Dart: Optimizing Noise Injection for Imitation Learning

Michael Laskey Jonathan Lee Roy Fox Anca Dragan Ken Goldberg
Department of Electrical Engineering and Computer Sciences
University of California 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. Interactive techniques alleviate this by iteratively collecting corrective actions for the current robot policy. However, these techniques can be difficult for human supervisors, add significant computation burden, and require the robot to visit potentially dangerous states during training. We examine an alternative: *injecting noise* into the supervisor's policy during the demonstration phase. This forces the supervisor and robot to explore and recover from errors without letting them compound. We propose a new algorithm, Dart, 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 provide a theoretical analysis to illustrate that Dart reduces covariate shift more than Behavior Cloning for a robot with non-zero error. We evaluate Dart in two domains: in simulation with an algorithmic supervisor on the MuJoCo locomotive tasks and in physical experiments with human supervisors training a Toyota HSR robot to perform grasping in clutter. For challenging tasks like Humanoid, Dart can be up to 280% 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 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. While deep reinforcement learning has the potential to learn such control policies, in practice it may require a very large number of samples [16].

Rather than pure ab initio learning, an alternative is to leverage supervision to guide the robot's policy. An intuitive 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 [10, 1]. This is an *off*-policy method, and suffers from the fact that errors compound when the robot executes what it has learned, leading it to drift to new and possibly dangerous states [13]. Theoretically, the drifting occurs because of covariate shift: 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 [12, 13] 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 [12, 13]. 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 challenging for human supervisors to provide these corrections [7] 2) it can be potentially dangerous for a physical robot to visit highly sub-optimal states [22], and 3) repeatedly updating the robot's policy is computationally expensive. In this paper, we introduce an alternative approach to address covariate shift.

1th Conference on Robot Learning (CoRL 2017), Mountain View, United States.

One way to show the robot corrective examples and prevent drift is to inject noise into the supervisor's demonstrations. 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 do Behavior Cloning, but to inject an optimized level of noise into the supervisor's control stream when we acquire demonstrations.

Injecting noise is only helpful if we select 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 optimizing its parameters by maximizing the likelihood between the state distributions induced by the robot and by the supervisor. We propose Dart: a noise injection hybrid of DAgger and 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 shows it can reduce covariate shift more so than Behavior Cloning when the robot has non-zero error 3) experimentation with algorithmic and human supervisors to show noise injection can reduce covariate shift.

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 non-linear robot is trained to walk forward, Dart can be at parity with state of the art on-policy methods, such as DAgger. For challenging tasks like Humanoid, Dart can be up to 280% faster in computation time and during training only decreases the supervisor's cumulative reward by 5%, whereas DAgger executes policies that has 80% less cumulative reward than the supervisor. Thus, suggesting corrections *at the boundary* 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.

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 [10, 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 [10]. 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 [12].

Ross et al. [13] 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 [22]. Further, on-policy methods require retraining the policy from scratch after each round of corrections.

We note 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 [18]. However, it has been shown local gradient updates suffer in performance compared to full retraining of the policy [21]. 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 [22]. However, this approach inherits the other limitations of on-policy, which is being challenging for human supervisors and computationally expensive.

In this paper, we are interested in reducing the covariate shift without incurring these limitations. We propose injecting zero-mean noise to the supervisor's control stream to simulate errors near the boundary of the supervisor's state distribution. Our hypothesis is that a supervisor could be robust to zero-mean noise and provide consistent demonstrations.

2.2 Noise Injection to Increase Robustness

Andereson et al. noted a phenomenon similar to the covariate shift when performing model identification [17]. 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 [14].

One way to achieve this condition 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 [9]. In light of this, we study the equivalent of this idea in learning a model-free policy: we inject 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 what the supervisor demonstrated.

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 $\pi_\theta : \mathcal{X} \to \mathcal{U}$ parametrized 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 in Reinforcement Learning [19], 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 $l : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}$ [13, 12]. The surrogate loss we consider is the L2-norm on the control vectors $l(\mathbf{u}_1, \mathbf{u}_2) = ||\mathbf{u}_1 - \mathbf{u}_2||_2^2$. 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 agent'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 [12]. In the next, section we will show how noise injection can be used to optimize an upper-bound of this objective.

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 a policy, so when the robot deviates from the supervisor it is able to recover. However, because the demonstrations are still concentrated around the supervisor, 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.

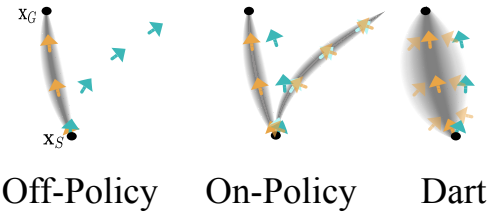

Figure 1: Robot Learning to Reach a Goal State $\mathbf{x}_G$. Left: Off-Policy learning in which the supervisor, the orange line, provides demonstrations. The robot, the teal line, deviates from the distributions and incurs high error. Middle: On-Policy which samples from the current robot's policy, the light teal line, to receive corrective examples from the supervisor. Right: Dart, which injects noise to widen the supervisor's distribution and provides corrective examples. Both On-Policy and Dart learn to head towards the goal state, but Off-Policy does not.

Simulating the robot's error can be challenging because it is not clear what the magnitude and direction the final robot's error will have. Determining this can become especially problematic in high-dimensional control problem such as a Humanoid robot. We will now present a method to do so by considering the test error on the supervisor's distribution, we refer to this algorithm as Dart.

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

As noted above, Eq. 1 cannot be directly optimized. However, 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:

$$\min_{\theta} E_{p(\xi|\pi_{\theta^*}, \psi)} J(\theta, \theta^*|\xi) \tag{2}$$

Eq. 2, though, does not indicate how well the robot will perform on its own distribution, or during execution of its policy. We introduce the following Lemma to bound the difference between performance on the supervisor distribution and performance on the robot's. This holds when the surrogate loss $l(\mathbf{u}, \mathbf{u}) \in [0, 1]$. Examples of when this occurs is if the robot has bounded controls and they are normalized during learning.

Lemma 4.1 *If* $\forall \mathbf{u}, \mathbf{u}\ 0 \leq l(\mathbf{u}, \mathbf{u}) \leq 1$ *the following is true:*

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

[See Appendix for Proof]

In order to minimize this difference, or covariate shift, we can optimize the KL-divergence with respect to the sufficient statistics, ψ, of the noise distribution. This can be formalized as the following optimization problem:

$$\psi^* = \underset{\psi}{\operatorname{argmin}}\ \mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*}, \psi)\big) \tag{3}$$

Optimizng a KL-divergence has been shown to be equivalent to maximixing the log-likilihood of matching the source distribution [2]. In this case, the robot's distribution is the source and ψ is computed to try and generate this distribution. The objective can be solved in closed form for the Gaussian noise distributions:

$$\Sigma^* = \frac{1}{T} E_{p(\xi|\pi_\theta)} \sum_{t=0}^{T-1} \big(\pi_{\theta^*}(\mathbf{x}_t) - \pi_\theta(\mathbf{x}_t)\big)\big(\pi_{\theta^*}(\mathbf{x}_t) - \pi_\theta(\mathbf{x}_t)\big)^T \tag{4}$$

[Derivation in the Appendix]

This closed form solution captures the intuitive idea of using noise to simulate the robot's error. Eq. 4 examines the error in each direction along the control vector and scales the covariance vector to inject noise in the directions with the highest error. For example, if a robot only has error in one control dimension, noise would only be injected in this dimension.

However, Eq. 4 computes the noise term with respect to the *current* robot. Ideally, we would want to inject noise with respect to the *final* robot's error because that is the policy we want to make robust. Furthermore, evaluating Eq. 4 will require sampling from the robot's policy, which is not ideal because this can be potentially unsafe and has been shown to be challenging for human supervisors [7]. Thus, in the next section we present an approximation algorithm which alleviates these problems.

4.1 Dart

Instead of sampling from the robot's distribution, $p(\xi|\pi_\theta)$, we can sample from the supervisor's distribution $p(\xi|\pi_{\theta^*})$ and use importance sampling to re-weight the distribution:

$$\Sigma^* = \frac{1}{T} E_{p(\xi|\pi_{\theta^*})} \sum_{t=0}^{T-1} \big(\pi_{\theta^*}(\mathbf{x}_t) - \pi_\theta(\mathbf{x}_t)\big)\big(\pi_{\theta^*}(\mathbf{x}_t) - \pi_\theta(\mathbf{x}_t)\big)^T \frac{\prod_{i=0}^{t} \pi_\theta(\mathbf{u}_i|\mathbf{x}_i)}{\prod_{i=0}^{t} \pi_{\theta^*}(\mathbf{u}_i|\mathbf{x}_i)}$$

Importance sampling could yield a very high variance estimate: since $\pi_\theta(\mathbf{u}|\mathbf{x})$ is deterministic, the robot would need to match the supervisor's controls exactly. Thus, we propose a biased estimate, which examines the test error on the current supervisor's distribution to infer the direction of the error, but a hyper-parameter is added to set the size, or magnitude. The intuition for this is that during learning the components of the control vector that are hard to learn will always remain challenging, but the overall level of error will decay at a unknown rate as a demonstrations are added. Furthermore, scaling the optimized covariance may be necessary when working with human supervisor because of the cognitive burden too much noise can create. This yields the following update for the noise term:

$$\Sigma^\alpha = \frac{\alpha}{T} E_{p(\xi|\pi_{\theta^*})} \sum_{t=0}^{T-1} \big(\pi_{\theta^*}(\mathbf{x}_t) - \pi_\theta(\mathbf{x}_t)\big)\big(\pi_{\theta^*}(\mathbf{x}_t) - \pi_\theta(\mathbf{x}_t)\big)^T \tag{5}$$

In Algorithm 1, we present Dart. 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. 5 based on sample estimates and the covariance matrix is scaled via α. 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: α, Σ_0
for $k = 1$ **to** K **do**
  for $n = 1$ **to** N **do**
    $\xi_{k,n} \sim p(\xi|\pi_{\theta^*}, \Sigma_k^\alpha)$
  end for
  $\hat{\theta} = \arg\min_\theta \sum_{i=1}^{k} \sum_{n=1}^{N} J(\theta, \theta^*|\xi_{i,n})$
  Σ_k^α is set with Eq. 5
end for

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 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* $E_{p(\xi|\pi_\theta)} J(\theta, \theta^*|\xi) > 0$*. 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. [See Appendix for Proof]

Proposition 4.1 shows that Dart reduces 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 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 answer the following questions: 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 decays 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 against on-policy methods such as DAgger and off-policy methods like Behavior Cloning, we use MuJoCo locomotion environments [20]. The challenge for these environments is 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 agent in a forward direction without falling.

We use an algorithmic supervisor in these domains: a policy whic is trained with TRPO [15] and is represented as a neural network with a two hidden layer of size 64. This is the same supervisor used in [6]. For all 4 domains, we used the same neural network as the supervisor and trained the 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 policies 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 space and control space for each task is $\{|\mathbf{x}| = 17, |\mathbf{u}| = 6\}$, $\{|\mathbf{x}| = 11, |\mathbf{u}| = 3\}$, $\{|\mathbf{x}| = 376, |\mathbf{u}| = 17\}$, $\{|\mathbf{x}| = 117, |\mathbf{u}| = 6\}$, 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 is at parity with DAgger, whereas Behavior Cloning and DAgger-B are below this performance level in Walker and Humanoid. For Humanoid, Dart is 280% 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 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 suggest the need for optimizing the level of noise.

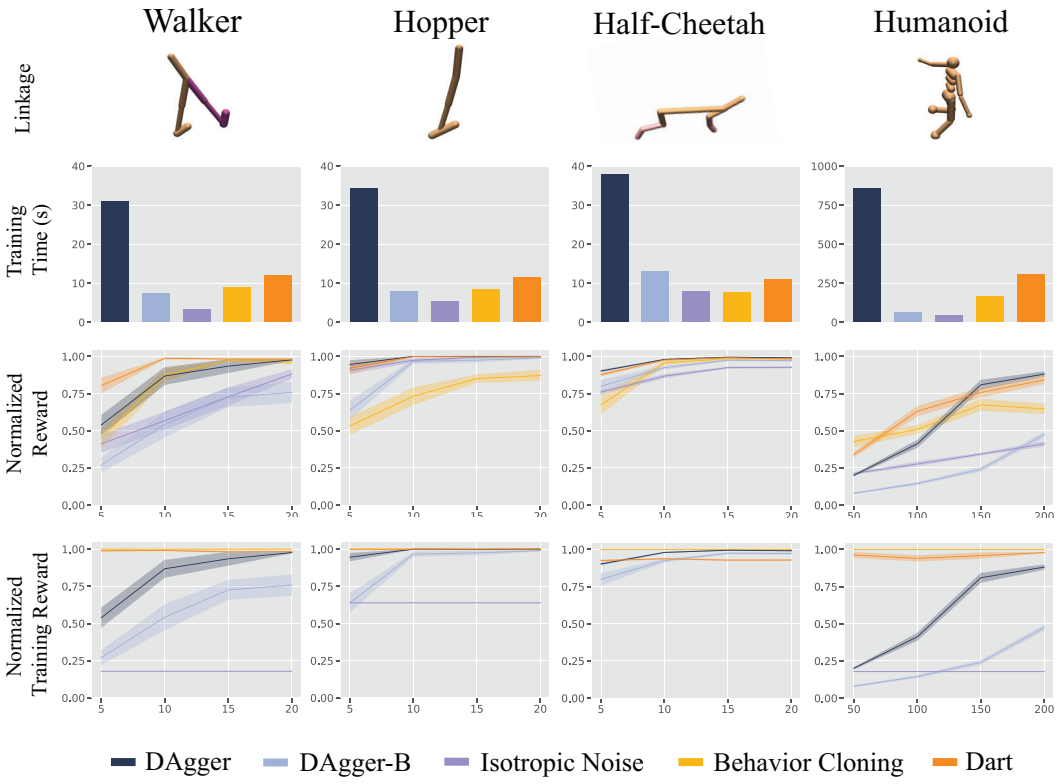

Figure 2: Top: The four different locomotive domains in MuJoCo we evaluated Dart on: Walker, Hopper, Half-Cheetah and Humanoid. Top Middle: The time, in seconds, to achieve the performance level reported below. Dart achieves similar performance to DAggger in all 4 domains, but requires significantly less computation because it doesn't require retraining the current robot policy. 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.

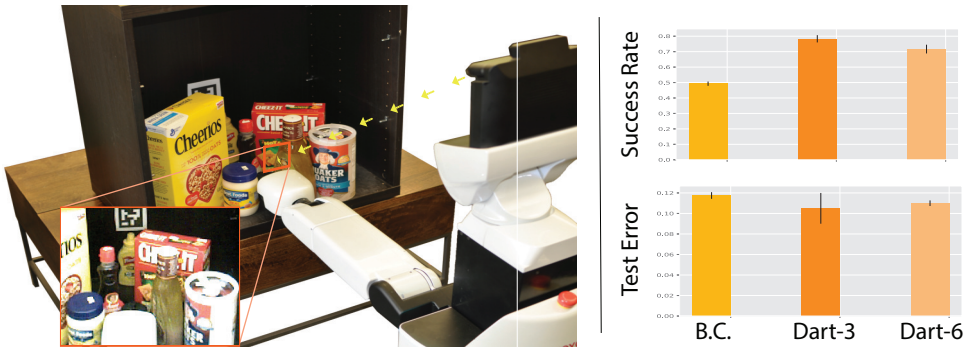

Figure 3: Left: Experimental setup for the grasping in clutter task. A Toyota HSR robot uses a head-mounted RGBD camera and its arm to push obstacle objects out of the way to reach the goal object, a mustard bottle. The robot's policy for pushing objects away uses a CNN trained on images taken from the robot's Primesense camera, an example image from the robot's view point is shown in the orange box. Right: the Success Rate (Top) and Test Error (Bottom) for Behavior Cloning, Dart($\alpha = 3$) and Dart($\alpha = 6$). Dart($\alpha = 3$) achieves the largest success rate and the lowest test error on held out demonstrations.

5.2 Robotic Grasping in Clutter

We evaluate Dart with human supervisors via a grasping in clutter task on a Toyota HSR robot. We specifically consider a task inspired by the Amazon Picking Challenge [3]. 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 test 4 different human supervisors, who have robotics experience but not specifically in the field of Imitation Learning. We compare three different techniques: Behavior Cloning, Dart with $\alpha = 3$ and Dart with $\alpha = 6$. We chose the α noise scaling parameters to correspond to a level that was comfortable for an author to perform the task($\alpha = 3$) and challenging($\alpha = 6$). 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. The final robot policy was then trained on the total of 40 demonstrations. Our experiment was within-subject, or every supervisor performed all three methods.

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, which is specified in the supplement material, an open-loop motion plan is generated to execute a grasp around the target object. We evaluate the policy 20 times on different sampled initial states.

In Fig. 3, we report the average performance of the three techniques. 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. We also report test error which is calculated via 10-fold cross validation on the dataset each policy is trained on. Empirically, we see a small improvement over B.C. with noise injection methods, which suggests it is easier to generalize to noise injected demonstrations. One reason for this maybe be that noise injection forces human supervisors to regularize their policies to be more robust.

6 Conclusion

In conclusion, we present an algorithm that minimizes covariate shift without the limitations of current on-policy methods. Our algorithm, Dart, provides the robot with corrective examples at the boundary of the supervisor's policy. By being concentrated around the superivsor's policy, it collects demonstrations without visiting highly-sub-optimal states and is easier for human supervisors tele-operating. We demonstrate it is at parity with on-policy methods in simulated MuJoCo domains and is significantly better than traditional Behavior Cloning on a real robot.

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] J.-F. Cardoso. Infomax and maximum likelihood for blind source separation. *IEEE Signal processing letters*, 4(4):112–114, 1997.

[3] 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.

[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] 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.

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

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

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

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

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

7 Appendix

7.1 Proofs for Theoretical Analysis

Lemma 4.1 *If* $\forall \mathbf{u}, \mathbf{u}\ 0 \leq l(\mathbf{u}, \mathbf{u}) \leq 1$*, 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:

$$|E_{p(\xi|\pi_{\theta^*},\psi)} J(\theta, \theta^*|\xi) - E_{p(\xi|\pi_\theta)} J(\theta, \theta^*|\xi)| \tag{6}$$
$$\leq T||p(\xi|\pi_{\theta^*}, \psi) - p(\xi|\pi_\theta)||_{TV} \tag{7}$$
$$\leq T\sqrt{\frac{1}{2}\mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*}, \psi)\big)} \tag{8}$$

The first line of the proof follos from Lemma 3.2, which is stated below and the fact $\forall \mathbf{u}, \mathbf{u}\ 0 \leq l(\mathbf{u}, \mathbf{u}) \leq 1$. 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:

$$\begin{aligned}
&|\int_x p(x)f(x) - \int_x q(x)f(x)| \\
&= |\int_x (p(x) - q(x))f(x)| \\
&= |\int_x (p(x) - q(x))|(f(x) - \frac{B}{2}) \\
&+ \frac{B}{2}(\int_x p(x) - q(x))| \\
&\leq \int_x |p(x) - q(x)||f(x) - \frac{B}{2}| \\
&\leq \frac{B}{2}\int_x |p(x) - q(x)| \\
&\leq B||P - Q||_{TV}
\end{aligned}$$

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 by deriving th definition of the KL -divergence:

$$\mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta),p(\xi|\pi_{\theta^*})\big) \tag{9}$$

$$= E_{p(\xi|\pi_\theta)} \log \frac{\prod_{t=0}^{T-1}\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\prod_{t=0}^{T-1}\pi_{\theta^*}(\mathbf{u}_t|\mathbf{x}_t)} \tag{10}$$

$$= E_{p(\xi|\pi_\theta)} \sum_{t=}^{T-1} \log \frac{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\pi_{\theta^*}(\mathbf{u}_t|\mathbf{x}_t)} \tag{11}$$

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 [8].

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

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 subesequently:

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

■

4.2 Derivation for Optimizing Noise

We derive the optimal noise sufficient statistics for the ϵ-greedy and Gaussian cases with respect to Eq. 3. For the ϵ-greedy case in the discrete domain with a finite number of controls K, the probability of choosing a control at a particular state is given by

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

Then the optimization problem in equation (3) becomes

$$\begin{aligned}
\epsilon^* &= \underset{\epsilon}{\text{argmin}}\ \mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta),p(\xi|\pi_{\theta^*},\epsilon)\big) \\
&= \underset{\epsilon}{\text{argmin}}\ E_{p(\xi|\pi_\theta)}\left[-\log p(\xi|\pi_{\theta^*},\epsilon)\right] \\
&= \underset{\epsilon}{\text{argmin}}\ E_{p(\xi|\pi_\theta)}\left[-\log\left(p(\mathbf{x}_0)\left(\frac{\epsilon}{K-1}\right)^{J(\theta,\theta^*|\xi)}(1-\epsilon)^{T-J(\theta,\theta^*|\xi)}\prod_{t=0}^{T-1}p(\mathbf{x}_{t+1}|\mathbf{u}_t,\mathbf{x}_t)\right)\right] \\
&= \underset{\epsilon}{\text{argmin}}\ E_{p(\xi|\pi_\theta)}\left[-\sum_{t=0}^{T-1}\log\left(\left(\frac{\epsilon}{K-1}\right)^{J(\theta,\theta^*|\xi)}\right)+\log\left((1-\epsilon)^{T-J(\theta,\theta^*|\xi)}\right)\right].
\end{aligned}$$

The third equality follows from the fact that along any given trajectory ξ from π_θ, the probability of ξ under π_{θ^*} is dependent on $J(\theta,\theta^*|\xi)$ which is the number of times π_θ 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

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

Therefore we have

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

Similarly, the covariance matrix for the Gaussian case in continuous domains can be derived. Consider a stochastic supervisor policy with distribution $\pi_{\theta^*}(\mathbf{u}_t|\mathbf{x}_t, \Sigma)$ defined by $\mathcal{N}(\pi_{\theta^*}(\mathbf{x}_t), \Sigma)$. Then the optimization problem is

$$\begin{aligned}\Sigma^* &= \underset{\Sigma}{\operatorname{argmin}}\, \mathcal{D}_{\mathcal{KL}}\big(p(\xi|\pi_\theta), p(\xi|\pi_{\theta^*}, \Sigma)\big) \\ &= \underset{\Sigma}{\operatorname{argmin}}\, -\frac{T}{2}\log\det\Sigma^{-1} - E_{p(\xi|\pi_\theta)} \sum_{t=0}^{T-1} \frac{1}{2} \left(\pi_\theta(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)\right)^T \Sigma^{-1} \left(\pi_\theta(\mathbf{x}_t) - \pi_{\theta^*}(\mathbf{x}_t)\right).\end{aligned}$$

As before, we take the derivative with respect to Σ and set it equal to zero:

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

Then the covariance matrix is

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

4.3 Additional Information for MuJoCo Experiments

To test how well Dart performs against on-policy methods such as DAgger and off-policy methods like Behavior Cloning, we use MuJoCo locomotion environments [20]. The challenge for these environments is 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 agent in a forward direction without falling.

We use an algorithmic supervisor in these domains: a policy whic is trained with TRPO [15] and is represented as a neural network with a two hidden layer of size 64. For all 4 domains, we used the same neural network as the supervisor and trained the 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 policies 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.

We used the following MuJoCo environments: Walker, Hopper, Humanoid, and Half-Cheetah. The size of the state space and control space for each task is $\{|\mathbf{x}| = 17, |\mathbf{u}| = 6\}$, $\{|\mathbf{x}| = 11, |\mathbf{u}| = 3\}$, $\{|\mathbf{x}| = 376, |\mathbf{u}| = 17\}$, $\{|\mathbf{x}| = 117, |\mathbf{u}| = 6\}$, 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.

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 $\alpha = 1$. 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 Information on Robot Setup

We evaluate DART with humans supervisors via a grasping in clutter task on a Toyota HSR robot. We specifically consider a task inspired by the Amazon Picking Challenge [3]. 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.

We collect demonstrations from the human supervisor via tele-operation with an Xbox controller. The robot's motion is constrained to its mobile base, which has three degrees of freedom translation and rotation, or x, y and θ. During tele-operation the supervisor sends change in position commands to these three degrees of freedom. RGB-D data is recorded, during a demonstration, from a Primesense structured light sensor, which is mounted at its head.

We test 4 different human supervisors, who have robotics experience but not specifically in the field of Imitation Learning. We compare three different techniques: Behavior Cloning, Dart with $\alpha = 3$ and Dart with $\alpha = 6$. We chose the α noise scaling parameters to correspond to a level that was comfortable for an author to perform the task($\alpha = 3$) and challenging($\alpha = 6$). 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. The final robot policy was then trained on the total of 40 demonstrations. Our experiment was within-subject, or every supervisor performed all three methods.

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 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 [11]. 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.