AC-Teach: A Bayesian Actor-Critic Method for Policy Learning with an Ensemble of Suboptimal Teachers

Andrey Kurenkov[†*], **Ajay Mandlekar**[†*], **Roberto Martin-Martin**[†],
Silvio Savarese[†], **Animesh Garg**[‡]
[†]Stanford University, [‡]University of Toronto, Vector Institute, Nvidia

Abstract: The exploration mechanism used by a Deep Reinforcement Learning (RL) agent plays a key role in determining its sample efficiency. Thus, improving over random exploration is crucial to solve long-horizon tasks with sparse rewards. We propose to leverage an ensemble of partial solutions as *teachers* that guide the agent's exploration with action suggestions throughout training. While the setup of learning with teachers has been previously studied, our proposed approach – Actor-Critic with Teacher Ensembles (AC-Teach) – is the first to work with an ensemble of suboptimal teachers that may solve only part of the problem or contradict other each other, forming a unified algorithmic solution that is compatible with a broad range of teacher ensembles. AC-Teach leverages a probabilistic representation of the expected outcome of the teachers' and student's actions to direct exploration, reduce dithering, and adapt to the dynamically changing quality of the learner. We evaluate a variant of AC-Teach that guides the learning of a Bayesian DDPG agent on three tasks – path following, robotic pick and place, and robotic cube sweeping using a hook – and show that it improves largely on sampling efficiency over a set of baselines, both for our target scenario of unconstrained suboptimal teachers and for easier setups with optimal or single teachers. Additional results and videos at `https://sites.google.com/view/acteach/`.

1 Introduction

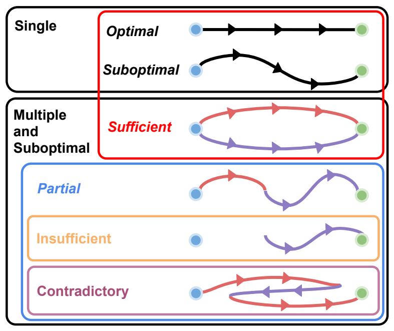
Reinforcement Learning (RL) algorithms have recently demonstrated impressive results in challenging problem domains such as robotic manipulation [1, 2], planning [3], Go [4], and Atari games [5]. However, RL algorithms typically require a large number of interactions with the environment to train policies that solve new tasks, which is particularly problematic for physical domains such as robotics, where gathering experience from interactions is slow and expensive. A possible approach to alleviate this problem is to train the agent to reproduce or bootstrap from the behavior demonstrated by an expert using either offline or online Imitation Learning (IL) [6–11]. However, IL has its own limitations: applying offline IL for long horizon tasks is costly and time consuming because large amounts of expert interactions are necessary to cover corner-cases the agent may encounter. Online IL may overcome this limitation by allowing the agent to query experiences on-demand, but requiring an online expert that can solve the entire long-horizon task may be an even stricter limitation.

We argue that, in domains such as robotics, an alternative to collecting demonstration data or providing full solutions is to encode knowledge into an *ensemble of heuristic solutions* (controllers, planners, previously trained policies, etc.) that address parts of the task. Leveraged properly, these heuristics act as *teachers* guiding agent exploration, leading to faster convergence during training and better asymptotic performance. Notably, the agent can also learn to outperform its teachers and learn parts of a task for which no teacher offers adequate advice. While prior work has addressed the problem of policy learning with teachers [12–16], this work discusses a collection of teacher attributes that characterize a single teacher's behavior compared to the optimal policy, as well as the aggregate behavior of the teachers in the ensemble, and proposes a unified algorithmic framework that efficiently leverages the set of teachers during the agent's training regardless of the teachers' attributes.

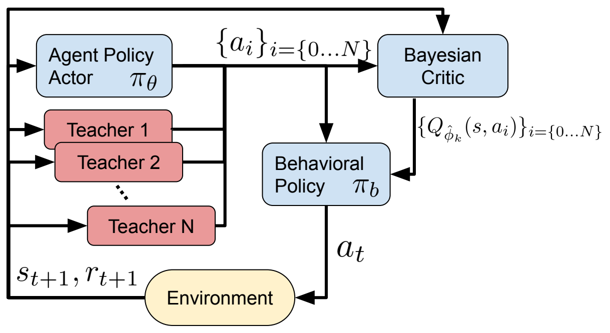
Specifically, we assume that the set of teachers can have the following attributes: (i) *partial* – teachers may recommend useful actions only in a part of the state space, (ii) *insufficient* – the agent cannot solve the task by only using actions suggested by the set of teachers, and (iii) *contradictory* – performing actions from different teachers in succession would hinder task completion (see Fig. 1).

[*]authors contributed equally

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.



(a)



Agent Policy Actor π_θ
$\{a_i\}_{i=\{0...N\}}$
Bayesian Critic
$\{Q_{\hat{\phi}_k}(s, a_i)\}_{i=\{0...N\}}$
Teacher 1
Teacher 2
Teacher N
Behavioral Policy π_b
a_t
s_{t+1}, r_{t+1}
Environment

(b)

Figure 1: (a) Visual representation of teachers' attributes, with arrows representing actions and line color representing different teacher policies. In this figure each example trajectory has the attributes of all the boxes it is contained within. Italicized terms apply to single policies, and non-italicized terms refer to sets of policies. We present formal definitions of these attributes in Sec. 3. (b) Our method, AC-Teach, (light blue) modifies the actor-critic architecture to leverage advice from a set of teachers (red) as part of the Behavioral Policy to train the Agent Policy using the probabilistic q-values of a Bayesian critic.

Policy learning in this setting is not straightforward - the agent needs to collect experience by deciding between its own *on-policy* actions or exploiting teacher advice, but also be able to leverage this heterogeneously generated experience to learn to solve the task. Learning from experience collected by another agent advocates using off-policy reinforcement learning [17], where a *behavioral policy* is used to gather the training experience. The main challenge in the teacher ensemble setting is that the utility of each teacher can vary in different state space regions, while the utility of the agent varies over time as learning progresses. This adds to the known complications of training an agent from off-policy experience, which may be unstable and result in poor performance – especially for Deep RL algorithms – due to the compounding effects of function approximation and bootstrapping [18–20].

To address these challenges, we present **Actor-Critic with Teacher Ensembles (AC-Teach)**, a policy learning framework to leverage advice from multiple teachers where individual teachers might only offer useful advice in certain states, offer advice that contradicts other teacher suggestions, and where the agent might need to learn behaviors from scratch in states where no teacher offers useful advice. AC-Teach is a novel behavioral policy mechanism for continuous control domains that is robust to low-quality teacher ensembles through quantifying uncertainty over the value of actions based on a probabilistic posterior critic network [21] and using the uncertainty to choose between different action proposals. It is agnostic to the source of action proposals, since it models a single action-value function (the critic) that is not explicitly conditioned on any teacher, and can therefore easily scale to settings with larger number of teachers and even to settings with changing teacher sets. By contrast, most prior work models each teacher independently in some form.

AC-Teach also includes a *commitment* mechanism that allows the behavioral policy to commit to actions from a single policy for longer periods of time; the posterior critic is used to estimate the probability of value improvement from switching the policy choice, and a switch is only enacted under high confidence. This allows the behavioral policy to collect meaningful experience from partial teachers without being hindered by contradictory teachers. Lastly, we also mitigate the instability of off-policy learning by basing the behavioral policy on the critic, and training the critic network using a target value that is generated by the behavioral policy. This helps reduce the off-policyness of the experience being used to train the critic by coupling the critic and the behavioral policy together.

Summary of Contributions. (a) we present a collection of teacher attributes that comprehensively characterizes the quality of a set of teachers for guiding agent training; (b) we propose Actor-Critic with Teacher Ensembles (AC-Teach), a policy learning framework to leverage advice from multiple teachers that is robust to low-quality teacher sets where individual teachers might only offer useful advice in certain states, offer advice that contradicts other teacher suggestions, and where the agent might need to learn behaviors from scratch in states where no teacher offers useful advice; (c) Our experiments demonstrate that AC-Teach is able to leverage such teacher ensembles to solve multi-step tasks while significantly improving sample efficiency over baselines; and (d) we also show that AC-Teach is not only able to generate policies using low-quality teacher sets but also surpasses baselines when using higher quality teachers, hence providing a unified algorithmic solution to a broad range of teacher attributes.

2 Related Work

Imitation and Reinforcement Learning from Offline Demonstrations: Imitation Learning has been used to train a policy from a set of demonstrations, which are offline samples collected from an expert that is assumed to be optimal. Recently, several works [22, 11, 23–26] have applied off-policy deep RL algorithms to train an agent from a set of suboptimal demonstrations. However, off-policy deep RL can be particularly unstable due to the compounding effects of function approximation and bootstrapping leading to inaccurate extrapolation [27, 18–20]. In our setting, the behavioral policy is able to follow any teacher in the policy set. Consequently, training the agent on this off-policy data requires addressing the issue of instability. **Imitation and Reinforcement Learning from Online Teachers:** Prior work has also investigated the use of one or more teacher policies for training an agent. Several works [6, 27, 28] focus on the setting where an agent must learn to imitate a teacher, who is available as an oracle during training. These methods assume that the expert is optimal and can solve the entire task. By contrast, [29–31] assume access to a stable controller or prior solution and use reinforcement learning to learn a residual to correct for suboptimal behavior. While these approaches relax the need for a teacher that is optimal, they cannot be applied to settings with more than one teacher.

Several prior works [12–15] present methods compatible with several teachers, but they assume that one teacher should be selected over the others for the entire episode and formulate the problem as regret minimization. This is unsuitable when considering teachers that might offer only partial solutions to the task, since the final solution might require knowledge from several teachers that excel in different parts of the state space. Li et al. [16] relaxes this assumption, but does so by treating the teachers as options [32], which the agent must rely on at test-time as well as train-time, and furthermore, their method does not support continuous control. The method presented in Xie et al. [33] trains a DQN [5] behavioral policy to select between an expert and a DDPG [34] learner policy. This method can be extended to settings with multiple experts, although this was not part of the original manuscript. We developed a DQN-based behavioral policy for multiple agents as a baseline in our experiments and show that AC-Teach outperforms this method.

Table 1: AC-Teach can be used in RL settings with both function approximation, and continuous control, and is able to leverage a wide variety of teacher sets. The symbol $\odot$ indicates that the algorithm could potentially be applied to this setting, but this has not been demonstrated empirically.

	Deep RL	Continuous control	No teachers at test time	Multiple teachers	Partial teachers	Contradictory Teachers
PRQL/OpsTL [12, 15]	$\odot$		✓		✓	$\odot$
DQfD/DDPGfD [22, 11]	✓	✓	✓	$\odot$		$\odot$
CAPS [16]				✓	✓	$\odot$
Residuals [30, 31]	✓	✓				$\odot$
Wheels [33]	✓	✓	✓	$\odot$	$\odot$	$\odot$
AC-Teach (ours)	✓	✓	✓	✓	✓	✓

3 Problem Statement and Notation

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, \mathcal{T}, \gamma, \rho_0)$ denote a discrete-time Markov Decision Process (MDP), where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R(s, a)$ is the reward function, $\mathcal{T}(s'|s, a)$ defines the transition dynamics, $\gamma \in [0, 1)$ is the discount factor, and ρ_0 is the start state distribution from which a start state $s_0 \sim \rho_0(\cdot)$ is sampled in every episode. At every step, the agent observes the state s_t, chooses an action $a_t \sim \pi_\theta(\cdot|s_t)$ sampled from the policy π parametrized by θ, and observes $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)$ and a reward $r_t = R(s_t, a_t)$. The goal in reinforcement learning is to learn an *agent* policy π_θ that maximizes the expected discounted sum of rewards $\mathbb{E}_{\mathbf{T}}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$ from experience acquired during exploration, where $\mathbf{T} = (s_0, a_0, s_1, \dots, a_{T-1}, s_T)$ is the trajectory produced by the policy.

In off-policy RL algorithms we consider two independent policies: (1) *a behavioral policy*, π_b, that decides on actions to collect experience, and 2) *an agent policy*, π_θ, that consumes the experience and is trained to accomplish the original task. Both policies are part of the training process but only the agent policy, π_θ, is evaluated at test time. We note that since we seek to train an agent using experience generated by different teacher policies, our algorithm must necessarily be off-policy.

Policy Learning with Teacher Sets: We consider the policy advice setting in which the behavioral policy π_b can receive action suggestions from a set of teacher policies $\Pi = \{\pi_1, \pi_2, \dots, \pi_N\}$ that are available during training but not at test time [13]. The problem is then to specify a behavioral policy π_b that efficiently leverages the advice of a set Π of teacher policies to generate experience to train an agent policy π_θ with the goal of achieving good test-time performance in minimal train-time environment interactions.

Teacher Attributes: To formalize the teacher attributes let us first consider an infinite-horizon MDPs with *deterministic* transition function $s' = \mathcal{T}(s,a)$, a set of absorbing goal states $\mathcal{G} \subset \mathcal{S}$, and a sparse reward function $r_\mathcal{G}(s) = \mathbb{1}(s \in \mathcal{G})$. A generalization of the teacher attributes to MDPs with stochastic dynamics and policies can be found in Appendix A.2. For a given deterministic policy π and a goal set $\mathcal{G}$, the state-value function at the initial state is $V^\pi_\mathcal{G}(s_0) = r_\mathcal{G}(s_0) + \sum_{t=1}^{\infty} \gamma^t r_\mathcal{G}(s_t)$. For this class of MDPs, the state-value function of deterministic policy π has a higher value at the state s than at another state s' if and only if π can reach a goal in $\mathcal{G}$ in fewer timesteps from s than from s' (see Proposition 1 and associated proof in Appendix A.2). In this context, we define three teacher attributes: *partial*, *sufficient*, and *contradictory*.

Definition 3.1 (Partial Teachers) Let π^* denote the optimal policy and $V^*_\mathcal{G}(s)$ denote the optimal state value function, with respect to a fixed set of goals $\mathcal{G}$. Then, a teacher policy is partial if in some non-empty strict subset $\exists S' \subset \mathcal{S}$, for all states $s \in S'$, we have that $V^*_\mathcal{G}(\mathcal{T}(s,a)) > V^*_\mathcal{G}(s)$.

By Proposition 1, it follows that in some region of the state space, a partial teacher transitions from one state to another state such that an optimal policy would reach the goal faster from the new state. Thus, in that region, the partial teacher offers useful advice.

Definition 3.2 (Sufficient Teachers and Teacher Sets) A teacher policy π is sufficient if it has non-zero value for some start state: $\exists s_0 \sim \rho_0(\cdot)$ such that $V^\pi_\mathcal{G}(s_0) > 0$. A teacher set $\Pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ is sufficient if there exists a mixture policy that is sufficient (a mixture policy is one that chooses $\pi_\Pi(s) \in \{\pi_1(s), \pi_2(s), \ldots, \pi_N(s)\}$).

In other words, by Proposition 1, a teacher is sufficient if it can reach a goal state from some start state and a teacher set is sufficient if some combination of teachers in the set can be used to reach a goal state from some start state. Conversely, a teacher or teacher set that is not sufficient is said to be *insufficient*. The ability to learn in settings with insufficient teachers is an important property for our method, since it requires learning parts of a task that teachers are unable to solve.

Definition 3.3 (Contradictory Teachers) A teacher policy π_2 is contradictory to teacher policy π_1 and a goal set $\mathcal{G}$ if there exists a state $s \in \mathcal{S}$ where following the advice of π_2 causes π_1 to take more timesteps to reach a goal in the goal set. Equivalently, following π_2 in s leads to a state with lower value for π_1: $V^{\pi_1}_\mathcal{G}(\mathcal{T}(s, \pi_2(s)) < V^{\pi_1}_\mathcal{G}(s)$.

In other words, the policy π_2 hinders the progress of π_1 in some portion of the state space with respect to the set of goals $\mathcal{G}$. As previously established, our goal is to define a behavioral policy π_b that benefits from teacher sets that may or may not be sufficient or contain contradictory teachers.

There are multiple methods to maximize the RL objective based on the policy gradient theorem [17]. One family of solutions is *actor-critic* methods [34–38]. Herein, we build AC-Teach with Bayesian DDPG [34, 21], a popular actor-critic algorithm for continuous action spaces. The agent policy π_θ in AC-Teach is the actor in the DDPG architecture. Conceptually, AC-Teach can leverage any off-policy actor-critic algorithm, such as, DDPG [34], SAC [37], or TD3 [38], among others.

DDPG. DDPG maintains a critic network $Q_\phi(s,a)$ and a deterministic actor network $\pi_\theta(s)$ (the agent policy), parametrized by ϕ and θ respectively. A behavioral policy π_b (usually the same as the agent policy, π_θ, with an additional exploration noise) is used to select actions that are executed in the environment, and state transitions are stored in a replay buffer $\mathcal{B}$. DDPG alternates between collecting experience and sampling the buffer to train the policy π_θ and the critic Q_ϕ. The critic is trained via the Bellman residual loss $\mathcal{L}_{\text{critic}} = (r + \gamma Q_{\phi'}(s', \pi_{\theta'}(s')) - Q_\phi(s,a))^2$ and the actor is trained with a deterministic policy gradient update to choose actions that maximize the critic $\mathcal{L}_{\text{actor}} = -Q_\phi(s, \pi_\theta(s))$ where ϕ' and θ' denote the use of target critic and actor networks.

The stability and performance of DDPG varies strongly between tasks. To alleviate these problems, Henderson *et al.* [21] introduced Bayesian DDPG, a Bayesian Policy Gradient method that extends DDPG by estimating a posterior value function for the critic. The posterior is obtained based on Bayesian dropout [39] with an α-divergence loss. AC-Teach trains a Bayesian critic and actor in a similar fashion - see Appendix B.1 for details.

4 Actor-Critic with Teacher Ensembles (AC-Teach)

In this section we introduce AC-Teach, an approach to leverage policy advice from a set of unconstrained teachers Π in actor-critic RL algorithms as illustrated in Fig. 1(b) and Algorithm 1. AC-Teach addresses four key challenges with regards to how to implement an efficient behavioral policy.

1. How to *evaluate the quality of the advice* from any given teacher in each state for a continuous state and action space? AC-Teach is based on a novel *critic-guided behavioral policy* that evaluates both the advice from the teachers as well as the actions of the learner.
2. How to *balance exploitation and exploration* in the behavioral policy? AC-Teach uses *Thompson sampling* on the posterior over expected action returns provided by a Bayesian critic to help the behavioral policy to select which advice to follow.
3. How to deal with *contradictory teachers*? AC-Teach implements a temporal *commitment* method based on the posterior from the Bayesian critic that executes actions from the same policy until the confidence in return improvement from switching to another policy is significant.
4. How to *alleviate extrapolation errors in the agent* arising from large differences between the behavioral and the agent policy, the "large off-policy-ness" problem? AC-Teach introduces a *behavioral target* into DDPG's policy gradient update, such that the critic is optimized with the target Q-value of the behavioral policy rather than the agent policy.

1. Critic-Guided Behavioral Policy. As introduced in Sec. 3, the behavioral policy, π_b collects experience in the environment during training and is typically the output of the actor network π_θ with added noise. However, when teachers are available, the behavioral policy should take their advice into consideration. To leverage teacher advice for exploration, we propose to use the critic to implement π_b in AC-Teach as follows. Given state s, the agent policy π_θ and each teacher policy $\pi_i \in \Pi$ generate a set of action proposals $\{\pi_\theta(s), \pi_1(s), \ldots, \pi_N(s)\}$. The critic Q_ϕ evaluates the set of action proposals and selects the most promising one to execute in the environment. This is equivalent to selecting between the teachers and the agent, but notice that this selection mechanism is agnostic to the source of the actions, enabling AC-Teach to scale to large teacher sets.

2. Thompson Sampling over a Bayesian Critic for Behavioral Policy. The behavioral policy needs to balance between exploration of teachers, whose utility in different states is not known at the start of training, and the agent policy, whose utility is non-stationary during learning versus exploitation of teachers that provided highly rewarded advice in the past. Inspired by the similarity between policy selection and the multi-arm bandits problem, we use Thompson sampling, a well-known approach for efficiently balancing exploration and exploitation in the bandit setting [40]. Thompson sampling is a Bayesian approach for decision making where the uncertainty of each decision is modeled by a posterior reward distribution for each arm. In our multi-step setting, we model the posterior distribution over action-values using a Bayesian dropout critic network similar to Henderson et al. [21].

Concretely, instead of maintaining a point estimate of ϕ and Q_ϕ, we maintain a distribution over weights, and consequently over values by using Bayesian dropout (Sec. 3). To evaluate an action for a given state-action pair, a new dropout mask is sampled at each layer of Q_ϕ, resulting in a set of weights $\hat{\phi}$, and then a forward pass through the network results in a sample $Q_{\hat{\phi}}(s, a)$. We then use critic $Q_{\hat{\phi}}$ to evaluate the set of action proposals $\{a_0, a_1, \ldots, a_N\}$ and selects $a_i = \arg\max_{a_0, a_1, \ldots, a_N} Q_{\hat{\phi}}(s, a)$ as the action to consider executing. The choice whether to execute this action depends on our commitment mechanism, explained next.

3. Confidence Based Commitment. In our problem setup, we consider the possibility of contradictory advice from different teachers that hinders task progress. Therefore, it is crucial to avoid switching excessively between teachers and *commit* to the advice from the same teacher for longer time periods. This is particularly important at the beginning of the training process when the critic has not yet learned to provide correct evaluations.

To achieve a longer time commitment, we compare the policy selected at this timestep π_i via the Thompson Sampling process to the policy selected at the previous timestep, π_j. We use the posterior critic to estimate the probability for the value of a_i to be larger than the value of a_j (see Appendix B.2). If the probability of value improvement is larger than a threshold $\beta\psi^{t_c}$, the behavioral policy acts using the new policy, otherwise it acts with the previous policy. The threshold β controls the behavioral policy's aversion to switch, and ψ controls the degree of multiplicative decay, to prevent

Algorithm 1 Actor-Critic with Teacher Ensembles (AC-Teach): Behavioral Policy

```
Require: $s_t, Q_\phi, \pi_\theta, \Pi = \{\pi_1, \dots \pi_N\}$   ▷ Current observation, Bayesian critic, Actor, Teacher set
 1: $A \leftarrow [\pi_\theta(s_t), \pi_1(s_t), \dots \pi_N(s_t)]$   ▷ Collect action proposals from actor and teachers
 2: $\hat{\phi} \sim q(\phi)$   ▷ Posterior sampling of critic weights via Bayesian Dropout
 3: $c_t \leftarrow \arg\max_{i \in \{[0,1,\dots,N\}} Q_{\hat{\phi}}(s_t, A_i)$   ▷ Proposed policy choice via Thompson Sampling
 4: $p_{better} \leftarrow P[Q_\phi(s_t, A_{c_t}) > Q_\phi(s_t, A_{c_{t-1}})]$   ▷ Probability of value improvement (Appendix B.2)
 5: if $p_{better} < \beta\psi^{t_c}$ then
 6:     $c_t \leftarrow c_{t-1}$   ▷ Retain previous policy choice for low improvement probability
 7:     $t_c \leftarrow t_c + 1$
 8: else   $t_c \leftarrow 0$   ▷ Accept proposed policy choice for high improvement probability
 9: end if
10: return $A_{c_t}$   ▷ Return action based on policy choice
```

over-commitment and make it easier to switch the policy choice when a policy is selected for several consecutive time steps, as in prior work [12, 15].

4. Addressing Extrapolation Error in the Critic. Off-policy learning can be unstable for deep reinforcement learning approaches [18–20]. We follow Henderson et al. [21] in training the learner policy π_θ and the Bayesian critic Q_ϕ on samples from the experience replay $\mathcal{B}$. However, we further improved the stability of training by modifying the critic target values used for the α-divergence Bayesian critic loss. Instead of using $r + \gamma Q_{\phi'}(s', \pi_{\theta'}(s'))$ as the target value for the critic, we opt to use $r + \gamma Q_{\phi'}(s', \pi_b(s'))$.

In other words, the behavioral policy is used to select target values for the critic. We observed that using this modified target value in conjunction with basing the behavioral policy on the critic improved off-policy learning (see Appendix B.3). This, along with the behavioral policy summarized in Algorithm 1, forms our method, AC-Teach.

5 Experimental Setup

We designed AC-Teach to be able to leverage experience from challenging sets of teachers that do not always provide good advice (see Table 1). In our experiments, we compare AC-Teach to other learning algorithms that use experience from teachers with the aforementioned attributes (see Sec. 3) for the following three control tasks (see Appendix D.1 for more task details). We outline the tasks and teacher sets below. For each task, we design a *sufficient* teacher that can complete each task that chooses the appropriate *partial* teacher to query per state, and unless differently specified, we add a Gaussian action perturbation to every teacher's actions during learning so that their behavior is more suboptimal.

1. `Path Following`**:** A point agent starts each episode at the origin of a 2D plane and needs to visit the four corners of a square centered at the origin. These corners must be visited in a specific order that is randomly sampled at each episode. The agent applies delta position actions to move.

Teacher Set: We designed one teacher per corner that, when queried, moves the agent a step of maximum length closer to that corner. Each of these teachers is *partial* since it can only solve part of the task (converging to the specific corner). The four teachers are needed for the teacher set to be *sufficient*. We additionally designed a set of *contradictory* teachers, which we explain in full detail in Appendix C.1.

2. `Pick and Place`**:** A robot manipulation problem from [41]. The objective is to pick up a cube and place it at a target location on the table surface. The initial position of the object and the robot end effector are randomized at each episode start, but the goal is constant. The agent applies delta position commands to the end effector and can actuate the gripper.

Teacher Set: We designed two partial teachers for this task, *pick* and *place*. The pick teacher moves directly toward the object and grasps it when close enough. The place agent is implemented to move the grasped cube in a parabolic motion towards the goal location and dropping it on the target location once it is overhead.

3. `Hook Sweep`**:** A robot manipulation problem adapted from [31]. The objective is to actuate a robot arm to move a cube to a particular goal location. The cube is initialized out of reach of the robot arm, and so the robot must use a hook to sweep the cube to the goal. The goal location and

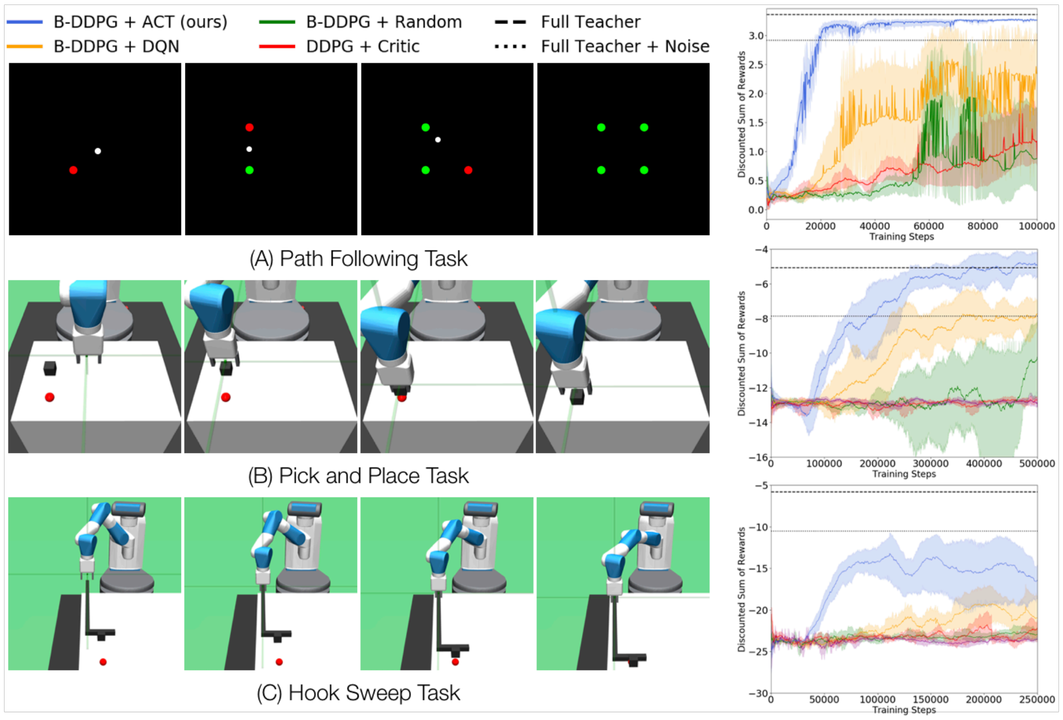


Figure 2: **Tasks and Performance with *Sufficient Partial* Teacher Set.** The `Path Following` (top), `Pick and Place` (middle), and `Hook Sweep` (bottom) tasks are shown above, during one task completion. The plots (right), showing the average evaluation return of the agent during training, demonstrate that our algorithm exhibits faster and better convergence than other baselines with a *sufficient* teacher set of noisy *partial* solutions.

initial cube location are randomized such that in some episodes the robot arm must use the hook to sweep the cube closer to its base and in other episodes the robot arm must use the hook to push the cube away from its base to a location far from the robot.

Teacher Set: We designed four partial teachers for this task, *hook-pick*, *hook-position*, *sweep-in*, and *sweep-out*. The *hook-pick* teacher guides the end-effector to the base of the hook and grasps the hook. The *hook-position* teacher assumes that the hook has been grasped at the handle and attempts to move the end effector into a position where the hook would be in a good location to sweep the cube to the goal. Note that this teacher is agnostic to whether the hook has actually been grasped and tries to position the arm regardless. The *sweep-in* and *sweep-out* teachers move the end effector toward or away from the robot base respectively such that the hook would sweep the cube into the goal, if the robot were holding the hook and the hook had been positioned correctly, relative to the cube.

Baselines. We compare AC-Teach against the following set of baselines:

1. **BDDPG:** Vanilla DDPG without teachers, using a Bayesian critic as in [21].
2. **DDPG + Teachers (Critic):** Train a point estimate of the critic parameters instead of using Bayesian dropout. The behavioral policy still uses the critic to choose a policy to run.
3. **BDDPG + Teachers (Random):** BDDPG with a behavioral policy that picks an agent to run uniformly at random.
4. **BDDPG + Teachers (DQN):** BDDPG with a behavioral policy that is a Deep Q Network (DQN), trained alongside the agent to select the source policy as in [33].

6 Results

Our experiments answer the following questions: (1) to what extent does AC-Teach improve the number of interactions needed by the agent to learn to solve the task by leveraging a set of teachers that are *partial*? (2) can AC-Teach still improve sample efficiency when the set of teachers is *insufficient* and parts of the task must be learned from scratch? (3) how sensitive is the performance of AC-Teach to the quality and size of the teacher set?

Fig. 3 addresses (1) by showing that AC-Teach outperforms all baselines in improving both the sample efficiency of learning and the asymptotic performance of the agent. We present our results on the `Pick-and-Place` task and defer results on the `Path Following` and `Hook Sweep` tasks in

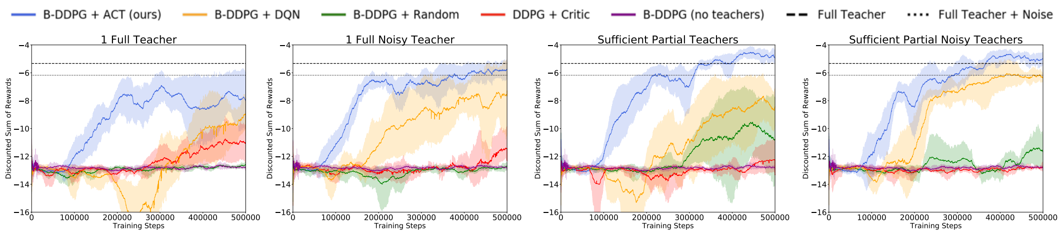


Figure 3: **Evaluation with *sufficient* teacher sets:** Test time performance of an agent trained for the `pick-and-place` task with AC-Teach and baselines. Whether the teacher set contains just one *sufficient* teacher (leftmost two graphs) or multiple *partial* teachers (rightmost two graphs), our method significantly outperforms all others in both convergence speed and asymptotic performance. With multiple *partial* teachers, the AC-Teach agent even outperforms our hand-coded teacher without noise, despite it being very close to optimal.

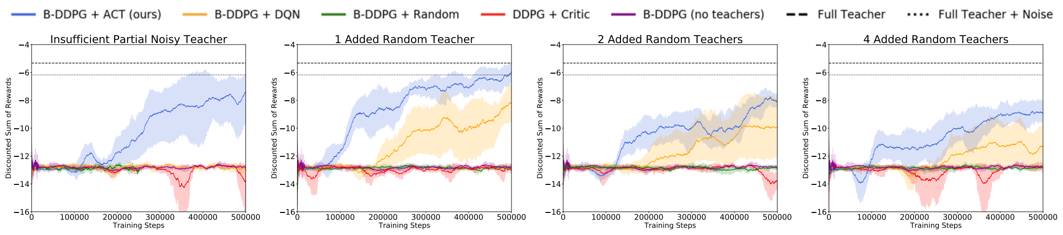


Figure 4: **Evaluation with *insufficient* and low-quality teacher sets.** Test time agent performance on the `pick-and-place` task having trained with AC-Teach and other baselines. Both when utilizing just the suboptimal pick teacher (left graphs) or when utilizing both suboptimal pick and place teachers along with added non-helpful teachers which always suggest random actions (right three graphs), our method outperforms all others in both convergence speed and asymptotic performance.

Appendix C.1 and Appendix C.2, along with ablation analysis (Appendix C.3) and experiments on parameter sensitivity (Appendix C.4). Ablation studies suggest that all 4 components of our approach are needed for it to work well across tasks.

We evaluate AC-Teach with *sufficient* and *partial* teachers by varying whether the teacher set contains a single *sufficient* teacher or both the pick and place *partial* teachers, and whether these teachers are made worse through the addition of noise. As shown in Fig. 4, our method consistently performs better than all baselines across all teacher sets. Interestingly, both introducing suboptimality into the teacher(s) and using partial teachers, instead of a single full teacher, improve performance for our algorithm as well as the next best performing baseline. We believe this is due to suboptimal teachers providing more varied advice and increasing the diversity of data in the replay buffer, which mitigates extrapolation error.

Next, we evaluate AC-Teach with an *insufficient* teacher set by providing only the noisy pick teacher and thus requiring the agent to learn the place part of the task on its own. As shown in the *insufficient partial noisy teacher* plot in Figure 4, none of the baselines are able to learn in this condition, whereas our method is able to approach the performance of our noisy full teacher.

Lastly, we demonstrate AC-Teach's robustness to learning with miss-specified teachers by training the agent with the two noisy pick and place teachers as well as 1, 2, and 4 teachers that always suggest random actions. As can be seen in Fig. 4, the agent's learning performance does degrade as we add more random teachers but AC-Teach consistently outperforms the other baselines and is able to get some benefit from the teacher set.

7 Conclusion

We presented AC-Teach, a unifying approach to leverage advice from an ensemble of sub-optimal teachers to accelerate the learning process of an off-policy RL agent. AC-Teach incorporates teachers' advice into the behavioral policy based on a Thomson sampling mechanism on the probabilistic evaluations of a Bayesian critic. We compare AC-Teach with prior approaches and show that it extracts useful exploratory experiences when the set of teachers is noisy, partial, incomplete, or even contradictory. In future, we plan to apply AC-Teach to real robot policy learning to demonstrate its applicability to solving challenging long-horizon manipulation in the real world.

References

[1] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[2] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[3] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5113–5120. IEEE, 2018.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

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

[6] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

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

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

[9] T. Zhang, Z. McCarthy, O. Jowl, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[10] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[11] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[12] F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727. ACM, 2006.

[13] M. G. Azar, A. Lazaric, and E. Brunskill. Regret bounds for reinforcement learning with policy advice. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 97–112. Springer, 2013.

[14] Y. Zhan, H. B. Ammar, et al. Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer. *arXiv preprint arXiv:1604.03986*, 2016.

[15] S. Li and C. Zhang. An optimal online method of selecting source policies for reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] S. Li, F. Gu, G. Zhu, and C. Zhang. Context-aware policy reuse. *arXiv preprint arXiv:1806.03793*, 2018.

[17] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[18] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[19] A. Bhatt, M. Argus, A. Amiranashvili, and T. Brox. Crossnorm: Normalization for off-policy td reinforcement learning. *arXiv preprint arXiv:1902.05605*, 2019.

[20] J. Achiam, E. Knight, and P. Abbeel. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

[21] P. Henderson, T. Doan, R. Islam, and D. Meger. Bayesian policy gradients via alpha divergence dropout inference. *arXiv preprint arXiv:1712.02037*, 2017.

[22] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[24] Y. Gao, J. Lin, F. Yu, S. Levine, T. Darrell, et al. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[25] Z. Wang and M. E. Taylor. Interactive reinforcement learning with dynamic reuse of prior knowledge from human/agent's demonstration. *arXiv preprint arXiv:1805.04493*, 2018.

[26] Z. Wang and M. E. Taylor. Improving reinforcement learning with confidence-based demonstrations. In *IJCAI*, pages 3027–3033, 2017.

[27] S. Ross and J. A. Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

[28] 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.

[29] M. T. Rosenstein and A. G. Barto. Reinforcement learning with supervision by a stable controller. In *Proceedings of the 2004 American Control Conference*, volume 5, pages 4517–4522. IEEE, 2004.

[30] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. *arXiv preprint arXiv:1812.03201*, 2018.

[31] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[32] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[33] L. Xie, S. Wang, S. Rosa, A. Markham, and N. Trigoni. Learning with training wheels: Speeding up training with a simple controller for deep reinforcement learning. Institute of Electrical and Electronics Engineers, 2018.

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

[35] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

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

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[38] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[39] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[40] E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

[41] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[42] A. Hill, A. Raffin, M. Ernestus, A. Gleave, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

Appendix

A Problem Formulation

A.1 Relationship between Value Function and Reaching Time

We consider the class of infinite-horizon MDPs (defined in Sec. 3) that have a deterministic transition function $s' = \mathcal{T}(s,a)$, a set of absorbing goal states $\mathcal{G} \subset \mathcal{S}$, and a sparse reward function $r_{\mathcal{G}}(s) = \mathbb{1}(s \in \mathcal{G})$. Note that for a given deterministic policy π and a goal set $\mathcal{G}$, we have

$$V^{\pi}_{\mathcal{G}}(s_0) = r_{\mathcal{G}}(s_0) + \sum_{t=1}^{\infty} \gamma^t r_{\mathcal{G}}(s_t) = \mathbb{1}(s_0 \in \mathcal{G}) + \sum_{t=1}^{\infty} \gamma^t \mathbb{1}(s_t \in \mathcal{G}).$$

Proposition 1. *In this class of MDPs, a deterministic policy π has a higher value for state s than another state s' if and only if π can reach a goal in $\mathcal{G}$ in fewer timesteps from s than from s'.*

Proof. Consider any goal state $s_g \in \mathcal{G}$. Since every goal state is absorbing with reward 1, we have $V^{\pi}(s_g) = \frac{1}{1-\gamma}$. Now consider any arbitrary state s. Since both the policy π and transition function $\mathcal{T}$ are deterministic, an agent acting according to π either reaches a goal state $s_g \in \mathcal{G}$ in a finite number of timesteps t_g or never reaches a goal state. Therefore, the value function $V^{\pi}(s) = \frac{\gamma^{t_g - 1}}{1-\gamma}$ if π can reach a goal state from s, and 0 otherwise. Now, assume π reaches a goal from state s in t_s timesteps and from s' in $t_{s'}$ timesteps. Then states s and s' have value $V^{\pi}(s) = \frac{\gamma^{t_s - 1}}{1-\gamma}$ and $V^{\pi}(s') = \frac{\gamma^{t_{s'} - 1}}{1-\gamma}$ respectively. Since the function $f(t) = \frac{\gamma^{t-1}}{1-\gamma}$ is monotone decreasing in t for $\gamma \in (0,1)$, we have that $t_s < t_{s'} \leftrightarrow V^{\pi}(s) > V^{\pi}(s')$, and the proposition holds. □

A.2 Extension to Stochastic MDPs

We now consider infinite-horizon MDPs with stochastic transition dynamics $\mathcal{T}(s'|s,a)$, a set of absorbing goal states $\mathcal{G} \subset \mathcal{S}$, and a sparse reward function $r_{\mathcal{G}}(s) = \mathbb{1}(s \in \mathcal{G})$.

A.2.1 Interpretation of value function

The value function of a stochastic policy π is

$$V^{\pi}_{\mathcal{G}}(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}(s_t \in \mathcal{G})\right].$$

Consider a trajectory collected by π, $\tau = (s_0, a_0, s_1, a_1, \dots)$ where $a_t \sim \pi(\cdot|s_t)$, $s_t \sim \mathcal{T}(\cdot|s_t, a_t)$. Every trajectory either reaches a goal state and stays there so that $\lim_{t\to\infty} s_t = s_g$ for some $s_g \in \mathcal{G}$, or never reaches a goal state. Then, all possible trajectories collected by π either belong to the set of goal-reaching trajectories T_g or the complementary set T_g^C. Every trajectory $\tau \in \mathrm{T}_g$ reaches a goal in some number of steps t_g and consequently generates return $\frac{\gamma^{t_g - 1}}{1-\gamma}$ and every trajectory $\tau \in \mathrm{T}_g^C$ generates 0 return since no state generates reward.

Now we write the value function as the expected discounted sum of rewards of trajectories $R(\tau)$

$$\begin{aligned} V^{\pi}_{\mathcal{G}}(s_0) &= \mathbb{E}_{\tau}[R(\tau)] \\ &= P(\tau \in \mathrm{T}_g)\, \mathbb{E}_{\tau}[R(\tau) \mid \tau \in \mathrm{T}_g] + P(\tau \in \mathrm{T}_g^C)\, \mathbb{E}_{\tau}[R(\tau) \mid \tau \in \mathrm{T}_g^C] \\ &= P(\tau \in \mathrm{T}_g)\, \mathbb{E}_{t_g}\left[\frac{\gamma^{t_g - 1}}{1-\gamma} \mid t_g < \infty\right] \end{aligned}$$

where the second line follows by the law of iterated expectations and the third line follows by rewriting the first term as an expectation over the reaching time of trajectories that reach a goal and using the fact that the second term vanishes since all returns of trajectories in the complementary set are 0.

Algorithm B.1 Actor-Critic with Teachers (AC-Teach) Training Loop

```
Require: $\pi_b$, $Q_\phi$, $\pi_\theta$, $\Pi$, $\mathcal{B}$ ▷ AC-Teach Behavioral Policy, Critic, Actor, Teacher Set, Replay Buffer
 1: for epoch $= 1, \ldots, n_{\text{epoch}}$ do
 2:     for episode $= 1, \ldots, n_{\text{episodes}}$ do
 3:         for $t = 1, \ldots, T$ do
 4:             $a_t \leftarrow \pi_b(s_t \mid Q_\phi, \pi_\theta, \Pi)$        ▷ Action from behavioral policy using Algorithm 1
 5:             $\mathcal{B} = \mathcal{B} \cup (s_t, a_t, r_t, s_{t+1}$        ▷ Execute action and collect transition into replay buffer
 6:         end for
 7:     end for
 8:     for $u = 1, \ldots, n_{\text{updates}}$ do
 9:         $\{(s_i, a_t, r_i, s_{i+1})\}_{i=1}^{B} \sim \mathcal{B}$        ▷ Sample mini-batch from replay buffer
10:         $\phi \leftarrow \phi - \nabla_\phi \mathcal{L}_{\text{critic}}(\phi)$        ▷ Update critic using loss in Equation 1
11:         $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}_{\text{actor}}(\theta)$        ▷ Update actor using loss in Equation 2
12:     end for
13: end for
```

Note that there is a natural decomposition in the value function. It is a product of $P(\tau \in \mathrm{T}_g)$, which is the probability that π reaches a goal when it starts in state s_0, and a return term that depends on how fast π can reach a goal. Consequently, in the stochastic case, the value of a state is dependent on two factors: reaching a goal from s_0 with higher certainty and reaching goals quickly, while in the deterministic case, the value of a state only depends on reaching goals quickly. Thus, we can continue to use value functions to extend our teacher attributes to the stochastic case, keeping in mind that the value function now describes a tradeoff between reaching goals reliably, and reaching goals quickly.

A.2.2 Teacher Attributes for Stochastic MDPs

We have justified continuing to use value functions for defining our teacher attributes in the stochastic MDP case. As such, we simply amend our definitions to account for stochasticity.

Definition A1.1 (*Partial* Teachers) Let π^* denote the optimal policy and $V_{\mathcal{G}}^*(s)$ denote the optimal state value function, with respect to a fixed set of goals $\mathcal{G}$. Then, a teacher policy is *partial* if in some non-empty strict subset $\exists S' \subset \mathcal{S}$, for all states $s \in S'$, we have that $\mathbb{E}_{s'} V_{\mathcal{G}}^*(s') > V_{\mathcal{G}}^*(s)$ for $s' \sim \mathcal{T}(\cdot|s, a)$ and $a \sim \pi(\cdot|s)$.

Here we have just amended the definition to take an expectation over the next state reached when following π.

Definition A1.2 (Sufficient Teachers and Teacher Sets) A teacher policy π is sufficient if it has non-zero value for some start state: $\exists s_0 \sim \rho_0(\cdot)$ such that $V_{\mathcal{G}}^{\pi}(s_0) > 0$. A teacher set $\Pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ is sufficient if there exists a mixture policy that is sufficient (a mixture policy is one that chooses $\pi_\Pi(s) \in \{\pi_1(s), \pi_2(s), \ldots, \pi_N(s)\}$).

This is the same definition as earlier - a non-zero state value can only occur if and only if there is some non-zero probability of reaching the goal.

Definition A1.3 (*Contradictory* Teachers) A teacher policy π_2 is *contradictory* to teacher policy π_1 and a goal set $\mathcal{G}$ if there exists a state $s \in \mathcal{S}$ where following the advice of π_2 causes π_1 to take more timesteps to reach a goal in the goal set. Equivalently, following π_2 in s leads to a state with lower value for π_1: $V_{\mathcal{G}}^{\pi_1}(s') < V_{\mathcal{G}}^{\pi_1}(s)$ for $s' \sim \mathcal{T}(\cdot|s, a)$ and $a \sim \pi_2(\cdot|s)$.

Once again we have just amended the definition to take an expectation over the next state reached when following π_2.

B Additional Details about AC-Teach

B.1 Training the Bayesian Critic and Agent Policy

Algorithm 1 details how the AC-Teach behavioral policy π_b collects samples into the replay buffer $\mathcal{B}$. In Algorithm B.1, we outline how these samples are used to train the Bayesian critic Q_ϕ and actor policy π_θ. Following Henderson et al. [21], we use an α-divergence loss for the critic, but use a behavioral target value, as described in Sec. 4 to mitigate extrapolation error

$$\mathcal{L}_{\text{critic}} = -\frac{1}{\alpha} \log \sum_{k=1}^{K} \exp\Big(-\frac{\alpha\tau}{2}\big(r + \gamma Q_{\phi'}(s', \pi_b(s')) - Q_{\hat{\phi}_k}(s,a)\big)^2\Big) + (1 - p_{\text{drop}})||\phi||_2^2 \;, \quad (1)$$

where K is the number of Monte Carlo dropout samples of the critic network weights $\hat{\phi}_k$ and p_{drop} is the dropout rate. Notice the use of the AC-Teach behavioral policy π_b for computing the target value.

The agent actor loss is simply the normal DDPG actor loss averaged over K Monte Carlo dropout critic weight samples

$$\mathcal{L}_{\text{actor}} = -\sum_{k=1}^{K} Q_{\hat{\phi}_k}(s, \pi_\theta(s)) \quad . \quad (2)$$

B.2 Commitment Probability Estimation

As discussed in Sec. 4, when considering to switch to a policy π_i from a previously used policy π_j, we use the posterior distributions over the action-values of both policy actions to estimate the probability for the expected return of a_i to be larger than the expected return of a_j. We consider the posterior distribution over expected value provided by the Bayesian critic for each action as an independent Gaussian distributed random variable $Z_i = Q_\phi(s, a_i)$ and $Z_j = Q_\phi(s, a_j)$. To obtain these distributions we take K Monte Carlo samples of our posterior critic for each action by sampling different dropout masks [39]. We use the MC samples to fit a Gaussian such that

$$\begin{aligned} \mu_{Z_i} &= \frac{1}{K}\sum_{k=1}^{K} Q_{\hat{\phi}_k}(s, a_i) \\ \sigma^2_{Z_i} &= \frac{1}{K}\sum_{k=1}^{K} (Q_{\hat{\phi}_k}(s, a_i))^2 - \mu^2_{Z_i} + \frac{1}{\tau} \end{aligned} \quad (3)$$

, where τ is the precision of the variance estimate. We can compute the estimate of the probability $P(Z_i > Z_j)$ by leveraging the fact that $Z_i - Z_j \sim \mathcal{N}(\mu_{Z_i} - \mu_{Z_j}, \sigma^2_{Z_i} + \sigma^2_{Z_j})$ and computing $P(Z_i - Z_j) > 0$ using the Gaussian cumulative distribution function. Intuitively, this probability estimate corresponds to the belief that the new policy prescribed by the critic will yield higher Q-value than the policy from the current timestep.

B.3 Mitigating Extrapolation Error

As discussed in Sec. 4, a key challenge when training with teachers is avoiding instability when training the critic from off-policy experience. We observed that basing the behavioral policy on the critic of the learner agent, as opposed to a separate model such as the DQN in Xie et al. [33], significantly helped with this problem, as did using the behavioral target described in Appendix B.1. We hypothesize that since the Bayesian critic is indirectly being used to generate data through the behavioral policy and it is trained on the same distribution of data, the algorithm enjoys similar stability benefits as on-policy methods. We intend to explore this further in future work.

C Additional Experiments

C.1 `Path Following` Experiments

We replicate the set of experiments presented in Fig. 3 and Fig. 4 for the `path-following` task, and also present the performance of the behavioral policies at train time (effectively, the quality of the collected experience the DDPG agent is trained with) in addition to the test time performance of the learned agent. The results are depicted in Fig. C.1. We observe that our proposed method, AC-Teach, converges faster and to larger reward level at test time than the baselines across the variety of teacher setups as shown in the main text for `Path Following` task. Furthermore, unlike the

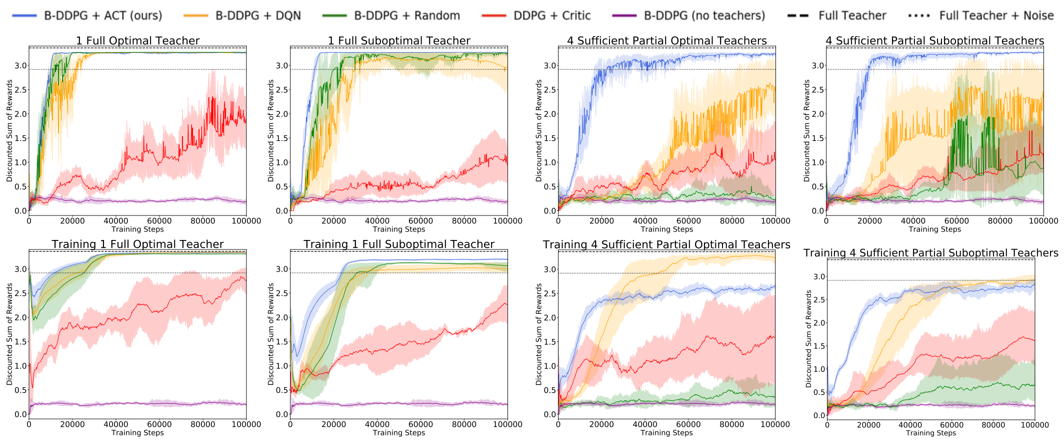


Figure C.1: **Evaluating AC-Teach with *sufficient* teacher sets.** Test and train time agent performance on the `Path Following` task for AC-Teach and other baselines. At test time, several baselines perform almost as well as our method when utilizing just one *sufficient* teacher (leftmost two graphs), but when there are multiple *partial* teachers (rightmost two graphs) our method significantly outperforms all others in both convergence speed and asymptotic performance. At train time, it is notable that the DQN baseline is rarely able to surpass the quality of the teachers it is supplied with and that the performance of the learned agent decreases substantially at test time despite having been trained with the experience of the better-performing DQN behavioral policy at train time.

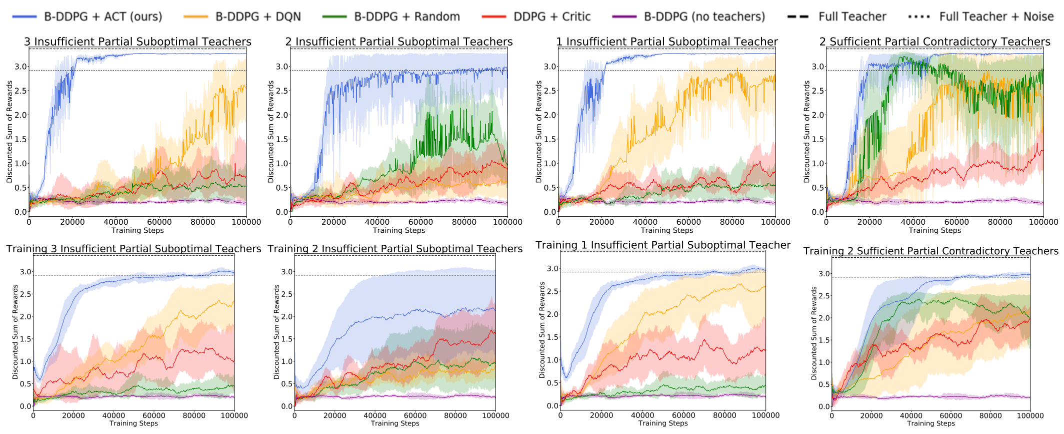


Figure C.2: **Evaluating AC-Teach on *insufficient* and *contradictory* teacher sets.** The first three sets are incomplete since they are missing teachers for 1, 2, and 3 goals respectively, while the last set is *contradictory* because the midpoint teacher and endpoint teacher try to move the agent to different parts of the state space. We see that our method significantly outperforms others in both convergence speed and asymptotic performance, indicating that it is capable of learning parts of the task where no teacher offers useful advice, as well as learning when to leverage each teacher's advice, even when the advice is *contradictory*.

baselines it performs at test time than at train time, indicating that it better avoids extrapolation error or over-reliance on the teachers.

Teacher set A consists of 3 *suboptimal*, *partial* teachers, one that can navigate to each goal. Notice that one of the 4 goals will not have a corresponding teacher. Similarly, teacher sets B and C consist of 2 *suboptimal*, *partial* teachers and 1 *suboptimal*, *partial* teacher respectively. Finally, teacher set D consists of a *midpoint* teacher and an *endpoint* teacher that can be *contradictory*.

Notice that sets A, B, and C all test for the ability to learn from *insufficient* sets, with set C being the most difficult as the teacher only helps with a quarter of the task. Meanwhile, set D tests for the ability to learn from a set of teachers that offers *contradictory* advice, as the midpoint teacher always tries to move to the midpoint between the current and previous goal and the endpoint teacher will move away from the midpoint, towards either one goal or the other, depending on which is closer.

Fig. C.2 demonstrates that our algorithm is consistently able to leverage the incomplete sets of teachers to learn the task - whether learning from 3, 2, or even just 1 of the single-goal teachers. The agent quickly learns where to trust the teachers' expertise and how to act in parts of the state space where teachers do not offer useful advice.

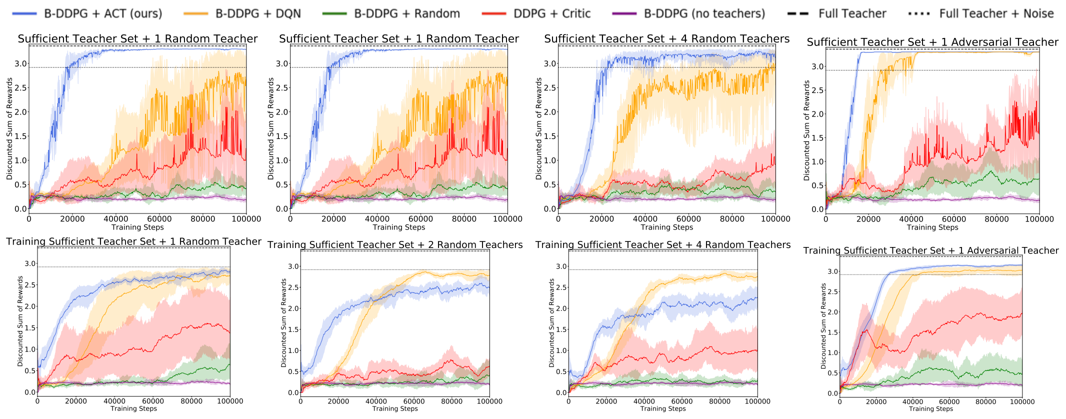


Figure C.3: **Evaluating AC-Teach on large and *adversarial* teacher sets.** Test and train time agent performance on the `Path Following` task for AC-Teach and other baselines for the following teacher sets (from left to right): 4 noisy *partial* teachers + 1 random teacher (set E), 4 noisy *partial* teachers + 2 random teachers (set F), 4 noisy *partial* teachers + 3 random teachers (set G), and 1 *sufficient* teacher + 1 *adversarial* teacher (set H). Our method significantly outperforms the baselines in both convergence speed and asymptotic performance, indicating robustness to large sets of teachers where some teachers are random, or even *adversarial*.

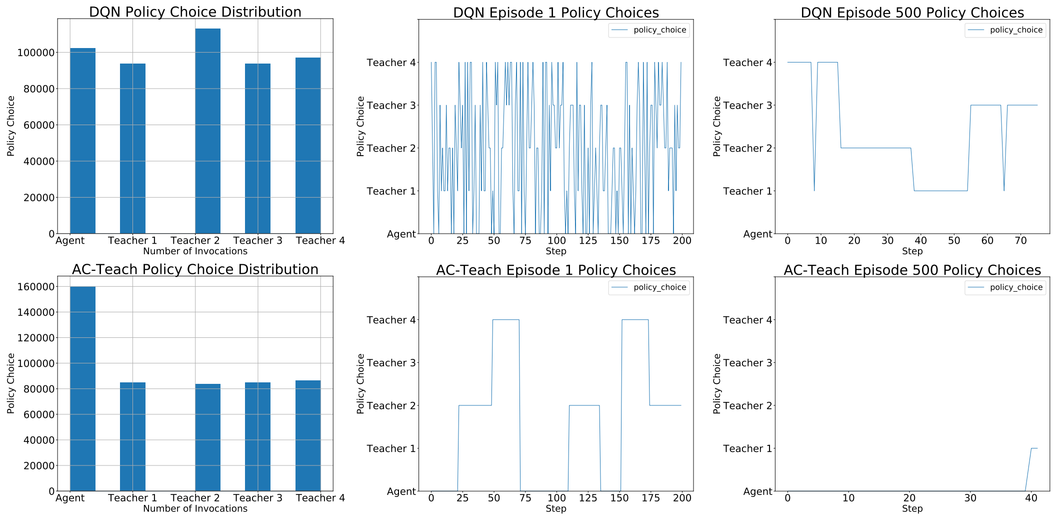


Figure C.4: **Comparing policy choices of the AC-Teach behavioural policy with the DQN behavioural policy.** Results from the `Path Following` task with the sufficient noisy teacher set. The first column demonstrates that over the course of training AC-Teach leverages the agent more than the DQN. The second column demonstrates that our commitment mechanism results in less switching between different policies than the baselines. The third column demonstrates that at the end of training the DQN learns to only leverage the teachers, whereas our method learns to use the agent.

Additionally, we construct four teacher sets to evaluate the tolerance of our algorithm to teacher sets that are larger and lower quality. Teacher sets E, F, and G all consist of four *suboptimal partial* teachers, but they additionally include 1, 2, and 4 additional random teachers respectively. Teacher set H consists of a *suboptimal sufficient* teacher and an *adversarial* teacher that tries to move away from the current goal.

Fig. C.3 shows the results of this experiment. The performance of our algorithm does not degrade in the presence of additional random teachers that do not help to accomplish the task (sets E, F, and G), while our algorithm quickly detects the *adversarial* teacher, learns to ignore it and to query the advice of the *sufficient* teacher in set H.

We also present a breakdown of the policy choices made during training by the baseline DQN behavioural policy and the AC-Teach behavioural policy in Fig. C.4. The figure demonstrates that AC-Teach leverages the learning agent more over the course of training, switches less between policies at the start of training due to its commitment mechanism, and converges to use the learner and not the teachers at the end of training.

C.2 `Hook Sweep` Experiments

We present experiments on the `Hook Sweep` environment in Fig. C.5.

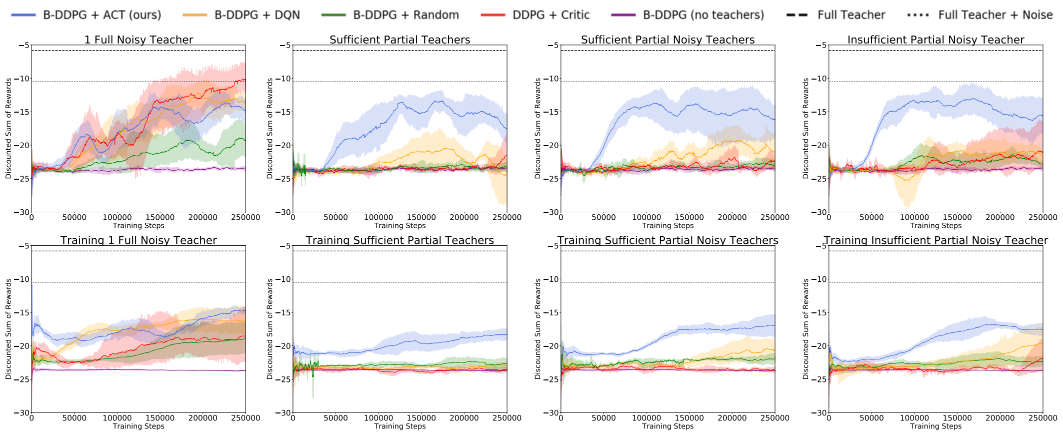


Figure C.5: **Evaluating AC-Teach on different teacher sets on the `Hook Sweep` task.** As with the prior tasks, we varied the types of teachers we trained the agent for the hook sweep task with. As in the prior tasks, our algorithm is demonstrated to be the most robust.

C.3 Ablation Analysis

We test several modified versions of AC-Teach to evaluate the effect of each of its components, in particular different variants of time commitment and behavioral target. Fig. C.6 (a) and (b) depicts the result of our analysis. Interestingly, time commitment does not grant any benefit in the `Path Following` task despite being useful in the `Pick and Place` and `Hook Sweep` tasks. On the contrary, the behavioral target is not relevant for the performance of AC-Teach in the `Pick and Place` and `Hook Sweep` tasks despite being important in the `Path Following`.

We hypothesize that the tasks present different characteristics that benefit (or not) from the features of AC-Teach. In the case of `Path Following`, it includes little stochasticity and so the potential extrapolation error that is countered with the behavioral target is larger. For the `Pick and Place` task, both the end-effector and cube locations are randomized so the collected experience is varied enough, making extrapolation error less problematic. However, the shorter time horizon to complete the `Pick and Place` task and the possibility of pushing the cube away make commitment beneficial.

C.4 Sensitivity Analysis

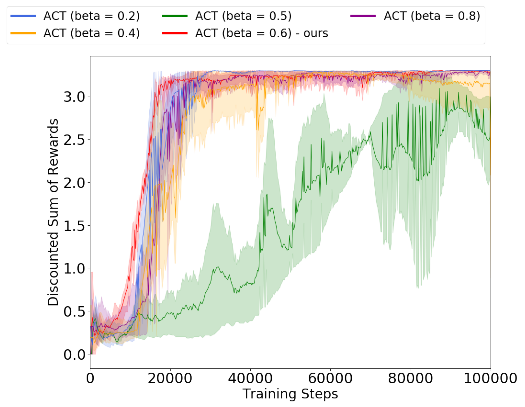
We also evaluate the sensitivity of our algorithm to different parameters and hyperparameters. In our sensitivity analysis we use the `Path Following` task because the simplified dynamics allows to better dissentangle and analyze the results.

Fig. C.6 depict the results of our sensitivity analysis. We observe that most parameters can have a range of values and still work well. An exception to this is that we also observe that our algorithm is sensitive to the value of the dropout precision parameter, τ, which is used in establishing the regularization of the critic network as well as in the computation of the probability of a new policy being better in our commitment mechanism. With more or less samples than the ones we used in our experiments the performance drops significantly.

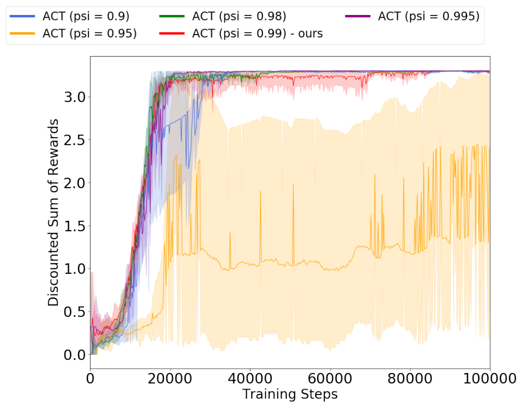
D Experimental Setup

D.1 Tasks

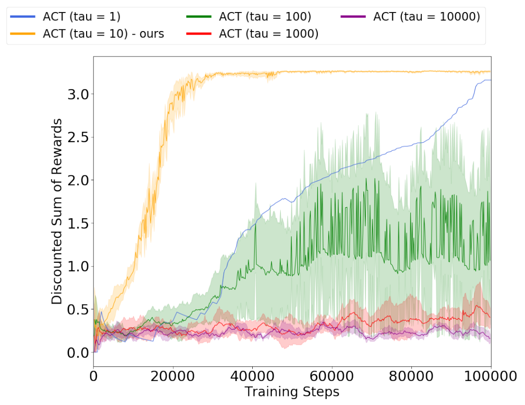
`Path Following`: A point agent starts each episode at the origin of a 2D plane and needs to visit the four corners of a square centered at the origin. These corners must be visited in a specific order that is randomly sampled at each episode. We set the goal locations to be at $(-0.25, -0.25)$, $(-0.25, 0.25)$, $(0.25, -0.25)$, and $(0.25, 0.25)$. A goal location is considered visited



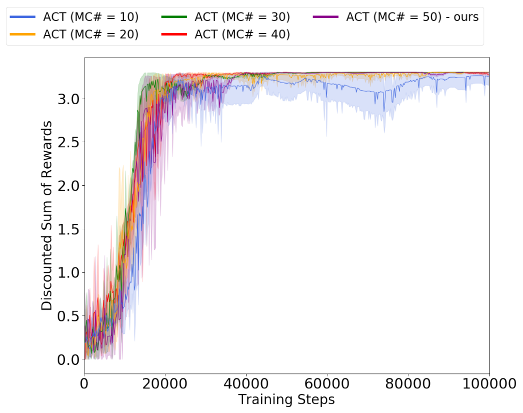
(a)



(b)



(c)



(d)

Figure C.6: **Sensitivity Analysis** to β (a), ψ (b), tau (c) parameters, and to the number of dropout Monte Carlo samples (c)

if the agent position is within 0.05 L2 distance. The observations are 5-dimensional and consist of its current location, the current goal point, and the number of goals left to visit. The agent generates bounded 2D displacements as actions with deterministic outcome. The max offset in each axis per single action is 0.045. A sparse reward of 1 is given when the agent visits each corner, making 4 the maximum undiscounted return per episode. Episodes are 200 steps long and do not terminate early.

`Pick and Place`: A robot manipulation problem from [41]. The goal is to pick up a cube and place it at a target location on the table surface. The initial position of the object and the robot end effector are randomized at each episode start, but the goal is constant. The initial cube location in each episode is centered near $(1.25, 0.55)$, with random offsets of magnitude $[-0.05, 0.05]$ in the x and y planes. The goal location is at $(1.45, 0.55, 0.425)$ – an arbitrarily chosen location near one of the corners of the table. The end-effector begins above the center of the table near $(1.34, 0.75, 0.5)$ with random offsets of magnitude $[-0.05, 0.05]$ in the x and y planes and a random offset of magnitude $[-0.025, 0.025]$ in the z plane. The observation and action space is the same as in Plappert et al. [41]. The agent applies delta position commands to the end effector and can actuate the gripper. Rewards are dense and defined as the negative L2 distance from the cube to the target location. Episodes are 100 steps long and do not terminate early.

`Hook Sweep`: A robot manipulation problem adapted from [31]. The objective is to actuate a robot arm to move a cube to a particular goal location. The cube is initialized out of reach of the robot arm, and so the robot must use a hook to sweep the cube to the goal. The goal location and initial cube location are randomized such that in some episodes the robot arm must use the hook to sweep the cube closer to its base and in other episodes the robot arm must use the hook to push the cube away from its base to a location far from the robot. On every environment reset, with 50% probability, the cube is initialized in front of the hook, and the goal is set to be far away from the robot base, and with 50% probability, the cube is initialized behind the hook, and the goal is set to be close to the robot base. Thus, the robot has to learn to leverage the hook to either sweep the cube closer to itself or sweep the cube further away, depending on the goal initialization. The observation and action space is the same as in Plappert et al. [41], with additional observation dimensions added for the hook state. The agent applies delta position commands to the end effector and can actuate the gripper. Rewards are dense and defined as the negative L2 distance from the cube to the target location. Episodes are 100 steps long and do not terminate early.

D.2 Implementation

We implement the BDDPG agent as well as the BDDPG + DQN and DDPG + Critic baselines on top of the existing implementation in Stable Baselines [42] along with relevant modifications from [21] taken from `https://github.com/Breakend/BayesianPolicyGradients`.

D.3 Training

All experiment results are presented over 5 seeds. Neither observations nor rewards were normalized. The `Path Following` task was run by alternating 200 steps of task interaction with 100 gradient updates to the policy, with a soft update to the target after every gradient update, until $100k$ task interactions were reached. The `Path Following` task was run by alternating 200 steps of task interaction with 40 gradient updates to the policy, with a soft update to the target after every gradient update, until $500k$ task interactions were reached.

D.4 ACT Hyperparameters

Unless otherwise stated for specific experiments, we use commitment threshold $\beta = 0.6$ and commitment decay $\psi = 0.99$ in experiments throughout the paper.

D.5 DDPG Hyperparameters

We use most hyperparameters from [21], with values derived from `https://github.com/Breakend/BayesianPolicyGradients`, as shown in Table D.1a.

Hidden Layers (`Path Following`)	$[64, 64]$
Hidden Layers (`Mujoco`)	$[64, 64, 64]$
DDPG Exploration (`Path Following`)	$\mathcal{N}(0, 0.3)$
DDPG Exploration (`Mujoco`)	$\mathcal{N}(0, 0.1)$
Target Update τ (`Path Following`)	0.01
Target Update τ (`Mujoco`)	0.001
α	0.5
Dropout τ	10.0
Discount Factor	0.99
Keep Prob (`Path Following`)	0.8
Keep Prob (`Mujoco`)	0.9
MC Samples	50
Batch Size	128
Reward Scale	1
Actor L2 Loss (`Path Following`)	0.0
Actor L2 Loss (`Mujoco`)	0.1

(a) DDPG Hyperparameters

Hidden Layers	$[64, 64]$
Learning Rate	$5.0e-4$
Exploration Final ϵ	0.02
Exploration Timesteps	100000
Buffer Size	100000
Discount Factor	0.99
Train Freq	10
Target Network Update Freq	1000
Batch Size	32
Reward Scale	1

(b) DQN Hyperparameters

Parameters are largely the same across the three tasks, except for the several with listed differences above; these were found via manual tuning and seem to reflect the Mujoco tasks being harder.

D.6 DQN hyperparameters

We use DQN with non-prioritized replay, with the hyperparameters listed in Table D.1b.