DART:
Noise Injection for Robust Imitation Learning

Michael Laskey
EECS Department
UC Berkeley

Jonathan Lee
EECS Department
UC Berkeley

Roy Fox
EECS Department
UC Berkeley

Anca Dragan
EECS Department
UC Berkeley

Ken Goldberg
IEOR and EECS Department
UC Berkeley

Abstract: One approach to Imitation Learning is Behavior Cloning, in which a robot observes a supervisor and infers a control policy. A known problem with this "off-policy" approach is that the robot's errors compound when drifting away from the supervisor's demonstrations. On-policy, techniques alleviate this by iteratively collecting corrective actions for the current robot policy. However, these techniques can be tedious for human supervisors, add significant computation burden, and may visit dangerous states during training. We propose an off-policy approach that *injects noise* into the supervisor's policy while demonstrating. This forces the supervisor to demonstrate how to recover from errors. We propose a new algorithm, DART (Disturbances for Augmenting Robot Trajectories), that collects demonstrations with injected noise, and optimizes the noise level to approximate the error of the robot's trained policy during data collection. We compare DART with DAgger and Behavior Cloning in two domains: in simulation with an algorithmic supervisor on the MuJoCo tasks (Walker, Humanoid, Hopper, Half-Cheetah) and in physical experiments with human supervisors training a Toyota HSR robot to perform grasping in clutter. For high dimensional tasks like Humanoid, DART can be up to $3x$ faster in computation time and only decreases the supervisor's cumulative reward by 5% during training, whereas DAgger executes policies that have 80% less cumulative reward than the supervisor. On the grasping in clutter task, DART obtains on average a 62% performance increase over Behavior Cloning.

Keywords: Imitation Learning, Robotics

1   Introduction

Robotic manipulation tasks are challenging to learn due to noncontinuous dynamics that are difficult to model, high dimensional state representations, and potentially delayed reward. Deep reinforcement learning has the potential to train such control policies, however in practice it may require a very large number of samples [18].

Rather than pure ab initio learning, an alternative is to leverage supervision to guide the robot's policy. A common approach to this problem is Behavior Cloning, in which a robot observes a supervisor's policy and learns a mapping from state to control via regression [11, 1]. This is an *off*-policy method, that suffers from compounding error when the robot executes its policy, leading it to drift to new and possibly dangerous states [14]. Theoretically, the drifting occurs because of covariate shift, where execution of the robot's policy causes it to move to a different distribution from the one on which it was trained.

Ross and Bagnell proposed DAgger [13, 14] to correct for covariate shift by sampling states from the robot's policy. DAgger is an *on*-policy method, in which the robot iteratively rolls out its current policy and asks for supervisor labels (or corrections) at the states it visits. Empirically, DAgger has

been shown to reduce covariate shift and lead to robust control policies [5]. However, DAgger suffers from three key limitations: 1) it can be tedious for human supervisors to provide these corrections [7] 2) it can be potentially dangerous for a physical robot to visit highly sub-optimal states [24], and 3) repeatedly updating the robot's policy is computationally expensive. In this paper, we introduce an alternative approach to address covariate shift and enable robust policies.

One way to expose the robot to corrective examples and reduce drift is to inject noise into the supervisor's demonstrations and let the supervisor provide corrections as needed. Our insight is that by injecting small levels of noise, we will focus on the states that the robot needs to recover from – the states *at the boundary* of the ideal policy. This has the potential to obtain the advantages of on-policy methods by recovering from errors as the robot starts making them, without the disadvantage of aggregating these errors at training time, and getting the robot to dangerous states with low reward. We propose to still perform off-policy learning, but to inject an optimized level of noise into the supervisor's control stream as we acquire demonstrations.

Injecting noise requires selecting the magnitude and direction of the noise appropriately. Intuitively, the noise should approximate the error of the trained robot's policy, so that the demonstrations have state distributions that are close to the one that the robot will encounter at test time. This may be challenging in robotic control tasks where the control signals are high-dimensional. We thus propose to approximate the optimal noise distribution by first selecting a parameterized noise model, and then iteratively optimizing the likelihood of the supervisor's noise-injected policy applying the robot's control. We propose DART(Disturbances for Augmenting Robot Trajectories): a noise injection variant of Behavior Cloning.

We evaluate DART on the MuJoCo locomotion environments and a grasping in clutter task on a Toyota HSR robot. In the locomotion environments, where a continuous control robot is trained to walk forward, DART achieves parity with state of the art on-policy methods, such as DAgger. For high dimensional tasks like Humanoid, DART is $3x$ faster in computation time and during training only decreases the supervisor's cumulative reward by 5%, whereas DAgger executes policies that have 80% less cumulative reward than the supervisor, thus suggesting only a small amount of noise is sufficient to reduce covariate shift. We then evaluate DART with four human supervisors, who trained a robot to perform a grasping in clutter task. In the task, a robot must reason from an eye-in-hand camera perspective how to push occluding objects away to reach a goal object. We observe that on average DART leads to a 62% performance increase over traditional Behavior Cloning.

This paper contributes: 1) a new algorithm, DART, that sets the appropriate level of noise to inject into the supervisor's control stream 2) a theoretical analysis of noise injection, which illustrates when DART can improve performance over Behavior Cloning 3) experimentation with algorithmic and human supervisors characterizing noise injection can reduce covariate shift.

2   Related Work

2.1   Imitation Learning

Imitation Learning schemes are either off-policy or on-policy. In off-policy Imitation Learning, the robot passively observes the supervisor, and learns a policy mapping states to controls by approximating the supervisor's policy. This technique has been successful, for instance, in learning visuomotor control policies for self-driving cars [11, 1]. However, Pomerleau et al. observed that the self-driving car would steer towards the edge of the road during execution and not be able to recover [11]. Later, Ross and Bangell theoretically showed that this was due to the robot's distribution being different than the supervisor's, a property known as covariate shift, which caused errors to compound during execution [13].

Ross et al. [14] proposed DAgger, an on-policy method in which the supervisor iteratively provides corrective feedback on the robot's behavior. This alleviates the problem of compounding errors, since the robot is trained to identify and fix small errors after they occur. But despite their ability to address covariate shift, on-policy methods have been shown to be challenging for human supervisors [7] and require the robot to visit potentially dangerous regions of the state space during training [24]. Further, on-policy methods require retraining the policy from scratch after each round of corrections.

Techniques have been proposed to address some of these limitations. Recently, Sun et al. proposed a gradient update for on-policy methods to alleviate the computation burden [20]. However, it has been

shown local gradient updates suffer in performance compared to full retraining of the policy [23]. Zhang et al. trained a classifier to predict when the robot is likely to error during execution and then proposed to transfer control over to the supervisor [24]. However, this approach inherits the other limitations of on-policy methods, the challenge for human supervisors and the computational burden.

2.2 Noise Injection to Increase Robustness

In robust control theory, Andereson et al. noted a phenomenon similar to the covariate shift when performing model identification [19]. They observed that once the model has been estimated and a control law has been optimized, the robot visited regions of the state space in which it was unstable. A condition was proposed to correct for this known as persistance excitation, which requires the training data to be informative enough to learn a robust model [15].

One way to achieve persistance excitation is to inject isotropic Gaussian noise into the control signal [4]. Due to its full-rank covariance matrix, such white noise can expose the system to any disturbance with positive probability [10]. In light of this, we propose injecting noise into the supervisor's policy in order to collect demonstrations that simulate the errors that the robot would make at test time.

3 Problem Statement

The objective of Imitation Learning is to learn a policy that matches the supervisor's policy.

Modeling Choices and Assumptions: We model the system dynamics as Markovian and stochastic. We model the initial state as sampled from a distribution over the state space. We assume a known state space and set of actions. We also assume access to a robot, such that we can sample from the state sequences induced by a policy. Lastly, we assume access to a supervisor who can provide a demonstration of the task.

Policies and State Densities. We denote by $\mathcal{X}$ the set consisting of observable states for a robot, and by $\mathcal{U}$ the set of actions. We model dynamics as Markovian, such that the probability of visiting state $\mathbf{x}_{t+1} \in \mathcal{X}$ can be determined from the previous state $\mathbf{x}_t \in \mathcal{X}$ and action $\mathbf{u}_t \in \mathcal{U}$:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_0, \mathbf{u}_0, \dots \mathbf{x}_t, \mathbf{u}_t) = p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t).$$

We assume a probability density over initial states $p(\mathbf{x}_0)$. An environment is thus defined as a specific instance of action and state spaces, initial state distribution, and dynamics.

Given a time horizon $T \in \mathbb{N}$, a trajectory ξ is a finite sequence of T pairs of states visited and corresponding control inputs at these states, $\xi = (\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1, \dots, \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathcal{X}$ and $\mathbf{u}_t \in \mathcal{U}$ for each t.

A policy is a measurable function $\pi : \mathcal{X} \to \mathcal{U}$ from states to controls. We consider policies $\pi_\theta : \mathcal{X} \to \mathcal{U}$ parameterized by some $\theta \in \Theta$. Under our assumptions, any such policy π_θ induces a probability density over the set of trajectories of length T:

$$p(\xi|\pi_\theta) = p(\mathbf{x}_0) \prod_{t=0}^{T-1} \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_{t+1}|\mathbf{u}_t, \mathbf{x}_t)$$

The term $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ indicates stochasticity in the applied policy and we consider this to be a user-defined distribution in which the deterministic output of the policy is a parameter in the distribution. An example distribution is ϵ-greedy, in which with probability ϵ a random control is applied instead of $\pi_\theta(\mathbf{x}_t)$.

While we do not assume knowledge of the distributions corresponding to $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ or $p(\mathbf{x}_0)$, we assume that we have a real robot or a simulator. Therefore, when 'rolling out' trajectories under a policy π_θ, we utilize the robot or the simulator to sample from the resulting distribution over trajectories rather than estimating $p(\xi|\pi_\theta)$ itself.

Objective. In Imitation Learning, we do not assume access to a reward function, like we do in Reinforcement Learning [21], but instead a supervisor, π_{θ^*}, where θ^* may not be contained in Θ. We assume the supervisor achieves a desired level of performance on the task, although it may not be optimal.

We measure the difference between controls using a surrogate loss $l : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}$ [14, 13]. A commonly considered surrogate loss is the squared L2-norm on the control vectors $l(\mathbf{u}_1, \mathbf{u}_2) = ||\mathbf{u}_1 - \mathbf{u}_2||_2^2$ [8]. We measure total loss along a trajectory with respect to two policies π_{θ_1} and π_{θ_2} by $J(\theta_1, \theta_2 | \xi) = \sum_{t=0}^{T-1} l(\pi_{\theta_1}(\mathbf{x}_t), \pi_{\theta_2}(\mathbf{x}_t))$.

The objective of Imitation Learning is to minimize the expected surrogate loss along the distribution induced by the robot's policy:

$$\min_{\theta} E_{p(\xi|\pi_\theta)} J(\theta, \theta^* | \xi). \tag{1}$$

In Eq. 1, the distribution on trajectories and the cumulative surrogate loss are coupled, which makes this a challenging optimization problem. The field of Imitation Learning has considered two types of algorithmic solutions to this objective, off-policy learning and on-policy learning [13]. A common off-policy technique is Behavior Cloning, which samples from the supervisor's distribution and performs expected risk minimization on the demonstrations:

$$\theta^R = \operatorname*{argmin}_{\theta} E_{p(\xi|\pi_{\theta^*})} J(\theta, \theta^* | \xi).$$

The performance of the policy π_{θ^R} can be written in terms of the following decomposition:
$$E_{p(\xi|\pi_{\theta^R})} J(\theta^R, \theta^* | \xi)$$
$$= \underbrace{E_{p(\xi|\pi_{\theta^R})} J(\theta^R, \theta^* | \xi) - E_{p(\xi|\pi_{\theta^*})} J(\theta^R, \theta^* | \xi)}_{\text{Shift}} + \underbrace{E_{p(\xi|\pi_{\theta^*})} J(\theta^R, \theta^* | \xi)}_{\text{Loss}},$$

which corresponds to the covariate shift and the standard loss. In this work, we focus on minimizing the covariate shift. For references on how to minimize the standard loss see [16]. In the next section, we will show how noise injection can be used to reduce covariate shift.

4 Off-Policy Imitation Learning with Noise Injection

Noise injected into the supervisor's policy simulates error occurring during testing. Under this noise, the supervisor is forced to take corrective actions in order to successfully perform the task. The corrective examples allow the robot to learn how to recover when it deviates from the supervisor's distribution. However, because the demonstrations are still concentrated around the supervisor's policy, it can have less cognitive load on the supervisor than on-policy methods, which are concentrated around the robot's policy. The intuition of providing corrective examples during data collection is shown in Fig. 1.
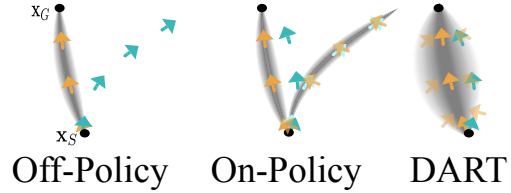

Figure 1: Robot Learning to reach a goal state $\mathbf{x}_G$. The grey denotes the distribution over trajectories. Left: Off-Policy learning in which the supervisor, the orange arrows, provides demonstrations. The robot, the teal arrows, deviates from the distributions and incurs high error. Middle: On-Policy which samples from the current robot's policy, the light teal arrows, to receive corrective examples from the supervisor. Right: DART, which injects noise to widen the supervisor's distribution and provides corrective examples. DART is off-policy but robust.

Denote by $p(\xi|\pi_{\theta^*}, \psi)$ a distribution over trajectories with noise injected into the supervisor's distribution $\pi_{\theta^*}(\mathbf{u}|\mathbf{x}, \psi)$. The parameter ψ represents the sufficient statistics that define the noise distribution. For example, if Gaussian noise is injected parameterized by ψ, $\pi_{\theta^*}(\mathbf{u}|\mathbf{x}, \psi) = \mathcal{N}(\pi_{\theta^*}(\mathbf{x}), \Sigma)$.

Similar to Behavior Cloning, we can sample demonstrations from the noise-injected supervisor and minimize the expected loss via standard supervised learning techniques. This can be written as follows:

$$\theta^R = \operatorname*{argmin}_{\theta} E_{p(\xi|\pi_{\theta^*}, \psi)} J(\theta, \theta^* | \xi) \tag{2}$$

Eq. 2, though, does not explicitly minimize the covariate shift for arbitrary choice of ψ. In order to reduce covariate shift, we introduce the following Lemma to bound the shift. We can then attempt

to optimize this bound with respect to ψ. This bound holds when the surrogate loss $\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U}$ $l(\mathbf{u}_1, \mathbf{u}_2) \in [0, 1]$. Examples of when this occurs is if the robot has a discrete set of actions or bounded continuous controls and they are normalized during learning.

Lemma 4.1 *If* $\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U}, \; 0 \leq l(\mathbf{u}_1, \mathbf{u}_2) \leq 1$ *the following is true for a time horizon of* T

$$|E_{p(\xi|\pi_{\theta^*}, \psi)} J(\theta^R, \theta^*|\xi) - E_{p(\xi|\pi_{\theta^R})} J(\theta^R, \theta^*|\xi)| \leq T \sqrt{\frac{1}{2} \mathcal{D}_{\mathcal{KL}} \big(p(\xi|\pi_{\theta^R}), p(\xi|\pi_{\theta^*}, \psi) \big)}$$

[See Appendix for Proof]

To minimize the covariate shift, we can optimize the KL-divergence, $\mathcal{D}_{\mathcal{KL}}$, with respect to the sufficient statistics, ψ, of the noise distribution. This can be reduced to the following optimization problem:

$$\min_{\psi} E_{p(\xi|\pi_{\theta^R})} - \sum_{t=0}^{T-1} \log[\pi_{\theta^*}(\pi_{\theta^R}(\mathbf{x}_t)|\mathbf{x}_t, \psi)]$$

This above optimization problem decreases the negative log-likelihood of the robot's control during data collection. Thus, we want to choose a noise parameter that makes the supervisor's distribution closer to the final robot's distribution. A clear limitation of this optimization problem is that it requires knowing the final robot's distribution $p(\xi|\pi_{\theta^R})$, which is determined only after the data is collected. The dependency on this term creates a *chicken and egg* problem. In the next section, we present DART, which applies an iterative approach to the optimization.

4.1 DART: Disturbances for Augmenting Robot Trajectories

The above objective cannot be solved because $p(\xi|\pi_{\theta^R})$ is not known until after the robot has been trained. We can instead iteratively sample from the supervisor's distribution with the current noise parameter, ψ_k, and minimize the negative log-likelihood of the noise-injected supervisor taking the current robot's, $\pi_{\hat{\theta}}$, control.

$$\hat{\psi}_{k+1} = \operatorname*{argmin}_{\psi} E_{p(\xi|\pi_{\theta^*}, \psi_k)} - \sum_{t=0}^{T-1} \log \left[\pi_{\theta^*}(\pi_{\hat{\theta}}(\mathbf{x}_t)|\mathbf{x}_t, \psi) \right] \tag{3}$$

However, the above iterative process can be slow to converge because it is optimizing the noise with respect to the current robot's policy and are optimizing on samples from the supervisor's distribution. We can obtain a better estimate by observing that the supervisor should simulate as much expected error as the final robot policy, $E_{p(\xi|\pi_{\theta^R})} J(\theta^R, \theta^*|\xi)$. It is possible that we have some knowledge of this quantity from previously training on similar domains.

Inspired by shrinkage estimation [3], we can obtain a better estimate of $\hat{\psi}$ by a target value based on the anticipated final robot's error and subsequently scaling the current simulated error to this level. Denote $E_{p(\xi|\pi_{\theta^*}, \hat{\psi})} \sum_{t=0}^{T-1} l(\mathbf{u}_t, \pi_{\theta^*}(\mathbf{x}_t))$, the expected deviation from the supervisor's control for a given surrogate loss. The scaling can be denoted

$$\psi_{k+1}^{\alpha} = \hat{\psi}_{k+1} * \operatorname*{argmin}_{\beta \geq 0} |\alpha - E_{p(\xi|\pi_{\theta^*}, \beta * \hat{\psi}_{k+1})} \sum_{t=0}^{T-1} l(\mathbf{u}_t, \pi_{\theta^*}(\mathbf{x}_t))| \tag{4}$$

where α corresponds to an estimated guess of the robot's final error and ψ_{k+1}^{α} corresponds to the sufficient statistic after the scaling has been applied. For Gaussian noise with covariance matrix Σ, the expected deviation has a closed form solution for the squared Euclidean loss: $E_{p(\xi|\pi_{\theta^*}, \hat{\psi})} \sum_{t=0}^{T-1} l(\mathbf{u}_t, \pi_{\theta^*}(\mathbf{x}_t))) = T \mathrm{tr}(\Sigma)$. Thus, a known closed form solution can be derived for both Eq. 3 and Eq. 4:

$$\Sigma_{k+1}^{\alpha} = \frac{\alpha}{T \mathrm{tr}(\hat{\Sigma}_{k+1})} \hat{\Sigma}_{k+1}, \quad \hat{\Sigma}_{k+1} = \frac{1}{T} E_{p(\xi|\pi_{\theta^*}, \Sigma_k^{\alpha})} \sum_{t=0}^{T-1} (\pi_{\hat{\theta}}(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t))(\pi_{\hat{\theta}}(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t))^T.$$

In Algorithm 1, we present DART, which iteratively solves for ψ_{k+1}^α and to collect data and train the final robot policy. First N demonstrations are collected from a supervisor with an initial noise parameter set. Then a policy, $\pi_{\hat\theta}$, is learned via empirical risk minimization. The learned policy is then used to optimize Eq. 3 based on sample estimates and the outcome is then scaled based on Eq. 4. Once the noise term is found N demonstrations are collected with the noise-injected supervisor and the robot is trained on the aggregate dataset. The algorithm is repeated for K iterations.

Algorithm 1: DART

Input: ψ_1^α, α
for $k = 1$ **to** K **do**
    for $n = 1$ **to** N **do**
        $\xi_{k,n} \sim p(\xi | \pi_{\theta^*}, \psi_k^\alpha)$
    end for
    $\hat\theta = \arg\min_\theta \sum_{i=1}^k \sum_{n=1}^N J(\theta, \theta^* | \xi_{i,n})$
    $\hat\psi_{k+1}$ is set with Eq. 3
    ψ_{k+1}^α is set with Eq. 4
end for
$\theta^R = \arg\min_\theta \sum_{k=1}^K \sum_{n=1}^N J(\theta, \theta^* | \xi_{k,n})$

4.2 Theoretical Analysis of DART

Compared to Behavior Cloning, DART reduces covariate shift by simulating errors the robot is likely to make. We show the following Proposition to provide better intuition for when this improvement occurs.

Proposition 4.1 *Given a deterministic supervisor's policy with Gaussian noise injected, $\pi^*(\mathbf{u}|\mathbf{x}, \psi) = \mathcal{N}(\pi^*(\mathbf{x}), \Sigma)$, when the robot policy has error $E_{p(\xi | \pi_{\theta^R})} J(\theta^R, \theta^* | \xi) > 0$. The following holds:*

$$\mathcal{D}_{\mathcal{KL}}\big(p(\xi | \pi_{\theta^R}), p(\xi | \pi_{\theta^*}, \Sigma)\big) < \mathcal{D}_{\mathcal{KL}}\big(p(\xi | \pi_{\theta^R}), p(\xi | \pi_{\theta^*})\big)$$

where the right hand side corresponds to Behavior Cloning. [See Appendix for Proof]

Proposition 4.1 shows that DART reduces an upper bound on covariate shift, or the distance between the distributions, more so than Behavior Cloning, when the robot has non-zero error with respect to the supervisor. We note this assumes the supervisor is not affected by the noise injection, which for human supervisor's this might not always be true. Thus, noise should be injected if the robot is expected to make some errors after training.

However, noise injection will offer no improvement if the robot can represent the supervisor perfectly and collect sufficient data. A similar result was shown in Laskey et al. [7] for when DAgger is expected to improve over Behavior Cloning. In practical applications, it is unlikely to obtain sufficient data to perfectly represent the supervisor, which highlights the need for DART.

5 Experiments

Our experiments are designed to explore: 1) Does DART reduce covariate shift as effectively as on-policy methods? 2) How much does DART reduce the computational cost and how much does it decay the supervisor's performance during data collection? 3) Are human supervisors able to provide better demonstrations with DART?

5.1 MuJoCo Locomotion Environments

To test how well DART performs compared with an on-policy methods such as DAgger and off-policy methods like Behavior Cloning, we use MuJoCo locomotion environments [22]. The challenge for these environments is that the learner does not have access to the dynamics model and must learn a control policy that operates in a high-dimensional continuous control space and moves the robot in a forward direction without falling.

We use an algorithmic supervisor in these domains: a policy which is trained with TRPO [17] and is represented as a neural network with two hidden layers of size 64. Note, while TRPO uses a stochastic policy, the supervisor is the deterministic mean of the learner. This is the same supervisor used in [6]. For all 4 domains, we used the same neural network as the supervisor and trained the
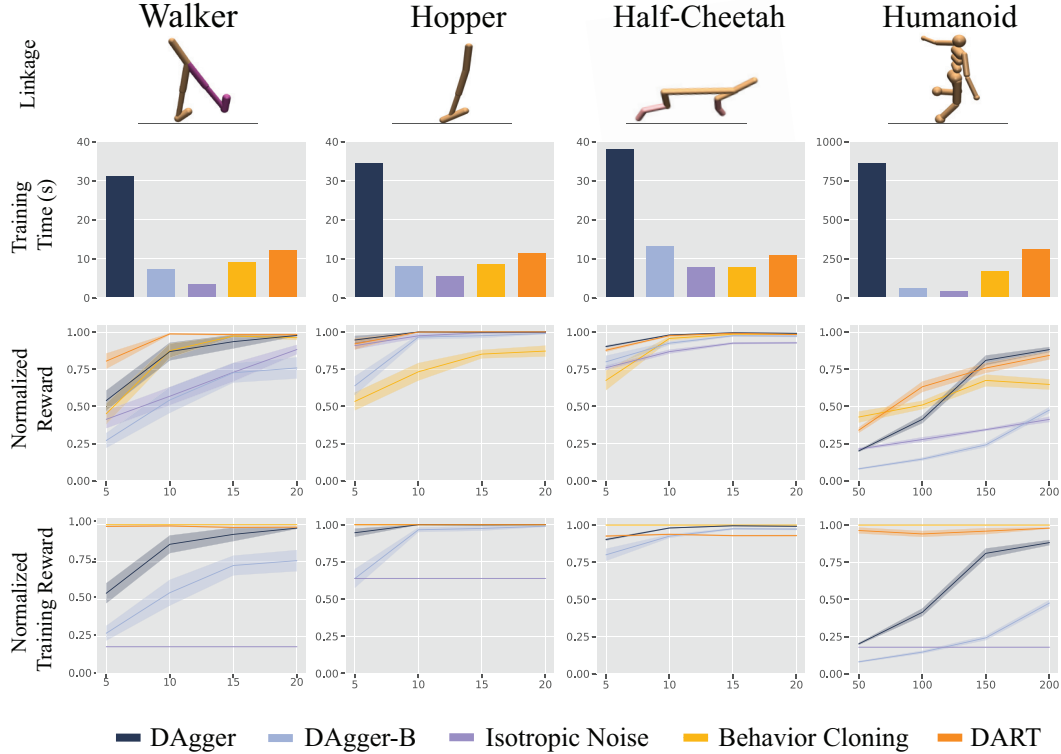
Figure 2: Top: The four different locomotive domains in MuJoCo we evaluated DART on: Walker, Hopper, Half-Cheetah and Humanoid. Top Middle: The time, in seconds, to achieve the performance level reported below. DART achieves similar performance to DAgger in all 4 domains, but requires significantly less computation because it doesn't require retraining the current robot policy after each demonstration. DAgger-B reduces the computation required by less frequently training the robot, but suffers significantly in performance in domains like Humanoid. Bottom Middle: The learning curve with respect to reward obtained during training with each algorithm plotted across the number of demonstrations. Bottom: The reward obtained during collection of demonstrations for learning. DART receives near the supervisor's reward at all iterations whereas DAgger can be substantially worse in the beginning.

policies in Tensorflow. At each iteration we collected one demonstration. In order to make the task challenging for Imitation Learning, we used the same technique as in [6], which is to sub-sample 50 state and control pairs from the demonstrated trajectories, making the learners receive less data at each iteration.

We compare the following five techniques: Behavior Cloning, where data is collected from the supervisor's distribution without noise injected; DAgger, which is an on-policy method that stochastically samples from the robot's distribution and then fully retrains the robot's policy after every demonstration is collected; DAgger-B, which is DAgger but the retraining is done intermittently to reduce computation time; Isotropic-Gaussian which injects non-optimized isotropic noise; and DART. See supplement material for how the hyper-parameters for each method are set.

We used the following MuJoCo environments: Walker, Hopper, Humanoid, and Half-Cheetah. The size of the state and control space for each task is $\{|\mathcal{X}| = 17, |\mathcal{U}| = 6\}$, $\{|\mathcal{X}| = 11, |\mathcal{U}| = 3\}$, $\{|\mathcal{X}| = 376, |\mathcal{U}| = 17\}$, $\{|\mathcal{X}| = 117, |\mathcal{U}| = 6\}$, respectively. All experiments were run on a MacBook Pro with an 2.5 GHz Intel Core i7 CPU. We measured the cumulative reward of each learned policy by rolling it for 500 timesteps, the total computation time for each method and the cumulative reward obtained during learning.

Fig. 2 shows the results. In all domains, DART achieves parity with DAgger, whereas Behavior Cloning and DAgger-B are below this performance level in Walker and Humanoid. For Humanoid, DART is $3x$ faster in computation time and during training only decreases the supervisor's cumulative reward by 5%, whereas DAgger executes policies that have over 80% less cumulative reward than the supervisor. The reason for this is that DAgger requires constantly updating the current robot policy and forces the robot into sub-optimal states. While, one way to reduce this computation is to decrease
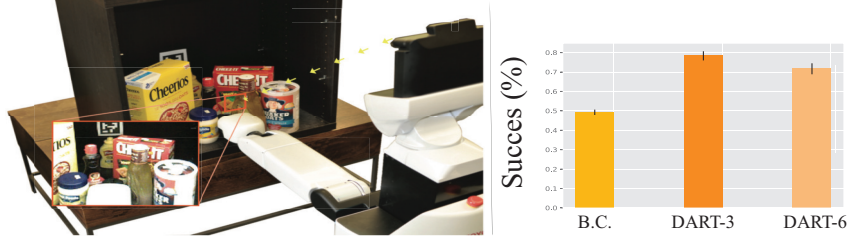
Figure 3: Left: Experimental setup for the grasping in clutter task. A Toyota HSR robot uses a head-mounted RGBD camera and its arm to push obstacle objects out of the way to reach the goal object, a mustard bottle. The robot's policy for pushing objects away uses a CNN trained on images taken from the robot's Primesense camera, an example image from the robot's view point is shown in the orange box. Right: the Success Rate for Behavior Cloning, DART($\alpha = 3$) and DART($\alpha = 6$). DART($\alpha = 3$) achieves the largest success rate.

the number of times the policy is updated. DAgger-B illustrates that this can significantly deteriorate performance. Lastly, naively applying isotropic noise does not perform well, and leads to unsafe policies during execution, which suggests the need for optimizing the level of noise.

5.2 Robotic Grasping in Clutter

We evaluate DART with human supervisors in a grasping in clutter task on a Toyota HSR robot. We consider a task inspired by the Amazon Picking Challenge [2]. The goal of the task is for the robot to retrieve a goal object from a cupboard. The task is challenging because the goal object is occluded by obstacle objects and the robot must reason about how to clear a path based on observations taken from an eye-in-hand perspective. The objects are 6 common household food items, which consist of boxes and bottles with varying textures and mass distributions. The target object is fixed to always be a mustard bottle. The robot, task and image viewpoint are shown in Fig. 3. See supplement material for additional information on the task.

We use 4 supervisors who have robotics experience but not specifically in the field of Imitation Learning and compare Behavior Cloning and DART. When performing the study, we first collect $N = 10$ demonstrations with Behavior Cloning (i.e. no noise) and then in a counter-balanced ordering collect $N = 30$ more demonstrations with each technique. Our experiment was within-subject, have every supervisor perform all three methods.

We only updated the noise parameter after the first 10 demonstrations (i.e. $K = 2$). The final robot policy was then trained on the total of 40 demonstrations. We consider two different choices of α: $\alpha = 3T\text{tr}(\hat{\Sigma}_1)$ and $\alpha = 3T\text{tr}(\hat{\Sigma}_1)$. These choices correspond to the intuition that with small datasets in high dimensional image space the robot will have significant shift from the current loss on the supervisor's distribution. Through the rest of the paper, we will write our choices as $\alpha = 3$ and $\alpha = 6$ for brevity in notation.

During policy evaluation, we measured success as 1) the robot is able to identify the goal object and 2) there being a clear path between its gripper and the object. Once these conditions are identified, as detailed in the supplement material, an open-loop motion plan is generated to execute a grasp around the target object.

In Fig. 3, we report the average performance of the three techniques across DART 20 times on different sampled initial states. DART with $\alpha = 3$ performs the best out of the three techniques, with a 79% success rate over traditional Behavior Cloning's 49% success rate. Interestingly, DART with $\alpha = 6$ performs better than Behavior Cloning, but only has a 72% success rate. This may suggest this level of noise was potentially too high for the human supervisor.

6 Conclusion

This paper considers injecting noise into the supervisor's demonstrations to minimize covariate shift. Our algorithm, DART, provides the robot with corrective examples at the boundary of the supervisor's policy. By being concentrated around the supervisor's policy, it collects demonstrations without visiting highly sub-optimal states and is easier for human supervisors tele-operating. We demonstrate it achieves parity with on-policy methods in simulated MuJoCo domains and is significantly better than traditional Behavior Cloning on a real robot. For code and more information see http://berkeleyautomation.github.io/DART/

7   Acknowledgements

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman. Lessons from the amazon picking challenge. *arXiv preprint arXiv:1601.05484*, 2016.

[3] M. J. Daniels and R. E. Kass. Shrinkage estimators for covariance matrices. *Biometrics*, 57(4): 1173–1184, 2001.

[4] M. Green and J. B. Moore. Persistence of excitation in linear systems. *Systems & control letters*, 7(5):351–360, 1986.

[5] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *NIPS*, pages 3338–3346. 2014.

[6] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[7] M. Laskey, C. Chuck, J. Lee, J. Mahler, S. Krishnan, K. Jamieson, A. Dragan, and K. Goldberg. Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations. *arXiv preprint arXiv:1610.00850*, 2016.

[8] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg. Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations. In *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, pages 827–834. IEEE, 2016.

[9] A. Lesne. Shannon entropy: a rigorous notion at the crossroads between probability, information theory, dynamical systems and statistical physics. *Mathematical Structures in Computer Science*, 24(3), 2014.

[10] P. Z. Marmarelis and V. Z. Marmarelis. The white-noise method in system identification. In *Analysis of physiological systems*, pages 131–180. Springer, 1978.

[11] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, Carnegie-Mellon University, 1989.

[12] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[13] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pages 661–668, 2010.

[14] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1011.0686*, 2010.

[15] S. Sastry and M. Bodson. *Adaptive control: stability, convergence and robustness*. Courier Corporation, 2011.

[16] B. Schölkopf and A. J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[17] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1889–1897, 2015.

[18] J. Schulman, J. Ho, C. Lee, and P. Abbeel. Learning from demonstrations through the use of non-rigid registration. In *Robotics Research*, pages 339–354. Springer, 2016.

[19] W. A. Sethares and C. Johnson Jr. Persistency of excitation and (lack of) robustness in adaptive systems. In *Advances in Computing and Control*, pages 340–351. Springer, 1989.

[20] W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. *arXiv preprint arXiv:1703.01030*, 2017.

[21] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

[22] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[23] A. Vlachos. An investigation of imitation learning algorithms for structured prediction. In *European Workshop on Reinforcement Learning*, pages 143–154, 2013.

[24] J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

8 Appendix

8.1 Proofs for Theoretical Analysis

Lemma 4.1 *If $\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U}, \; 0 \leq l(\mathbf{u}_1, \mathbf{u}_2) \leq 1$, the following is true:*

$$|E_{p(\xi|\pi_{\theta^*}, \psi)} J(\theta, \theta^*|\xi) - E_{p(\xi|\pi_\theta)} J(\theta, \theta^*|\xi)| \leq T\sqrt{\frac{1}{2}\mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*}, \psi)\big)}$$

Proof:

$$|E_{p(\xi|\pi_{\theta^*}, \psi)} J(\theta, \theta^*|\xi) - E_{p(\xi|\pi_\theta)} J(\theta, \theta^*|\xi)| \tag{5}$$

$$\leq T||p(\xi|\pi_{\theta^*}, \psi) - p(\xi|\pi_\theta)||_{TV} \tag{6}$$

$$\leq T\sqrt{\frac{1}{2}\mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*}, \psi)\big)} \tag{7}$$

The first line of the proof follos from Lemma 4.2, which is stated below and the fact $\forall \mathbf{u}_1, \mathbf{u}_2 \; 0 \leq l(\mathbf{u}_1, \mathbf{u}_2) \leq 1$. The second line follows from Pinsker's Inequality. ∎

Lemma 4.2 *Let P and Q be any distribution on $\mathcal{X}$. Let $f : \mathcal{X} \to [0, B]$. Then*

$$|E_P[f(x)] - E_Q[f(x)]| \leq B||P - Q||_{TV}$$

Proof:

$$|E_P[f(x)] - E_Q[f(x)]| = |\int_x p(x)f(x)dx - \int_x q(x)f(x)dx|$$

$$= |\int_x (p(x) - q(x))f(x)dx|$$

$$= |\int_x (p(x) - q(x))(f(x) - \frac{B}{2})dx$$

$$+ \frac{B}{2}\int_x (p(x) - q(x))dx|$$

$$\leq \int_x |p(x) - q(x)||f(x) - \frac{B}{2}|dx$$

$$\leq \frac{B}{2}\int_x |p(x) - q(x)|dx$$

$$\leq B||P - Q||_{TV}$$

The last line applies the definition of total variational distance, which is $||P - Q||_{TV} = \frac{1}{2}\int_x |p(x) - q(x)|$. ∎

Proposition 4.1 *Given a supervisor's policy with Gaussian noise injected, $\pi^*(\mathbf{u}|\mathbf{x}, \psi) = \mathcal{N}(\pi^*(\mathbf{x}), \Sigma)$, when the robot policy has error $E_{p(\tau|\pi_\theta)}J(\theta, \theta^*|\xi) > 0$. The following is true:*

$$\mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*}, \Sigma)\big) < \mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*})\big)$$

where the right hand side corresponds to Behavior Cloning.

Proof: We begin the proof with the definition of the KL-divergence:

$$\mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*})\big) = E_{p(\xi|\pi_\theta)} \log \frac{\prod_{t=0}^{T-1} \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\prod_{t=0}^{T-1} \pi_{\theta^*}(\mathbf{u}_t|\mathbf{x}_t)} \qquad (8)$$

$$= E_{p(\xi|\pi_\theta)} \sum_{t=}^{T-1} \log \frac{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\pi_{\theta^*}(\mathbf{u}_t|\mathbf{x}_t)} \qquad (9)$$

We know that if $E_{p(\tau|\pi_\theta)}J(\theta, \theta^*|\psi) = 0$, then the KL divergence would be zero because at all states under the distribution the two policies would perfectly agree. However, if they do not agree then the KL-divergence becomes the there $\exists \xi$ such that $p(\xi|\pi_{\theta^*}) = 0$ when $p(\xi|\pi_\theta) > 0$, which implies [9].

$$\mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*})\big) = \infty$$

One technique to correct for this is to ensure that the supervisor has finite probability of applying any control (i.e. $\forall \xi \; p(\xi|\pi_{\theta^*}) > 0$ when $p(\xi|\pi_\theta) > 0$). Injecting Gaussian noise ensures non-zero probability for all controls at a given trajectory and subsequently:

$$\mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*}, \Sigma)\big) < \infty$$

∎

4.2 Derivation for Optimizing Noise

When optimizing noise, we need to first optimize Eq. 3, which finds the noise parameter that maximizes the likelihood of the current robot policy and then we can scale the parameter to our prior of the error in the final robot's policy.

In this section, we re-derive the closed from solutions for the Gaussian and ϵ-greedy scenario for the MLE objective of the current robot's control. We then derive a solution for Eq. 4 for the Gaussian case.

Gaussian MLE We begin with the continuous case where the supervisor's policy is stochastic with distribution $\pi_{\theta^*}(\mathbf{u}_t|\mathbf{x}_t, \Sigma)$ defined by $\mathcal{N}(\pi_{\theta^*}(\mathbf{x}_t, \Sigma))$ and the learner's policy is given by $\pi_{\hat{\theta}}$. Then the optimization problem is

$$\hat{\Sigma} = \arg\min_{\Sigma} -E_{p(\xi|\pi_{\theta^*})} \sum_{t=0}^{T-1} \log\left[\pi_{\theta^*}(\pi_{\hat{\theta}}(\mathbf{x}_t)|\mathbf{x}_t, \Sigma)\right]$$

$$= \arg\min_{\Sigma} E_{p(\xi|\pi_{\theta^*})} -\frac{T}{2}\log\det\Sigma^{-1} + \frac{1}{2}\sum_{t=0}^{T-1}\left(\pi_{\hat{\theta}}(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)\right)^T \Sigma^{-1}\left(\pi_{\hat{\theta}}(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)\right).$$

We take the derivative with respect to Σ and set it equal to zero:

$$-T\Sigma + E_{p(\xi|\pi_{\theta^*})} \sum_{t=0}^{T-1}\left(\pi_{\hat{\theta}}(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)\right)\left(\pi_{\hat{\theta}}(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)\right)^T.$$

Then the optimal covariance matrix is

$$\hat{\Sigma} = \frac{1}{T}E_{p(\xi|\pi_{\theta^*})} \sum_{t=0}^{T-1}\left(\pi_{\hat{\theta}}(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)\right)\left(\pi_{\hat{\theta}}(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)\right)^T.$$

ϵ**-greedy MLE** For the ϵ-greedy case in the discrete action domain with a finite number of controls K, the probability of the supervisor choosing a control at a particular state is given by

$$\pi_{\theta^*}(\mathbf{u}_t|\mathbf{x}_t, \epsilon) = \begin{cases} 1 - \epsilon & \pi_{\theta^*}(\mathbf{x}_t) = \mathbf{u}_t \\ \frac{\epsilon}{K-1} & \text{otherwise} \end{cases}.$$

For any $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U}$, let the loss function $l(\mathbf{u}_1, \mathbf{u}_2)$ be defined as the indicator function so that $l(\mathbf{u}_1, \mathbf{u}_2) = 0$ for $\mathbf{u}_1 = \mathbf{u}_2$ and 1 otherwise. Then the optimization problem in Eq. 3 becomes

$$\hat{\epsilon} = \arg\min_{\epsilon} -E_{p(\xi|\pi_{\theta^*})} \sum_{t=0}^{T-1} \log\left[\pi_{\theta^*}(\pi_{\hat{\theta}}(\mathbf{x}_t)|\mathbf{x}_t, \epsilon)\right]$$

$$= \arg\min_{\epsilon} -E_{p(\xi|\pi_{\theta^*})}J(\hat{\theta}, \theta^*|\xi)\log\left[\frac{\epsilon}{K-1}\right] + (T - J(\hat{\theta}, \theta^*|\xi))\log\left[1 - \epsilon\right]$$

The previous line follows from the fact that along any given trajectory ξ from π_{θ^*}, the probability of ξ can be written using only $J(\hat{\theta}, \theta^*|\xi)$, which is the number times that $\pi_{\hat{\theta}}$ and π_{θ^*} disagree along ξ. Note that this is convex in ϵ. By taking the derivative with respect to ϵ and setting it equal to zero, we get

$$\frac{E_{p(\xi|\pi_{\theta^*})}J(\hat{\theta}, \theta^*|\xi)}{\epsilon} - \frac{T - E_{p(\xi|\pi_{\theta^*})}J(\hat{\theta}, \theta^*|\xi)}{1 - \epsilon} = 0.$$

Therefore, we have

$$\hat{\epsilon} = \frac{1}{T}E_{p(\xi|\pi_{\theta^*})}J(\hat{\theta}, \theta^*|\xi).$$

Gaussian Scaling Given an optimized covaraince matrix $\hat{\Sigma}$, we wish to scale it to some prior over what we expect the robot's final error to be.

$$\arg\min_{\beta \geq 0} |\alpha - E_{p(\xi|\pi_{\theta^*}, \beta*\hat{\Sigma})} \sum_{t=0}^{T-1} l(\mathbf{u}_t, \pi_{\theta^*}(\mathbf{x}_t)))|$$

$$= \arg\min_{\beta \geq 0} |\alpha - E_{p(\xi|\pi_{\theta^*}, \beta*\hat{\Sigma})} \sum_{t=0}^{T-1} ||\mathbf{u}_t - \pi_{\theta^*}(\mathbf{x}_t)))||_2^2$$

$$= \arg\min_{\beta \geq 0} |\alpha - \beta T \mathrm{tr}(\hat{\Sigma})|$$

Where the last line followed from known properties of the multi-variate Gaussian and the fact that the level of noise is independent of the current state $\mathbf{x}_t$. We can derive a solution for β.

$$= \frac{\alpha}{T\text{tr}(\hat{\Sigma})}$$

4.3 Additional Information for MuJoCo Experiments

The following hyperparameters were used for all domains. For DAgger and DAgger-B, we swept several values for β, the stochastic mixing hyperparameter, and found that setting $\beta = 0.5$ yields the best results. For noise injection algorithms, we chose $\alpha = T\text{tr}(\hat{\Sigma}_k)$, which corresponds to no additional knowledge being used. For Isotropic-Gaussian noise injection, we set $\Sigma_k^\alpha = I$ for all k. Additionally, the covariance matrix for DART was always estimated using held-out demonstrations collected in prior iterations.

In the Walker, Hopper and HalfCheetah domains, we ran these algorithms for 20 iterations, evaluating the learners at 5, 10, 15, and 20 iterations. One initial supervisor demonstration was always collected in the first iteration. To remain consistent, we updated DAgger-B and DART at the same iterations: iteration 2 and iteration 8.

In the Humanoid domain, we ran the algorithms for 200 iterations and evaluated the learners at iterations 50, 100, 150, and 200. Again, we collected one initial supervisor demonstration for DAgger and Isotropic-Gaussian. For DAgger-B and DART, we collected 50 initial supervisor demonstrations before retraining every 25 iterations.

4.4 Additional Experiments in MuJoCo Domains

To better understand how DART was able to achieve the reported performance, we asked the additional questions: 1) Does DART reduce the covariate shift? 2) How well do other noise injection methods perform against DART?

Reducing Covariate Shift To test how effective DART was at reducing the shift, we evaluate the surrogate loss on both the supervisor's distribution $p(\xi|\pi_\theta^*, \hat{\psi})$ and the robot's distribution $p(\xi|\pi_\theta^*)$. In Fig. 4, we report the losses for each algorithm. Both DAgger and DART are capable of reducing the covariate shift, which is reflected by the smaller disparity between the losses on the supervisor distributions and those on the robot distributions.

Random Covariance Matrices We also compared DART against learners that obtained data from supervisors with random covariance matrices. We define the simulated error of a noisy supervisor to be the expected error of the supervisor on its noisy distribution. In the Gaussian case, note that the expected squared L2 error that the noisy supervisor simulates is equivalent to the trace of the covariance matrix. That is,

$$E_{p(\xi|\pi_{\theta^*}, \hat{\psi})} \left[\frac{1}{T} \sum_{t=0}^{T-1} \|\mathbf{u}_t - \pi_{\theta^*}(\mathbf{x}_t)\|_2^2 \right] = \text{tr } E_{p(\xi|\pi_{\theta^*}, \hat{\psi})} \left[\frac{1}{T} \sum_{t=0}^{T-1} (\mathbf{u}_t - \pi_{\theta^*}(\mathbf{x}_t))(\mathbf{u}_t - \pi_{\theta^*}(\mathbf{x}_t))^T \right]$$
$$= \text{tr } (\Sigma),$$

where $\hat{\psi}$ corresponds to the parameters of the multivariate Gaussian distribution with zero mean and covariance matrix Σ. Thus, the amount of simulated error can be tuned by tuning the trace of the covariance matrix. We used this fact generate random covariance matrices drawn from the Wishart distribution and scaled the traces of the covariance matrices such that they simulated losses of 0.005, 0.5, and 5.0 for Walker, Hopper, and HalfCheetah and 0.005, 0.5, and 10.0 for Humanoid. The results, reported in Fig. 5, suggest that, as long as the simulated error is carefully chosen, randomly sampled covariance matrices may perform as well as DART; however it is not often known in practice what the simulated error should be before training the learner and evaluating it.

4.5 Additional Information on Robot Setup

During policy evaluation, we measured success as 1) the robot is able to identify the goal object and 2) there being a clear path between its gripper and the object. Once these conditions are identified an
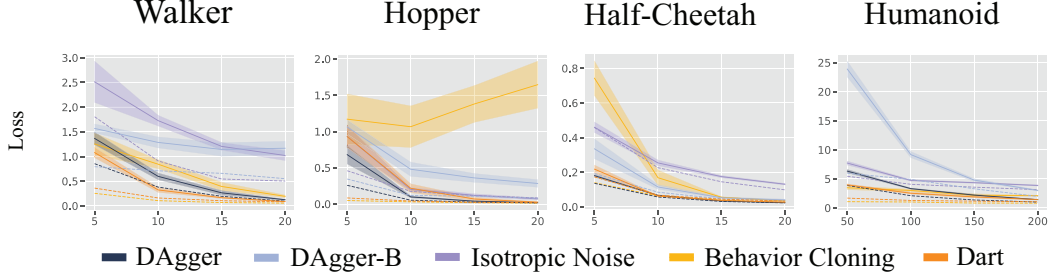
Figure 4: The covariate shift on the four different locomotive domains in MuJoCo we evaluated DART on: Walker, Hopper, Half-Cheetah and Humanoid. The dashed lines correspond to the surrogate losses on the supervisor distributions, $E_{p(\xi|\pi_{\theta^*}, \psi_k^\alpha)} J(\theta^R, \theta^*|\xi)$, and the solid lines correspond to the loss on the robot distributions $E_{p(\xi|\pi_{\theta R})} J(\theta^R, \theta^*|\xi)$. DART is able to reduce the covariate shift in all four domains, which is illustrated by the dash line converging to the solid line. Additionally, DAgger is also able to reduce the shift. However, Behavior Cloning is less effective at doing so, which is illustrated by a slower convergence rate.
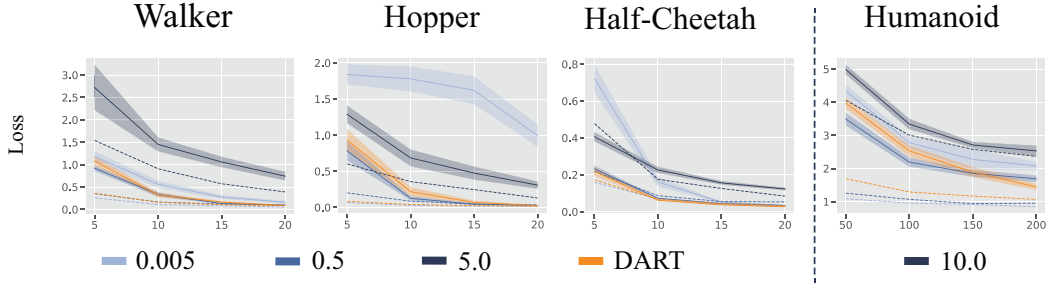

Figure 5: The losses of learners trained from supervisors with randomly sampled covariance matrices are compared against the losses of DART. Again, the dashed lines represent the loss on the supervisor disributions while the solid lines represent the loss on the robot distributions. When the simulated error is well-chose, such as $Tr(\Sigma) = 0.5$ in these experiments, the performance matches that of DART and the covariate shift is reduced. However higher or lower levels of the noise can cause drastically higher losses.

open-loop motion plan is generated to execute a grasp around the target object. In order for the robot to identify the target object, we use the Faster R-CNN trained on the PASCAL VOC 2012 dataset, which has a bottle object category [12]. Once the mustard bottle has been identified and the a path is cleared a human supervisor tells the robot to execute the open-loop motion plan towards a fix position that the goal object is at. The human supervisor is used to ensure that the success criteria is not prone to error in the system, but the supervisor is blinded to which policy is being evaluated to prevent biases. We evaluate the policy 20 times on different sampled initial states. A video of the learned policy and the robot setup can be found here: https://youtu.be/LfMD69llesg.