \\
&\quad -\exp\left(-\gamma\left(x + q^\intercal S + Q^\intercal A(t) q + B(t)^\intercal q + C(t)\right)\right),
\end{aligned}
$$

and straightforward calculations find that the problem reduces to solving the following ODE system

$$
\begin{cases}
A'(t) = \frac{\gamma}{2}\Sigma - A(t)\eta^{-1}A(t) \\
B'(t) = -A(t)\eta^{-1}B(t) \\
C'(t) = -\frac{1}{4}B(t)^\intercal \eta^{-1} B(t),
\end{cases} \qquad (36)
$$

with terminal conditions

$$
A(T) = -\Gamma, \quad B(T) = C(T) = 0. \qquad (37)
$$

Clearly, the solutions for B and C are $B = C = 0$. To obtain a solution, we use the change of variables

$$
a(t) = \eta^{-\frac{1}{2}} A(t) \eta^{-\frac{1}{2}} \quad \forall t \in [0,T]\,,
$$

so the problem reduces to the following terminal value problem

$$
\begin{cases}
a'(t) = & \hat{A}^2 - a(t)^2 \\
a(T) = & -C,
\end{cases} \qquad (38)
$$

where

$$
\hat{A} = \sqrt{\frac{\gamma}{2}}\left(\eta^{-\frac{1}{2}}\,\Sigma\,\eta^{-\frac{1}{2}}\right)^{\frac{1}{2}} \in \mathcal{S}_d^{++}(\mathbb{R})\,,
$$

and

$$
C = \eta^{-\frac{1}{2}}\Gamma\eta^{-\frac{1}{2}} \in \mathcal{S}_d^{+}(\mathbb{R})\,.
$$

We solve (38) in the next result.

Proposition H.1. *Define* $\xi : [0,T] \to \mathcal{S}_d(\mathbb{R})$

$$
\begin{aligned}
\xi\,(t) = &-\frac{\hat{A}^{-1}}{2}\left(I - e^{-2\hat{A}(T-t)}\right) \\
&- e^{-\hat{A}(T-t)}\left(C + \hat{A}\right)^{-1} e^{-\hat{A}(T-t)} \qquad (39)
\end{aligned}
$$

as the unique solution to the ODE system

$$\begin{cases} \xi'(t) = \hat{A}\xi(t) + \xi(t)\hat{A} + I_d \\ \xi(T) = -\left(C + \hat{A}\right)^{-1}. \end{cases} \tag{40}$$

Then $\forall t \in [0, T]$*,* $\xi(t)$ *is invertible and*

$$a : t \in [0, T] \to \hat{A} + \xi(t)^{-1} \in \mathcal{S}_d(\mathbb{R})$$

is the unique solution of (38).

Thus, the value function, which we use as the oracle in Section 4.1 is given by

$$V(t, x, q, s) = \\ - \exp\left(-\gamma\left(x + q^\intercal S + Q^\intercal A(t) q + B(t)^\intercal q + C(t)\right)\right),$$

where

$$A(t) = \eta^{\frac{1}{2}} \left(\hat{A} - \left\{ \frac{\hat{A}^{-1}}{2} \left(I - e^{-2\hat{A}(T-t)} \right) \right. \right. \\ \left. \left. + e^{-\hat{A}(T-t)} \left(C + \hat{A} \right)^{-1} e^{-\hat{A}(T-t)} \right\}^{-1} \right) \eta^{\frac{1}{2}}.$$

Finally, Figure H shows the true value function and the solution learned by `DARE` for a set of model parameters in dimension 5, and Figure 9 shows the associated training loss.

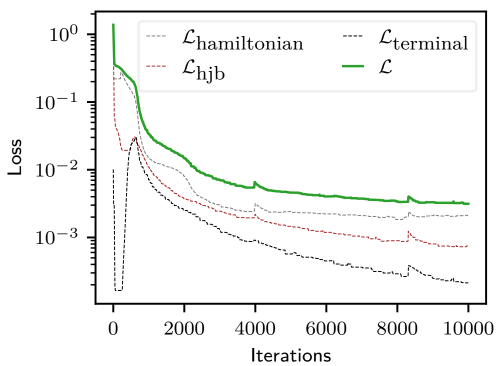

Figure 9: Training loss (10) of `DARE` for the multidimensional OC problem (31). The parameter values are $\Gamma = 10^{-2} \times I_5$,

$$\eta = 10^{-3} \times \begin{pmatrix} 25.13 & 10.41 & 11.67 & 13.75 & 22.21 \\ 10.41 & 6.42 & 7.68 & 9.12 & 12.4 \\ 11.67 & 7.68 & 12.95 & 11.2 & 18.71 \\ 13.75 & 9.12 & 11.2 & 17.04 & 17.25 \\ 22.21 & 12.4 & 18.71 & 17.25 & 29.02 \end{pmatrix}, \quad \text{and} \quad \Sigma = \begin{pmatrix} 2.83 & 2.02 & 2.53 & 1.91 & 1.59 \\ 2.02 & 1.96 & 2.04 & 1.28 & 1.31 \\ 2.53 & 2.04 & 2.85 & 2.04 & 1.32 \\ 1.91 & 1.28 & 2.04 & 1.76 & 0.96 \\ 1.59 & 1.31 & 1.32 & 0.96 & 1.06 \end{pmatrix}.$$

I Gaussian Process mathematics

Formally, a GP is a random function $f : \mathcal{X} \mapsto \mathbb{R}$, such that, for any finite set of points $\mathbf{X}_\star \subseteq \mathcal{X}$, the random vector $\boldsymbol{f}_\star = \{f(\boldsymbol{x})\}_{\boldsymbol{x} \in \mathbf{X}_\star}$ follows a multivariate Gaussian distribution. The shape of the function f is determined by a finite set of (training) observations $\boldsymbol{y} = \{y_i\}_{i \in \{1,\ldots,n\}}$ collected at the (training) observation points $\mathbf{X} = \{\mathbf{x}_i\}_{i \in \{1,\ldots,n\}}$, where $y_i = f(\mathbf{x}_i) + \epsilon_i$ is subject to i.i.d.

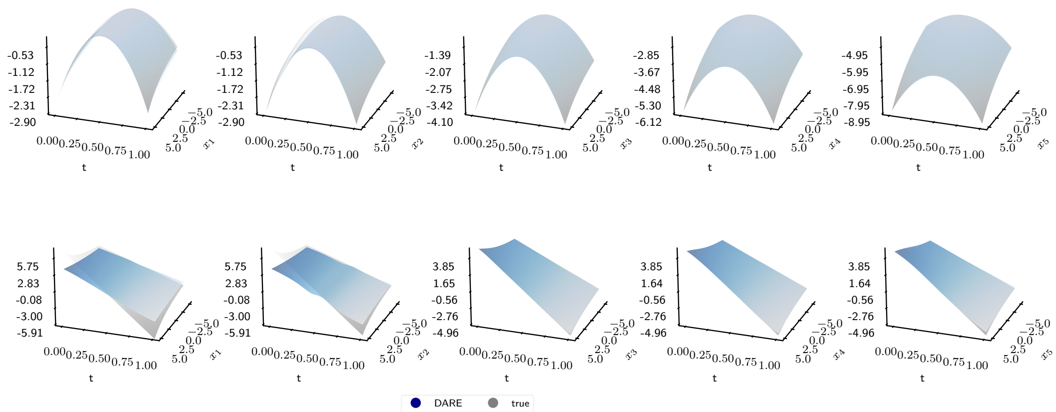

Figure 10: True and approximated (with DARE) value function (31) for $t \in [0, T]$ and $\boldsymbol{X} = X_1, X_2, X_3, X_4, X_5 \in [-5, 5]^5$. Each surface corresponds to the value function for time and one dimension in X, where the value of the system in all other dimensions is fixed to $x_i = 0$.

Gaussian measurement noise $\epsilon_i \sim \mathcal{N}(0, s^2)$ for $s > 0$. GPs are fully specified by a mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and a covariance (kernel) function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, In particular, if $f \sim \mathcal{GP}(\mu, k)$ and $\mathbf{X}_\star$ is a set of test points in the domain $\mathcal{X}$ of the GP, then the set of random variables $\boldsymbol{f}_\star$ is Gaussian with parameters $\mathcal{N}(\boldsymbol{\mu}_\star, \boldsymbol{K}_{\star,\star})$, where

$$\boldsymbol{\mu}_\star = \{\mu(\boldsymbol{x})\}_{\boldsymbol{x} \in \mathbf{X}_\star} \quad \text{and} \quad \boldsymbol{K}_{\star,\star} = \{k(\boldsymbol{x}, \boldsymbol{x'})\}_{(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbf{X}_\star}.$$

A convenient property of GPs is that one computes the posterior distribution with analytic formulae. Suppose we collect n noisy observations $\boldsymbol{y} = \{y_1, \ldots, y_n\}$ at the domain points $\mathbf{X} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$, where $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ and $\epsilon_i \sim \mathcal{N}(0, s^2)$. Then, the posterior distribution over f given the previous (training) observations $\mathbf{X}$ and $\boldsymbol{y}$, is also a GP with mean function μ_{post} and covariance function k_{post} given by

$$\begin{cases} \mu_{\text{post}}(\boldsymbol{x}_\star) & = \boldsymbol{k}(\boldsymbol{x}_\star, \mathbf{X})\,(\boldsymbol{K} + s^2\,\boldsymbol{I})^{-1}\,\boldsymbol{y}\,, \\ k_{\text{post}}(\boldsymbol{x}_\star, \boldsymbol{x}'_\star) & = k(\boldsymbol{x}_\star, \boldsymbol{x}'_\star) \\ & \quad -\boldsymbol{k}(\boldsymbol{x}_\star, \mathbf{X})\,(\boldsymbol{K} + s^2\,\boldsymbol{I})^{-1}\,\boldsymbol{k}(\mathbf{X}, \boldsymbol{x}'_\star)\,, \end{cases} \tag{41}$$

where

$$\boldsymbol{k}(\boldsymbol{x}_\star, \mathbf{X}) = \boldsymbol{k}(\mathbf{X}, \boldsymbol{x}_\star)^\intercal = (k(\boldsymbol{x}_\star, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}_\star, \boldsymbol{x}_n))$$

is the n-dimensional covariance vector of the test point $\mathbf{x}_\star$ with training points $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, $\boldsymbol{K} = (k(\boldsymbol{x_i}, \boldsymbol{x_j}))_{i,j \in \{1,\ldots,n\}}$ is the positive semi-definite kernel matrix from training data, and $\boldsymbol{I}$ is the n-dimensional identity matrix. See Figure I for an example.

Let the elements of the vector $\boldsymbol{\theta} \in \Theta$ be hyper-parameters of the prior's kernel function and s^2 is the variance of the i.i.d. Gaussian noise that corrupts reward observations. Both $\boldsymbol{\theta}$ and s^2 are inferred with the log marginal likelihood of the data given by

$$\begin{aligned} L(\boldsymbol{\theta}, s) &= \log p(\,\boldsymbol{y} \mid \mathbf{X}, \boldsymbol{\theta}, s) \\ &= -\frac{1}{2}\log\Big(\det\left(\boldsymbol{K_\theta} + s^2\,\boldsymbol{I}\,\right)\Big) \\ &\quad -\frac{1}{2}\,\boldsymbol{y}^\intercal\left(\boldsymbol{K_\theta} + s^2\,\boldsymbol{I}\,\right)^{-1}\boldsymbol{y} - \frac{n}{2}\log(2\,\pi)\,, \end{aligned} \tag{42}$$

for a zero-mean GP, where $\mathbf{X}$ and $\boldsymbol{y}$ are the n training samples and $\boldsymbol{K_\theta}$ is the prior's positive covariance matrix with kernel $k_{\boldsymbol{\theta}}$. The vector of hyper-parameters $\boldsymbol{\theta}$ and the variance s^2 maximize the quantity (42), i.e., $(\boldsymbol{\theta}^\star, s^\star) \in \underset{\boldsymbol{\theta} \in \Theta, s \in \mathbb{R}^+}{\arg\max}\ L(\boldsymbol{\theta}, s)$, which one solves with classical gradient descent-based optimization algorithms.

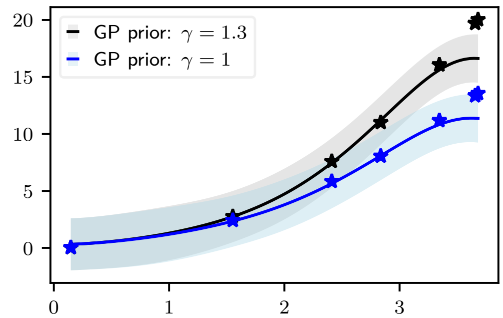

Figure 11: Two GPs fitted to $f(u) = u^{1+\gamma_i}$ for $\gamma_0 = 1.3$ and $\gamma_1 = 1$.

J Reinforcement Learning Benchmarks

Table 1 shows the average elapsed real time i.e. wall-clock time comparison of Reinforcement Learning Algorithms (PPO, SAC, A2C) with that of `DARE` rounded to the nearest second training on the LQG problem from section 4.3. We utilize the implementations in the Stable-Baselines-3 library [41] for the Reinforcement Learning experiments.

We measure the elapsed time from the first step in the training loop until convergence using the `time` module from Python 3.9.19.

The hyperparameter ranges in Table 5 are determined in a way that includes the recommended values on the Stable-Baselines-3 [41] implementation used in this work.

Table 5: Hyperparameter Range considered for each RL algorithm

Hyperparameter	Algorithm	Range
Learning Rate	PPO	$[10^{-4}, 10^{-3}]$ (continuous range)
	SAC	$[10^{-4}, 10^{-3}]$ (continuous range)
	A2C	$[10^{-4}, 10^{-3}]$ (continuous range)
Action Repeats	PPO	$\{10, 20, 40, 50, 100\}$
	SAC	$\{10, 20, 40, 50, 100\}$
	A2C	$\{10, 20, 40, 50, 100\}$
Number of Steps	PPO	$[128, 4096]$ (integer range)
	A2C	$[4, 32]$ (integer range)

K Impact Statement

This paper presents a novel deep learning methodology for solving decision-making problems in noisy and non-stationary environments, with wide-ranging applications in finance, robotics, and biology. Our contribution is a highly accurate and efficient method for solving model predictive control problems. Possible implications include more efficient and effective risk management in finance, safer robot-human interaction, and improved biomedical engineering. We use tractable examples to test our approach and to demonstrate that our model produces reasonable policies. Before implementing our model for critical problems, we believe further specific experimentation and validation is necessary.

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: The claims in the introduction and abstract accurately reflect the set of problems the paper is focused on and how the proposed methods were tested and compared.

   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: Yes, we discuss the limitations of the work like its requirements for PDE representations that are not required by other fields like Reinforcement Learning that also deal with high dimensional control problems.

   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: Section E of the Appendix includes the proofs with the relevant definitions and propositions.

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: We provide all experimental details along with an anonymized repository containing all code.

   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: [Yes]

Justification: All code can be run using open source packages without external data.

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: Outlines of each of the experiments as well as our choice of hyperparameters is given in Section 4.1 and Appendix J.

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: Each test is run with a sufficient across a sufficient number of seeds when necessary.

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: We provide hardware information in Appendix F. Our tests do not require any specialized or intensive computing power.

   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 conform with the NeurIPS Code of Ethics.

   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: An impact statement is included in Appendix K.

    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: We believe our model does not have a high risk for misuse.

    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: All sources are cited and we use publicly available packages.

    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: [NA]

    Justification: The paper does not release new assets.

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: The paper does not involve crowdsourcing nor research with human subjects

    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.