

Bridging the Imitation Gap by Adaptive Insubordination

Luca Weihs^{*,1}, Unnat Jain^{*,2,†}, Iou-Jen Liu², Jordi Salvador¹,
Svetlana Lazebnik², Aniruddha Kembhavi¹, Alexander Schwing²

¹Allen Institute for AI, ²University of Illinois at Urbana-Champaign

{luca, jordi, anik}@allenai.org

{uj2, iliu3, slazebni, aschwing}@illinois.edu

<https://unnat.github.io/advisor/>

Abstract

In practice, imitation learning is preferred over pure reinforcement learning whenever it is possible to design a teaching agent to provide expert supervision. However, we show that when the teaching agent makes decisions with access to privileged information that is unavailable to the student, this information is marginalized during imitation learning, resulting in an “imitation gap” and, potentially, poor results. Prior work bridges this gap via a progression from imitation learning to reinforcement learning. While often successful, gradual progression fails for tasks that require frequent switches between exploration and memorization. To better address these tasks and alleviate the imitation gap we propose ‘Adaptive Insubordination’ (ADVISOR). ADVISOR dynamically weights imitation and reward-based reinforcement learning losses during training, enabling on-the-fly switching between imitation and exploration. On a suite of challenging tasks set within gridworlds, multi-agent particle environments, and high-fidelity 3D simulators, we show that on-the-fly switching with ADVISOR outperforms pure imitation, pure reinforcement learning, as well as their sequential and parallel combinations.

1 Introduction

Imitation learning (IL) can be remarkably successful in settings where reinforcement learning (RL) struggles. For instance, IL has been shown to succeed in complex tasks with sparse rewards [9, 51, 48], and when the observations are high-dimensional, *e.g.*, in visual 3D environments [35, 58]. To succeed, IL provides the agent with consistent expert supervision at every timestep, making it less reliant on the agent randomly attaining success. To obtain this expert supervision, it is often convenient to use “privileged information,” *i.e.*, information that is unavailable to the student at inference time. This privileged information takes many forms in practice. For instance, in navigational tasks, experts are frequently designed using shortest path algorithms which access the environment’s connectivity graph [*e.g.*, 20]. Other forms of privilege include semantic maps [*e.g.*, 64, 14], the ability to see into “the future” via rollouts [65], and ground-truth world layouts [8]. The following example shows how this type of privileged information can result in IL dramatically failing.

Example 1 (Poisoned Doors). Suppose an agent is presented with $N \geq 3$ doors $d_1, \dots, d_N$. As illustrated in Fig. 1 (for $N = 4$), opening d_1 requires entering an unknown fixed code of length M . Successful code entry results in a reward of 1, otherwise the reward is 0. Since the code is unknown to the agent, it would need to learn the code by trial and error. All other doors can be opened without a code. For some randomly chosen $2 \leq j \leq N$ (sampled each episode), the reward behind d_j is 2 but

^{*}denotes equal contribution by LW and UJ; [†]work done, in part, as an intern at Allen Institute for AI

for all $i \in \{2, \dots, N\} \setminus \{j\}$ the reward behind d_i is -2 . Without knowing j , the optimal policy is to always enter the correct code to open d_1 obtaining an expected reward of 1. In contrast, if the expert is given the privileged knowledge of the door d_j with reward 2, it will always choose to open this door immediately. It is easy to see that an agent without knowledge of j attempting to imitate such an expert will learn to open a door among $d_2, \dots, d_N$ uniformly at random obtaining an expected return of $-2 \cdot (N - 3)/(N - 1)$. In this setting, training with reward-based RL after a ‘warm start’ with IL is strictly worse than starting without it: the agent needs to unlearn its policy and then, by chance, stumble into entering the correct code for door d_1 , a practical impossibility when M is large.

To characterize this imitation failure, we show that training a student to imitate a teacher who uses privileged information results in the student learning a policy which marginalizes out this privileged information. This can result in a sub-optimal, even uniformly random, student policy over a large collection of states. We call the discrepancy between the teacher’s and student’s policy the *imitation gap*. To bridge the imitation gap, we introduce **Adaptive Insubordination** (ADVISOR). ADVISOR adaptively weights imitation and RL losses. Specifically, throughout training we use an auxiliary actor which judges whether the current observation is better treated using an IL or a RL loss. For this, the auxiliary actor attempts to reproduce the teacher’s action using the observations of the student at every step. Intuitively, the weight corresponding to the IL loss is large when the auxiliary actor can reproduce the teacher’s action with high confidence.

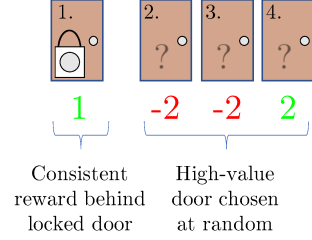


Figure 1: POISONEDDOORS.

We study the benefits of ADVISOR on thirteen tasks, including ‘POISONEDDOORS’ from Ex. 1, a 2D “lighthouse” gridworld, a suite of tasks set within the MINIGRID environment [9, 10], Cooperative Navigation with limited range (COOPNAV) in the multi-agent particle environment (MPE) [47, 42], and two navigational tasks set in 3D, high visual fidelity, simulators of real-world living environments (POINTNAV in AIHABITAT [58] and OBJECTNAV in ROBOTHOR [35, 15]). Our results show that,

- the imitation gap’s size directly impacts agent performance when using modern learning methods,
- ADVISOR is *performant* (outperforming IL and RL baselines), *robust*, and *sample efficient*,
- ADVISOR can succeed even when expert supervision is partially corrupted, and
- ADVISOR can be easily integrated in existing pipelines spanning diverse observations (grids and pixels), actions spaces (discrete and continuous), and algorithms (PPO and MADDPG).

2 Related Work

A series of methods [e.g., 45, 69, 3, 59] have made off-policy deep Q-learning stable for complex environments like Atari Games. Several high-performance (on-policy) policy-gradient methods for deep-RL have also been proposed [60, 46, 38, 72, 65]. For instance, Trust Region Policy Optimization (TRPO) [60] improves sample-efficiency by safely integrating larger gradient steps. Proximal Policy Optimization (PPO) [62] employs a clipped variant of TRPO’s surrogate objective and is widely adopted in the deep RL community. We use PPO as a baseline in our experiments.

As environments get more complex, navigating the search space with only deep RL and simple heuristic exploration (such as ϵ -greedy) is increasingly difficult. Therefore, methods that imitate expert (*i.e.*, teacher) supervision were introduced. A popular approach to imitation learning (IL) is Behaviour Cloning (BC), *i.e.*, use of a supervised classification loss between the policy of the student and expert agents [57, 2]. However, BC suffers from compounding errors. Namely, a single mistake of the student may lead to settings that have never been observed in training [55]. To address this, Data Aggregation (DAGGER) [56] trains a sequence of student policies by querying the expert at states beyond those that would be reached by following only expert actions. IL is further enhanced by, e.g., hierarchies [37], improving over the expert [5, 4, 31], bypassing any intermediate reward function inference [25], and/or learning from experts that differ from the student [19, 30, 17]. Importantly, a sequential combination of IL and RL, *i.e.*, pre-training a model on expert data before letting the agent interact with the environment, performs remarkably well. This strategy has been applied in a wide range of applications – the game of Go [65], robotic and motor skills [53, 34, 52, 54], navigation in visually realistic environments [20, 13, 27, 29], and web & language based tasks [22, 12, 63, 71].

More recent methods mix expert demonstrations with the agent’s own rollouts instead of using a sequential combination of IL followed by RL. Chemali and Lazaric [6] perform policy iteration from expert and on-policy demonstrations. DQfD [24] initializes the replay buffer with expert episodes and adds rollouts of (a pretrained) agent. They weight experiences based on the previous temporal difference errors [59] and use a supervised loss to learn from the expert. For continuous action spaces, DDPGfD [70] analogously incorporates IL into DDPG [39]. POfD [32] improves by adding a demonstration-guided exploration term, *i.e.*, the Jensen-Shannon divergence between the expert’s and the learner’s policy (estimated using occupancy measures). THOR uses suboptimal experts to reshape rewards and then searches over a finite planning horizon [66]. Zhu et al. [78] show that a combination of GAIL [25] and RL can be highly effective for difficult manipulation tasks.

Critically, the above methods have, implicitly or explicitly, been designed under certain assumptions (*e.g.*, the agent operates in an MDP) which imply the expert and student observe the same state. Different from the above methods, we investigate the difference of privilege between the expert policy and the learned policy. Contrary to a sequential, static, or rule-based combination of supervised loss or divergence, we train an auxiliary actor to adaptively weight IL and RL losses. To the best of our knowledge, this hasn’t been studied before. In concurrent work, Warrington et al. [74] address the imitation gap by jointly training their teacher and student to adapt the teacher to the student. For our applications of interest, this work is not applicable as our expert teachers are fixed.

Our approach attempts to reduce the imitation gap directly, assuming the information available to the learning agent is fixed. An indirect approach to reduce this gap is to enrich the information available to the agent or to improve the agent’s memory of past experience. Several works have considered this direction in the context of autonomous driving [11, 21] and continuous control [18]. We expect that these methods can be beneficially combined with the method that we discuss next.

3 ADVISOR

We first introduce notation to define the imitation gap and illustrate how it arises due to ‘policy averaging.’ Using an ‘auxiliary policy’ construct, we then propose ADVISOR to bridge this gap. Finally, we show how to estimate the auxiliary policy in practice using deep networks. In what follows we will use the terms teacher and expert interchangeably. Our use of “teacher” is meant to emphasize that these policies are (1) designed for providing supervision for a student and (2) need not be optimal among all policies.

3.1 Imitation gap

We want an agent to complete task $\mathcal{T}$ in environment $\mathcal{E}$. The environment has states $s \in \mathcal{S}$ and the agent executes an action $a \in \mathcal{A}$ at every discrete timestep $t \geq 0$. For simplicity and w.l.o.g. assume both $\mathcal{A}$ and $\mathcal{S}$ are finite. For example, let $\mathcal{E}$ be a 1D-gridworld in which the agent is tasked with navigating to a location by executing actions to move left or right, as shown in Fig. 2a. Here and below we assume states $s \in \mathcal{S}$ encapsulate historical information so that s includes the full trajectory of the agent up to time $t \geq 0$. The objective is to find a policy π , *i.e.*, a mapping from states to distributions over actions, which maximizes an evaluation criterion. Often this policy search is restricted to a set of feasible policies $\Pi^{\text{feas.}}$, for instance $\Pi^{\text{feas.}}$ may be the set $\{\pi(\cdot; \theta) : \theta \in \mathbb{R}^D\}$ where $\pi(\cdot; \theta)$ is a deep neural network with D -dimensional parameters θ . In classical (deep) RL [45, 46], the evaluation criterion is usually the expected γ -discounted future return.

We focus on the setting of partially-observed Markov decision processes (POMDPs) where an agent makes decisions without access to the full state information. We model this restricted access by defining a *filtration function* $f : \mathcal{S} \rightarrow \mathcal{O}_f$ and limiting the space of feasible policies to those policies $\Pi_f^{\text{feas.}}$ for which the value of $\pi(s)$ depends on s only through $f(s)$, *i.e.*, so that $f(s) = f(s')$ implies $\pi(s) = \pi(s')$. We call any π satisfying this condition an *f -restricted policy* and the set of feasible *f -restricted policies* $\Pi_f^{\text{feas.}}$. In a gridworld example, f might restrict s to only include information local to the agent’s current position as shown in Figs. 2c, 2d. If a *f -restricted policy* is optimal among all other *f -restricted policies*, we say it is *f -optimal*. We call $o \in \mathcal{O}_f$ a *partial-observation* and for any *f -restricted policy* π_f we write $\pi_f(o)$ to mean $\pi_f(s)$ if $f(s) = o$. It is frequently the case that, during training, we have access to a teacher policy which is able to successfully complete the task $\mathcal{T}$. This teacher policy may have access to the whole environment state and thus may be optimal among all policies. Alternatively, the teacher policy may, like the student, only make decisions given partial

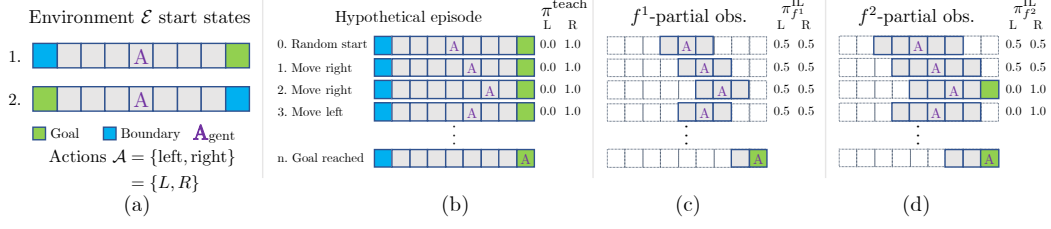


Figure 2: **Effect of partial observability in a 1-dimensional gridworld environment.** (a) The two start states and actions space for 1D-Lighthouse with $N = 4$. (b) A trajectory of the agent following a hypothetical random policy. At every trajectory step we display output probabilities as per the shortest-path expert (π^{teach}) for each state. (c/d) Using the same trajectory from (b) we highlight the partial-observations available to the agent (shaded gray) under different filtration function f^1, f^2 . Notice that, under f^1 , the agent does not see the goal within its first four steps. The policies $\pi_{f^1}^{\text{IL}}, \pi_{f^2}^{\text{IL}}$, learned by imitating π^{teach} , show that imitation results in sub-optimal policies, i.e., $\pi_{f^1}^{\text{IL}}, \pi_{f^2}^{\text{IL}} \neq \pi^{\text{teach}}$.

information (e.g., a human who sees exactly the same inputs as the student). For flexibility we will define the teacher policy as $\pi_{f^{\text{teach}}}^{\text{teach}}$, denoting it is an f^{teach} -restricted policy for some filtration function f^{teach} . For simplicity, we will assume that $\pi_{f^{\text{teach}}}^{\text{teach}}$ is f^{teach} -optimal. Subsequently, we will drop the subscript f^{teach} unless we wish to explicitly discuss multiple teachers simultaneously.

In IL [49, 56], π_f is trained to mimic π^{teach} by minimizing the (expected) cross-entropy between π_f and π^{teach} over a set of sampled states $s \in \mathcal{S}$:

$$\min_{\pi_f \in \Pi_f^{\text{feas}}} \mathbb{E}_{\mu} [CE(\pi^{\text{teach}}, \pi_f)(S)], \quad (1)$$

where $CE(\pi^{\text{teach}}, \pi_f)(S) = -\pi^{\text{teach}}(S) \odot \log \pi_f(S)$, $\odot$ denotes the usual dot-product, and S is a random variable taking values $s \in \mathcal{S}$ with probability measure $\mu : \mathcal{S} \rightarrow [0, 1]$. Often $\mu(s)$ is chosen to equal the frequency with which an exploration policy (e.g., random actions or π^{teach}) visits state s in a randomly initialized episode. When it exists, we denote the policy minimizing Eq. (1) as $\pi_f^{\mu, \pi^{\text{teach}}}$.

When μ and π^{teach} are unambiguous, we write $\pi_f^{\text{IL}} = \pi_f^{\mu, \pi^{\text{teach}}}$.

What happens when there is a difference of privilege (or filtration functions) between the teacher and the student? Intuitively, if the information that a teacher uses to make a decision is unavailable to the student then the student has little hope of being able to mimic the teacher's decisions. As we show in our next example, even when optimizing perfectly, depending on the choice of f and f^{teach} , IL may result in π_f^{IL} being uniformly random over a large collection of states. We call the phenomenon that $\pi_f^{\text{IL}} \neq \pi^{\text{teach}}$ the *imitation gap*.

Example 2 (1D-Lighthouse). We illustrate the imitation gap using a gridworld spanning $\{-N, \dots, N\}$. The two start states correspond to the goal being at either $-N$ or N , while the agent is always initialized at 0 (see Fig. 2a). Clearly, with full state information, π^{teach} maps states to an ‘always left’ or ‘always right’ probability distribution, depending on whether the goal is on the left or right, respectively. Suppose now that the agent’s visibility is constrained to a radius of i (Fig. 2c shows $i = 1$), i.e., an f^i -restricted observation is accessible. An agent following an optimal policy with a visibility of radius i will begin to move deterministically towards any corner, w.l.o.g. assume right. When the agent sees the rightmost edge (from position $N - i$), it will either continue to move right if the goal is visible or, if it’s not, move left until it reaches the goal (at $-N$). Now we may ask: what is the best f^i -restricted policy that can be learnt by imitating π^{teach} (i.e., what is $\pi_{f^i}^{\text{IL}}$)? Tragically, the cross-entropy loss causes $\pi_{f^i}^{\text{IL}}$ to be uniform in a large number of states. In particular, an agent following policy $\pi_{f^i}^{\text{IL}}$ will execute left (and right) actions with probability 0.5, until it is within a distance of i from one of the corners. Subsequently, it will head directly to the goal. See the policies highlighted in Figs. 2c, 2d. The intuition for this result is straightforward: until the agent observes one of the corners it cannot know if the goal is to the right or left and, conditional on its observations, each of these events is equally likely under μ (assumed uniform). Hence for half of these events the teacher will instruct the agent to go right. For the other half the instruction is to go left. See App. A.1 for a rigorous treatment of this example. In Sec. 4 and Fig. 6, we train f^i -restricted policies with

f^j -optimal teachers for a 2D variant of this example. We empirically verify that a student learns a better policy when imitating teachers whose filtration function is closest to their own.

The above example shows: when a student attempts to imitate an expert that is privileged with information not available to the student, the student learns a version of π^{teach} in which this privileged information is marginalized out. We formalize this intuition in the following proposition.

Proposition 1 (Policy Averaging).

In the setting of Section 3.1, suppose that $\Pi^{\text{feas.}}$ contains all f -restricted policies. Then, for any $s \in \mathcal{S}$ with $o = f(s)$, we have that $\pi_f^{\text{IL}}(o) = \mathbb{E}_\mu[\pi^{\text{teach}}(S) \mid f(S) = o]$.

Given our definitions, the proof of this proposition is quite straightforward, see Appendix A.2.

The imitation gap provides theoretical justification for the common practical observation that an agent trained via IL can often be significantly improved by continuing to train the agent using pure RL (e.g., PPO) [42, 14]. Obviously training first with IL and then via pure RL is ad hoc and potentially sub-optimal as discussed in Ex. 1 and empirically shown in Sec. 4. To alleviate this problem, the student should imitate the teacher’s policy only in settings where the teacher’s policy can, in principle, be exactly reproduced by the student. Otherwise the student should learn via ‘standard’ RL. To achieve this we introduce ADVISOR.

3.2 Adaptive Insubordination (ADVISOR) with Policy Gradients

To close the imitation gap, ADVISOR adaptively weights reward-based and imitation losses. Intuitively, it supervises a student by asking it to imitate a teacher’s policy only in those states $s \in \mathcal{S}$ for which the imitation gap is small. For all other states, it trains the student using reward-based RL. To simplify notation, we denote the reward-based RL loss via $\mathbb{E}_\mu[L(\theta, S)]$ for some loss function L .² This loss formulation is general and spans all policy gradient methods, including A2C and PPO. The imitation loss is the standard cross-entropy loss $\mathbb{E}_\mu[CE(\pi^{\text{teach}}(S), \pi_f(S; \theta))]$. Concretely, the ADVISOR loss is:

$$\mathcal{L}^{\text{ADV}}(\theta) = \mathbb{E}_\mu[w(S) \cdot CE(\pi^{\text{teach}}(S), \pi_f(S; \theta)) + (1 - w(S)) \cdot L(\theta, S)] . \quad (2)$$

Our goal is to find a *weight function* $w : \mathcal{S} \rightarrow [0, 1]$ where $w(s) \approx 1$ when the imitation gap is small and $w(s) \approx 0$ otherwise. For this we need an estimator of the distance between π^{teach} and π_f^{IL} at a state s and a mapping from this distance to weights in $[0, 1]$.

We now define a distance estimate $d^0(\pi, \pi_f)(s)$ between a policy π and an f -restricted policy π_f at a state s . We can use any common non-negative distance (or divergence) d between probability distributions on $\mathcal{A}$, e.g., in our experiments we use the KL-divergence. While there are many possible strategies for using d to estimate $d^0(\pi, \pi_f)(s)$, perhaps the simplest of these strategies is to define $d^0(\pi, \pi_f)(s) = d(\pi(s), \pi_f(s))$. Note that this quantity does not attempt to use any information about the fiber $f^{-1}(f(s))$ which may be useful in producing more holistic measures of distances.³ Appendix A.3 considers how those distances can be used in lieu of d^0 . Next, using the above, we need to estimate the quantity $d^0(\pi^{\text{teach}}, \pi_f^{\text{IL}})(s)$.

Unfortunately it is, in general, impossible to compute $d^0(\pi^{\text{teach}}, \pi_f^{\text{IL}})(s)$ exactly as it is intractable to compute the optimal minimizer π_f^{IL} . Instead we leverage an estimator of π_f^{IL} which we term π_f^{aux} , and which we will define in the next section.

Given π_f^{aux} we obtain the estimator $d^0(\pi^{\text{teach}}, \pi_f^{\text{aux}})$ of $d^0(\pi^{\text{teach}}, \pi_f^{\text{IL}})$. Additionally, we make use of the monotonically decreasing function $m_\alpha : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$, where $\alpha \geq 0$. We define our weight

²For readability, we implicitly make three key simplifications. First, computing the expectation $\mathbb{E}_\mu[\dots]$ is generally intractable, hence we cannot directly minimize losses such as $\mathbb{E}_\mu[L(\theta, S)]$. Instead, we approximate the expectation using rollouts from μ and optimize the empirical loss. Second, recent RL methods adjust the measure μ over states as optimization progresses while we assume it to be static for simplicity. Our final simplification regards the degree to which any loss can be, and is, optimized. In general, losses are often optimized by gradient descent and generally no guarantees are given that the global optimum can be found. Extending our presentation to encompass these issues is straightforward but notationally dense.

³Measures using such information include $\max_{s' \in f^{-1}(f(s))} d(\pi(s'), \pi_f(s))$ or a corresponding expectation instead of the maximization, i.e., $\mathbb{E}_\mu[d(\pi(S), \pi_f(S)) \mid f(S) = o]$.

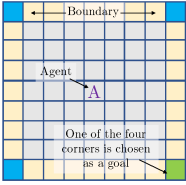
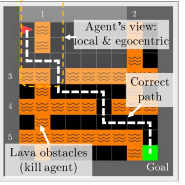
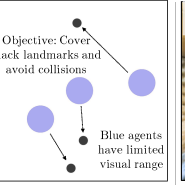
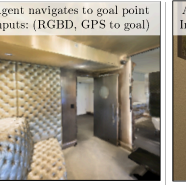
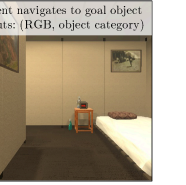
Tasks					
Simulators	–	MiniGrid	MPE	AIHabitat	AI2-THOR
Experts	Optimal experts with varying visibility radii $j \geq$ student range	Global shortest-path action, unaffected by switch/light	RL agent trained with full visibility range	Global shortest-path expert (access to scene layout)	Global shortest-path expert (access to scene layout)
Objective	Study how the size of the imitation gap influences performance	Fast simulation enables extensive experimentation and a robustness study	Demonstrate ADVISOR can be applied in continuous, multi-agent, environments	Study ADVISOR's performance within a rich visual environment	Demonstrate that ADVISOR succeeds in diverse 3D environments

Figure 4: **Representative tasks from experiments.** 2D-LH: Harder 2D variant of the gridworld task introduced in Ex. 2. LAVACROSSING: one of our 8 tasks in the MINIGRID environment requiring safe navigation. We test up-to 15×15 grids with 10 lava rivers. COOPNAV: A multi-agent cooperative task set in multi-agent particle environments. POINTNAV: An agent embodied in the AIHABITAT environment must navigate using egocentric visual observations to a goal position specified by a GPS coordinate. OBJECTNAV: An agent in ROBOTHOR must navigate to an object of a given category.

function $w(s)$ for $s \in \mathcal{S}$ as:

$$w(s) = m_\alpha(d^0(\pi^{\text{teach}}, \pi_f^{\text{aux}})(s)) \quad \text{with} \quad m_\alpha(x) = e^{-\alpha x}. \quad (3)$$

3.3 The Auxiliary Policy π^{aux} : Estimating π_f^{IL} in Practice

In this section we describe how we can, during training, obtain an *auxiliary policy* π_f^{aux} which estimates π_f^{IL} . Given this auxiliary policy we estimate $d^0(\pi^{\text{teach}}, \pi_f^{\text{IL}})(s)$ using the plug-in estimator $d^0(\pi^{\text{teach}}, \pi_f^{\text{aux}})(s)$. While plug-in estimators are intuitive and simple to define, they need not be statistically efficient. In Appendix A.4 we consider possible strategies for improving the statistical efficiency of our plug-in estimator via prospective estimation.

In Fig. 3 we provide an overview of how we compute the estimator π_f^{aux} via deep nets. As is common practice [46, 23, 26, 50, 44, 9, 7, 28, 73], the policy net $\pi_f(\cdot; \theta)$ is composed via $a_\nu \circ r_\lambda$ with $\theta = (\nu, \lambda)$, where a_ν is the *actor head* (possibly complemented in actor-critic models by a *critic head* v_ν) and r_λ is called the *representation network*. Generally a_ν is lightweight, for instance a linear layer or a shallow MLP followed by a soft-max function, while r_λ is a deep, and possibly recurrent neural, net. We add another actor head $a_{\nu'}$ to our existing network which shares the underlying representation r_λ , i.e., $\pi_f^{\text{aux}} = a_{\nu'} \circ r_\lambda$. We experiment with the actors sharing their representation r_λ and estimating π_f^{IL} via two separate networks, i.e., $\theta' = (\nu', \lambda')$. In practice we train $\pi_f(\cdot; \theta)$ and $\pi_f^{\text{aux}}(\cdot; \theta')$ jointly using stochastic gradient descent, as summarized in Alg. A.1.

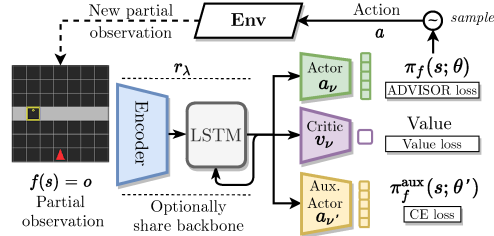


Figure 3: **Model overview.** An auxiliary actor is added and trained only using IL. The ‘main’ actor policy is trained using the ADVISOR loss.

4 Experiments

We rigorously compare ADVISOR to IL methods, RL methods, and popularly-adopted (but often ad hoc) IL & RL combinations. In particular, we evaluate 15 learning methods. We do this over thirteen tasks – realizations of Ex. 1 & Ex. 2, eight tasks of varying complexity within the fast, versatile MINIGRID environment [9, 10], Cooperative Navigation (COOPNAV) with reduced visible range in the multi-agent particle environment (MPE) [47, 41], PointGoal navigation (POINTNAV) using the

Tasks → Training routines ↓	PD -	LAVACROSSING				WALLCROSSING			
		Base Ver.	Corrupt Exp.	Faulty Switch	Once Switch	Base Ver.	Corrupt Exp.	Faulty Switch	Once Switch
RL	0	0	0	0.01	0	0.09	0.07	0.12	0.05
IL	-0.59	0.88	0.02	0.02	0	0.96	0.05	0.17	0.11
IL & RL	-0.17	0.94	0.74	0.04	0	0.97	0.18	0.17	0.1
Demo. Based	-0.09	0.96	0.2	0.02	0	0.97	0.07	0.18	0.11
ADV. Based (ours)	1	0.96	0.94	0.77	0.8	0.97	0.31	0.38	0.45

Table 1: **Expected rewards for the POISONEDDOORS task and MINIGRID tasks.** For each of our 15 training routines we report the expected maximum validation set performance (when given a budget of 10 random hyperparameter evaluations) after training for $\approx 300k$ steps in POISONEDDOORS and $\approx 1Mn$ steps in our 8 MINIGRID tasks. The maximum reward is 1 for the MINIGRID tasks.

Gibson dataset in AIHABITAT [77, 58], and ObjectGoal Navigation (OBJECTNAV) in ROBOTHOR [15].⁴ Furthermore, to probe robustness, we train 50 hyperparameter variants for each of the 15 learning methods for our MINIGRID tasks. We find ADVISOR-based methods outperform or match performance of all baselines.

All code to reproduce our experiments will be made public under the Apache 2.0 license.⁵ The environments used are public for academic and commercial use under the Apache 2.0 (MINIGRID and ROBOTHOR) and MIT licence (MPE and AIHABITAT).

4.1 Tasks

Detailed descriptions of our tasks (and teachers) are deferred to Appendix A.5. See Fig. 4 for a high-level overview of 5 representative tasks.

4.2 Baselines and ADVISOR-based Methods

We briefly introduce baselines and variants of our ADVISOR method. Further details of all methods are in Appendix A.7. For fairness, the same model architecture is shared across all methods (recall Fig. 3, Sec. 3.3). We defer implementation details to Appendix A.8.

- **RL only.** Proximal Policy Optimization [62] serves as the pure RL baseline for all our tasks with a discrete action space. For the continuous and multi-agent COOPNAV task, we follow prior work and adopt MADDPG [41, 40].
- **IL only.** IL baselines where supervision comes from an expert policy with different levels of teacher-forcing (tf), *i.e.*, $tf=0$, tf annealed from $1 \rightarrow 0$, and $tf=1$. This leads to Behaviour Cloning (BC), Data Aggregation (DAGger or $\dagger$), and $BC^{tf=1}$, respectively [57, 2, 56].
- **IL & RL.** Baselines that use a mix of IL and RL losses, either in sequence or in parallel. These are popularly adopted in the literature to warm-start agent policies. Sequential combinations include BC then PPO (BC $\rightarrow$ PPO), DAGger then PPO ($\dagger \rightarrow$ PPO), and $BC^{tf=1} \rightarrow$ PPO. The parallel combination of BC + PPO(static) is a static analog of our adaptive combination of IL and RL losses.
- **Demonstration-based.** These agents imitate expert demonstrations and hence get no supervision beyond the states in the demonstrations. We implement BC^{demo} , its combination with PPO ($BC^{demo} +$ PPO), and Generative Adversarial Imitation Learning (GAIL) [25].
- **ADVISOR-based (ours).** Our Adaptive Insubordination methodology can learn from an expert policy and can be given a warm-start via BC or DAGger. This leads to ADVISOR (ADV), $BC^{tf=1} \rightarrow$ ADV, and $\dagger \rightarrow$ ADV) baselines. Similarly, $ADV^{demo} +$ PPO employs Adaptive Insubordination to learn from expert demonstrations while training with PPO on on-policy rollouts.

4.3 Evaluation

Fair Hyperparameter Tuning. Often unintentionally done, extensively tuning the hyperparameters (hps) of a proposed method and not those of the baselines can introduce unfair bias into evaluations.

⁴The ROBOTHOR environment is a sub-environment of AI2-THOR [35].

⁵See <https://unnat.github.io/advisor/> for an up-to-date link to this code.

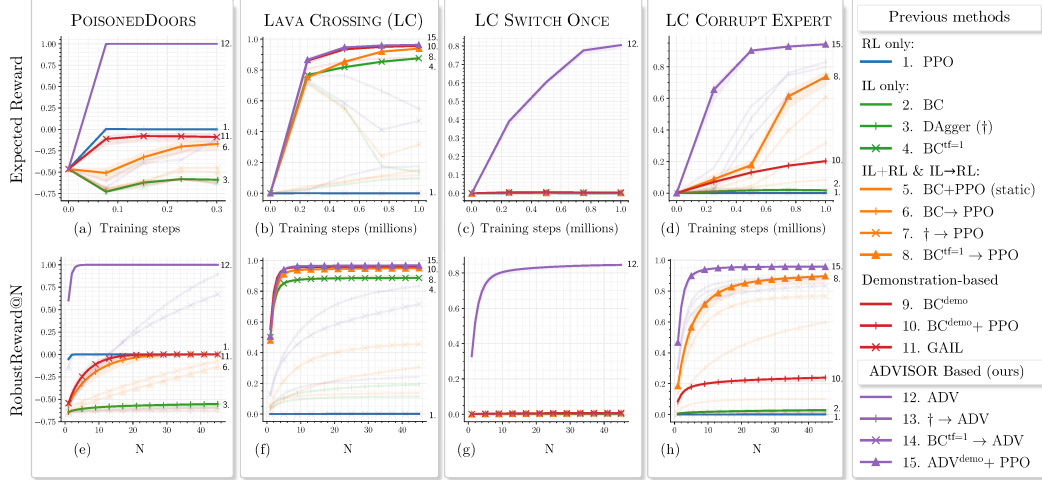


Figure 5: **Evaluation following [16]**. Plots for 15 training routines in four selected tasks (additional plots in appendix). For clarity, we highlight the best performing training routine within five categories, *e.g.*, RL only, IL only *etc.* (details in Sec. 4.2) with all other plots shaded lighter. (a)-(d) As described in Sec. 4.3 we plot *RobustReward@10* at multiple points during training. (e)-(f) Plots of *RobustReward@N* for values of $N \in \{1, \dots, 45\}$. Recall that *RobustReward@N* is the expected validation reward of best-found model when allowed N random hyperparameter evaluations.

Tasks → Training routines ↓	PointGoal Navigation				ObjectGoal Navigation				Cooperative Navigation	
	SPL @10%	SPL @100%	Success @10%	Success @100%	SPL @10%	SPL @100%	Success @10%	Success @100%	Reward @10%	Reward @100%
RL only	30.9	54.7	54.7	79.0	6.7	13.1	11.1	31.6	-561.8	-456.0
IL only	30.1	68.7	35.5	76.7	3.8	9	8.8	13.6	-460.3	-416.7
IL + RL static	48.9	71.5	56.7	78.2	6.5	11.3	11.7	19.8	-475.5	-424.6
ADVISOR (ours)	57.7	77.1	67.3	88.2	11.9	14.1	22.7	29.9	-419.9	-405.6

Table 2: **Quantitative results for high-fidelity visual environments and continuous control**. Validation set performance after 10% and 100% of training has completed for four training routines on the POINTNAV, OBJECTNAV, and COOPNAV tasks (specifics of these routines can be found in the Appendix). For POINTNAV and OBJECTNAV we include the common success weighted path length (SPL) metric [1] in addition to the success rate.

We avoid this by considering two strategies. For PD and all MINIGRID tasks, we follow recent best practices [16]. Namely, we tune each method by randomly sampling a fixed number of hps and reporting, for each baseline, an estimate of

$$\text{RobustReward}@K = \mathbb{E}[\text{Val. reward of best model from } k \text{ random hyperparam. evaluations}] \quad (4)$$

for $1 \leq k \leq 45$. For this we must train 50 models per method, *i.e.*, 750 for each of these nine tasks. In order to show learning curves over training steps we also report *RobustReward@10* at 5 points during training. More details in Appendix A.9. For 2D-LH, we tune the hps of a competing method and use these hps for all other methods.

Training. For the eight MINIGRID tasks, we train each of the 50 training runs for 1 million steps. For 2D-LH/PD, models saturate much before $3 \cdot 10^5$ steps. POINTNAV, OBJECTNAV, and COOPNAV are trained for standard budgets of 50Mn, 100Mn, and 1.5Mn steps. Details are in Appendix A.10.

Metrics. We record standard metrics for each task. This includes avg. rewards (PD, MINIGRID tasks, and OBJECTNAV), and avg. episode lengths (2D-LH). Following visual navigation works [1, 58, 15], we report success rates and success-weighted path length (SPL) for POINTNAV and OBJECTNAV. In the following, we report a subset of the above and defer additional plots to Appendix A.11.

4.4 Results

In the following, we include takeaways based on the results in Fig. 5, Fig. 6, Tab. 1, and Tab. 2.

Smaller imitation gap $\implies$ better performance. A central claim of our paper is that the imitation gap is not merely a theoretical concern: the degree to which the teacher is privi-

leged over the student has significant impact on the student’s performance. To study this empirically, we vary the degree to which teachers are privileged over its students in our 2D-LH task. In particular, we use behavior cloning to train an f^i -restricted policy (*i.e.*, an agent that can see i grid locations away) using an f^j -optimal teacher 25 times. Each policy is then evaluated on 200 random episodes and the average episode length (lower being better) is recorded. For select i, j pairs we show boxplots of the 25 average episode lengths in Fig. 6. See our appendix for similar plots when using other training routines (*e.g.*, ADVISOR). Grey vertical lines show optimal average episode lengths for f^i -restricted policies. We find that training an f^i -restricted policy with an f^j -expert results in a near optimal policy when $i = j$ but even small increases in j dramatically decrease performance. While performance tends to drop with increasing j , the largest i, j gaps do not consistently correspond to the worst performing models. While this seems to differ from our results in Ex. 2, recall that there the policy μ was fixed while here it varies through training, resulting in complex learning dynamics. Surprisingly we also find that, even when there is no imitation gap (*e.g.*, the $i = j$ case), ADVISOR can outperform BC, see App. A.6.

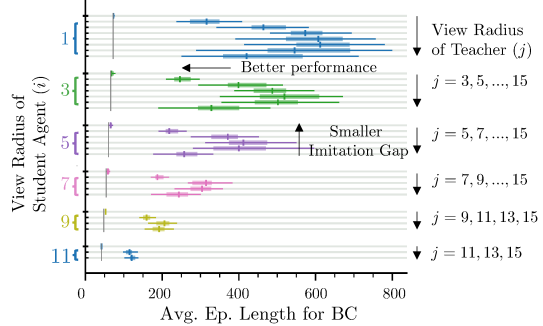


Figure 6: The size of the imitation gap directly impacts performance (in 2D-LH).

ADVISOR outperforms, even in complex visual environments. Across all of our tasks, ADVISOR-based methods perform as well or better than competing methods. In particular, see Tab. 1 for our results on the POISONEDDOORS (PD) and MINIGRID tasks and Tab. 2 for our results on the POINTNAV, OBJECTNAV, and COOPNAV tasks. 2D-LH results are deferred to the Appendix.

While the strong performance of ADVISOR is likely expected on our PD, MINIGRID, and 2D-LH tasks (indeed we designed a subset of these with the explicit purpose of studying the imitation gap), it is nonetheless surprising to see that in the PD and LC ONCE SWITCH tasks, all non-ADVISOR methods completely fail. Moreover, it is extremely promising to see that ADVISOR can provide substantial benefits in a variety of standard tasks, namely OBJECTNAV, POINTNAV, and COOPNAV with limited visible range. Note that OBJECTNAV and POINTNAV are set in 3D high-fidelity visual environments while COOPNAV requires multi-agent collaboration in a continuous space.

ADVISOR is sample efficient. To understand the sample efficiency of ADVISOR, we plot validation set performance over training of select tasks (see Figures 5a to 5d) and, in Table 2 we show performance of our models after 10% of training has elapsed for the OBJECTNAV, POINTNAV, and COOPNAV tasks. Note that in Table 2, ADVISOR trained models frequently reach better performance after 10% of training than other methods manage to reach by the end of training.

ADVISOR is robust. Rigorously studying sensitivity to hyperparameter choice requires retraining every method under consideration tens to hundreds of times. This computational task can make evaluating our methods on certain tasks infeasible (training a single POINTNAV or OBJECTNAV model can easily require a GPU-week of computation). Because of these computational constraints, we limit our study of robustness to the PD and MINIGRID tasks. In Figures 5e to 5h (additional results in Appendix) we plot, for each of the 15 evaluated methods, how the expected performance of each method behaves as we increase the budget of random hyperparameter evaluations. In general, relatively few hyperparameter evaluations are required for ADVISOR before a high performance model is expected to be found.

Expert demonstrations can be critical to success. While it is frequently assumed that on-policy expert supervision is better than learning from off-policy demonstrations, we found several instances in our MINIGRID experiments where demonstration-based methods outperformed competing methods. See, for example, Figures 5b and 5f. In such cases our demonstration-based ADVISOR variant (see Appendix A.7 for details) performed very well.

ADVISOR helps even when the expert is corrupted. In LC CORRUPT EXPERT and WC CORRUPT EXPERT, where the expert is designed to be corrupted (outputting random actions as supervision) when the agent gets sufficiently close to the goal. While ADVISOR was not designed with the

possibility of corrupted experts in mind, Figures 5d and 5h (see also Table 1) show that ADVISOR can succeed despite this corruption.

5 Conclusion

We propose the *imitation gap* as one explanation for the empirical observation that imitating “more intelligent” teachers can lead to worse policies. While prior work has, implicitly, attempted to bridge this gap, we introduce a principled adaptive weighting technique (ADVISOR), which we test on a suite of thirteen tasks. Due to the fast rendering speed of MINIGRID, PD, and 2D-LH, we could undertake a study where we trained over 6 billion steps, to draw statistically significant inferences.

6 Limitations and Societal Impact

While we have attempted to robustly evaluate our proposed ADVISOR methodology, we have primarily focused our experiments on navigational tasks where shortest path experts can be quickly computed. Further work is needed to validate that ADVISOR can be successful in other domains, *e.g.*, imitation in interactive robotic tasks or natural language applications.

While the potential for direct negative societal impact of this work is small, it is worth noting that, in enabling agents to learn more effectively from expert supervision, this work makes imitation learning a more attractive option to RL researchers. If expert supervision is obtained from humans, RL agents trained with such data will inevitably reproduce any (potentially harmful) biases of these humans.

Acknowledgements

This material is based upon work supported in part by the National Science Foundation under Grants No. 1563727, 1718221, 1637479, 165205, 1703166, 2008387, 2045586, 2106825, MRI #1725729, NIFA award 2020-67021-32799, Samsung, 3M, Sloan Fellowship, NVIDIA Artificial Intelligence Lab, Allen Institute for AI, Amazon, AWS Research Awards, and Siebel Scholars Award. We thank Nan Jiang and Tanmay Gangwani for feedback on this work.

References

- [1] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.
- [2] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence*, 1995.
- [3] M. G. Bellemare, G. Ostrovski, A. Guez, P. S. Thomas, and R. Munos. Increasing the action gap: New operators for reinforcement learning. In *AAAI*, 2016.
- [4] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé. Reinforcement learning from demonstration through shaping. In Q. Yang and M. J. Wooldridge, editors, *IJCAI*, 2015.
- [5] K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daume, and J. Langford. Learning to search better than your teacher. In *ICML*, 2015.
- [6] J. Chemali and A. Lazaric. Direct policy iteration with demonstrations. In Q. Yang and M. J. Wooldridge, editors, *IJCAI*, 2015.
- [7] C. Chen, U. Jain, C. Schissler, S. V. A. Gari, Z. Al-Halah, V. K. Ithapu, P. Robinson, and K. Grauman. Audio-visual embodied navigation. In *ECCV*, 2020.
- [8] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *CoRL*, 2020.
- [9] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *ICLR*, 2018.

- [10] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for openai gym. <https://github.com/maximecb/gym-minigrid>, 2018.
- [11] F. Codevilla, M. Müller, A. M. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *ICRA*, 2018.
- [12] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *ICCV*, 2017.
- [13] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *CVPR*, 2018.
- [14] A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural Modular Control for Embodied Question Answering. In *CoRL*, 2018.
- [15] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In *CVPR*, 2020.
- [16] J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith. Show your work: Improved reporting of experimental results. In *EMNLP*, 2019.
- [17] T. Gangwani and J. Peng. State-only imitation with transition dynamics mismatch. In *ICLR*, 2020.
- [18] T. Gangwani, J. Lehman, Q. Liu, and J. Peng. Learning belief representations for imitation learning in pomdps. In A. Globerson and R. Silva, editors, *UAI*, 2019.
- [19] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. In *ICLR*, 2017.
- [20] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. In *CVPR*, 2017.
- [21] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, and A. Kendall. Urban driving with conditional imitation learning. In *ICRA*, 2020.
- [22] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma. Dual learning for machine translation. In *NeurIPS*, 2016.
- [23] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.
- [24] 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 *AAAI*, 2018.
- [25] J. Ho and S. Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016.
- [26] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *ICLR*, 2017.
- [27] U. Jain, L. Weihs, E. Kolve, M. Rastegari, S. Lazebnik, A. Farhadi, A. G. Schwing, and A. Kembhavi. Two body problem: Collaborative visual task completion. In *CVPR*, 2019.
- [28] U. Jain, L. Weihs, E. Kolve, A. Farhadi, S. Lazebnik, A. Kembhavi, and A. G. Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *ECCV*, 2020.
- [29] U. Jain, I.-J. Liu, S. Lazebnik, A. Kembhavi, L. Weihs, and A. G. Schwing. Gridtopix: Training embodied agents with minimal supervision. In *ICCV*, 2021.
- [30] N. Jiang. On value functions and the agent-environment boundary. *arXiv preprint arXiv:1905.13341*, 2019.

- [31] M. Jing, X. Ma, W. Huang, F. Sun, C. Yang, B. Fang, and H. Liu. Reinforcement learning from imperfect demonstrations under soft expert guidance. In *AAAI*, 2020.
- [32] B. Kang, Z. Jie, and J. Feng. Policy optimization with demonstrations. In *ICML*, 2018.
- [33] D. Kingma and J. Ba. A method for stochastic optimization. In *CVPR*, 2017.
- [34] J. Kober and J. R. Peters. Policy search for motor primitives in robotics. In *NeurIPS*, 2009.
- [35] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2019.
- [36] I. Kostrikov. Pytorch implementations of reinforcement learning algorithms. <https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail>, 2018.
- [37] H. Le, N. Jiang, A. Agarwal, M. Dudik, Y. Yue, and H. Daumé. Hierarchical imitation and reinforcement learning. In *ICML*, 2018.
- [38] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.
- [39] 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. In *ICLR*, 2016.
- [40] I.-J. Liu, R. Yeh, and A. G. Schwing. PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning. In *CORL*, 2019.
- [41] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *NeurIPS*, 2017.
- [42] R. Lowe, A. Gupta, J. N. Foerster, D. Kiela, and J. Pineau. On the interaction between supervision and self-play in emergent communication. In *ICLR*, 2020.
- [43] A. R. Mahmood, H. P. van Hasselt, and R. S. Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In *NeurIPS*, 2014.
- [44] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. In *ICLR*, 2017.
- [45] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.
- [46] 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 *ICML*, 2016.
- [47] I. Mordatch and P. Abbeel. Emergence of Grounded Compositional Language in Multi-Agent Populations. In *AAAI*, 2018.
- [48] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *ICRA*, 2018.
- [49] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 2018.
- [50] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.
- [51] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 2018.
- [52] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 2008.

- [53] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 1991.
- [54] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS*, 2018.
- [55] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.
- [56] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.
- [57] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Machine Learning Proceedings*, 1992.
- [58] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *ICCV*, 2019.
- [59] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized Experience Replay. In *ICLR*, 2016.
- [60] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, 2015.
- [61] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
- [62] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [63] T. T. Shi, A. Karpathy, L. J. Fan, J. Hernandez, and P. Liang. World of bits: An open-domain platform for web-based agents. In *ICML*, 2017.
- [64] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *CVPR*, 2020.
- [65] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.
- [66] W. Sun, J. A. Bagnell, and B. Boots. Truncated horizon policy search: Combining reinforcement learning & imitation learning. In *ICLR*, 2018.
- [67] M. van der Laan and S. Gruber. One-step targeted minimum loss-based estimation based on universal least favorable one-dimensional submodels. *The international journal of biostatistics*, 12(1):351–378, 2016.
- [68] A. van der Vaart. *Asymptotic Statistics*. Asymptotic Statistics. Cambridge University Press, 2000. ISBN 9780521784504. URL <https://books.google.com/books?id=UEuQEM5RjWgC>.
- [69] H. van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.
- [70] M. Vecerik, 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.
- [71] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Yang Wang. Video captioning via hierarchical reinforcement learning. In *CVPR*, 2018.
- [72] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. In *ICLR*, 2017.

- [73] S. Wani, S. Patel, U. Jain, A. X. Chang, and M. Savva. Multi-on: Benchmarking semantic map memory using multi-object navigation. In *NeurIPS*, 2020.
- [74] A. Warrington, J. W. Lavington, A. Ścibior, M. Schmidt, and F. Wood. Robust asymmetric learning in pomdps. *CoRR*, abs/2012.15566, 2020. URL <https://arxiv.org/abs/2012.15566>.
- [75] L. Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387251456.
- [76] L. Weihs, J. Salvador, K. Kotar, U. Jain, K.-H. Zeng, R. Mottaghi, and A. Kembhavi. Allenact: A framework for embodied ai research. *arXiv preprint arXiv:2008.12760*, 2020.
- [77] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018.
- [78] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess. Reinforcement and imitation learning for diverse visuomotor skills. In *RSS*, 2018.

Appendix: Bridging the Imitation Gap by Adaptive Insubordination

The appendix includes theoretical extensions of ideas presented in the main paper and details of empirical analysis. We structure the appendix into the following subsections:

- A.1 A formal treatment of Ex. 2 on 1D-Lighthouse.
- A.2 Proof of Proposition 1.
- A.3 Distance measures beyond $d_\pi^0(\pi_f)(s) = d(\pi(s), \pi_f(s))$ utilized in ADVISOR.⁶
- A.4 Future strategies for improving statistical efficiency of $d_{\pi^{\text{teach}}}^0(\pi_f^{\text{IL}})(s)$ estimator and a prospective approach towards it.
- A.5 Descriptions of all the tasks that we evaluate baselines on, including values for grid size, obstacles, corruption distance *etc.* We also include details about observation space for each of these tasks.
- A.6 Initial results showing that ADVISOR can outperform behavior cloning even when there is no imitation gap.
- A.7 Additional details about nature of learning, expert supervision and hyperparameters searched for each baseline introduced in Sec. 4.2.
- A.8 Details about the underlying model architecture for all baselines across different tasks.
- A.9 Methodologies adopted for ensuring fair hyperparameter tuning of previous baselines when comparing ADVISOR to them.
- A.10 Training implementation including maximum steps per episode, reward structure and computing infrastructure adopted for this work. We clearly summarize all structural and training hyperparameters for better reproducibility.
- A.11 Additional results including plots for all tasks to supplement Fig. 5, a table giving an expanded version of the Tab. 1, and learning curves to supplement Tab. 2.

A Additional Information

A.1 Formal treatment of Example 2

Let $N \geq 1$ and consider a 1-dimensional grid-world with states $\mathcal{S} = \{-N, N\} \times \{0, \dots, T\} \times \{-N, \dots, N\}^T$. Here $g \in \{-N, N\}$ are possible goal positions, elements $t \in \{0, \dots, T\}$ correspond to the episode’s current timestep, and $(p_i)_{i=1}^T \in \{-N, \dots, N\}^T$ correspond to possible agent trajectories of length T . Taking action $a \in \mathcal{A} = \{\text{left}, \text{right}\} = \{-1, 1\}$ in state $(g, t, (p_i)_{i=1}^T) \in \mathcal{S}$ results in the deterministic transition to state $(g, t+1, (p_1, \dots, p_t, \text{clip}(p_t + a, -N, N), 0, \dots, 0))$. An episode start state is chosen uniformly at random from the set $\{(\pm N, 0, (0, \dots, 0))\}$ and the goal of the agent is to reach some state $(g, t, (p_i)_{i=1}^T)$ with $p_t = g$ in the fewest steps possible. We now consider a collection of filtration functions f^i , that allow the agent to see spaces up to i steps left/right of its current position but otherwise has perfect memory of its actions. See Figs. 2c, 2d for examples of f^1 - and f^2 -restricted observations. For $0 \leq i \leq N$ we define f^i so that

$$f^i(g, t, (p_i)_{i=1}^T) = ((\ell_0, \dots, \ell_t), (p_1 - p_0, \dots, p_t - p_{t-1})) \quad \text{and} \quad (5)$$

$$\ell_j = (1_{[p_j+k=N]} - 1_{[p_j+k=-N]} \mid k \in \{-i, \dots, i\}) \quad \text{for } 0 \leq j \leq t. \quad (6)$$

Here ℓ_j is a tuple of length $2 \cdot i + 1$ and corresponds to the agent’s view at timestep j while $p_{k+1} - p_k$ uniquely identifies the action taken by the agent at timestep k . Let π^{teach} be the optimal policy given full state information so that $\pi^{\text{teach}}(g, t, (p_i)_{i=1}^T) = (1_{[g=-N]}, 1_{[g=N]})$ and let μ be a uniform distribution over states in $\mathcal{S}$. It is straightforward to show that an agent following policy $\pi_{f^i}^{\text{IL}}$ will take random actions until it is within a distance of i from one of the corners $\{-N, N\}$ after which it will head directly to the goal, see the policies highlighted in Figs. 2c, 2d. The intuition for this result is straightforward: until the agent observes one of the corners it cannot know if the goal is to the right or left and, conditional on its observations, each of these events is equally likely under μ . Hence in half of these events the expert will instruct the agent to go right and in the other half

⁶We overload main paper’s notation $d^0(\pi, \pi_f)(s)$ with $d_\pi^0(\pi_f)(s)$

Algorithm A.1: On-policy ADVISOR algorithm overview. Some details omitted for clarity.

Input: Trainable policies $(\pi_f, \pi_f^{\text{aux}})$, expert policy π^{teach} , rollout length L , environment $\mathcal{E}$.

Output: Trained policy

```

1 begin
2   Initialize the environment  $\mathcal{E}$ 
3    $\theta \leftarrow$  randomly initialized parameters
4   while Training completion criterion not met do
5     Take  $L$  steps in the environment using  $\pi_f(\cdot; \theta)$  and record resulting rewards and
      observations (restarting  $\mathcal{E}$  whenever the agent has reached a terminal state)
6     Evaluate  $\pi_f^{\text{aux}}(\cdot; \theta)$  and  $\pi^{\text{teach}}$  at each of the above steps
7      $L \leftarrow$  the empirical version of the loss from Eq. (2) computed using the above rollout
8     Compute  $\nabla_{\theta} L$  using backpropagation
9     Update  $\theta$  using  $\nabla_{\theta} L$  via gradient descent
10  return  $\pi_f(\cdot; \theta)$ 

```

to go left. The cross entropy loss will thus force $\pi_{f_i}^{\text{IL}}$ to be uniform in all such states. Formally, we will have, for $s = (g, t, (p_i)_{i=1}^T)$, $\pi_{f_i}^{\text{IL}}(s) = \pi^{\text{teach}}(s)$ if and only if $\min_{0 \leq q \leq t} (p_q) - i \leq -N$ or $\max_{0 \leq q \leq t} (p_q) + i \geq N$ and, for all other s , we have $\pi_{f_i}^{\text{IL}}(s) = (1/2, 1/2)$. In Sec. 4, see also Fig. 6, we train f^i -restricted policies with f^j -optimal teachers for a 2D variant of this example. ■

A.2 Proof of Proposition 1

We wish to show that the minimizer of $\mathbb{E}_{\mu}[-\pi_{f^e}^{\text{teach}}(S) \odot \log \pi_f(S)]$ among all f -restricted policies π_f is the policy $\bar{\pi} = \mathbb{E}_{\mu}[\pi^{\text{teach}}(S) \mid f(S)]$. This is straightforward, by the law of iterated expectations and as $\pi_f(s) = \pi_f(f(s))$ by definition. We obtain

$$\begin{aligned}
\mathbb{E}_{\mu}[-\pi_{f^e}^{\text{teach}}(S) \odot \log \pi_f(S)] &= -\mathbb{E}_{\mu}[E_{\mu}[\pi_{f^e}^{\text{teach}}(S) \odot \log \pi_f(S) \mid f(S)]] \\
&= -\mathbb{E}_{\mu}[E_{\mu}[\pi_{f^e}^{\text{teach}}(S) \odot \log \pi_f(f(S)) \mid f(S)]] \\
&= -\mathbb{E}_{\mu}[E_{\mu}[\pi_{f^e}^{\text{teach}}(S) \mid f(S)] \odot \log \pi_f(f(S))] \\
&= \mathbb{E}_{\mu}[-\bar{\pi}(f(S)) \odot \log \pi_f(f(S))] .
\end{aligned} \tag{7}$$

Now let $s \in \mathcal{S}$ and let $o = f(s)$. It is well known, by Gibbs' inequality, that $-\bar{\pi}(o) \odot \log \pi_f(o)$ is minimized (in $\pi_f(o)$) by letting $\pi_f(o) = \bar{\pi}(o)$ and this minimizer is feasible as we have assumed that Π_f contains *all* f -restricted policies. Hence it follows immediately that Eq. (7) is minimized by letting $\pi_f = \bar{\pi}$ which proves the claimed proposition.

A.3 Other Distance Measures

As discussed in Section 3.2, there are several different choices one may make when choosing a measure of distance between the expert policy π^{teach} and an f -restricted policy π_f at a state $s \in \mathcal{S}$. The measure of distance we use in our experiments, $d_{\pi^{\text{teach}}}^0(\pi_f)(s) = d(\pi^{\text{teach}}(s), \pi_f(s))$, has the (potentially) undesirable property that $f(s) = f(s')$ does not imply that $d_{\pi^{\text{teach}}}^0(\pi_f)(s) = d_{\pi^{\text{teach}}}^0(\pi_f)(s')$. While an in-depth evaluation of the merits of different distance measures is beyond this current work, we suspect that a careful choice of such a distance measure may have a substantial impact on the speed of training. The following proposition lists a collection of possible distance measures with a conceptual illustration given in Fig. A.1.

Proposition 2. *Let $s \in \mathcal{S}$ and $o = f(s)$ and for any $0 < \beta < \infty$ define, for any policy π and f -restricted policy π_f ,*

$$d_{\mu, \pi}^{\beta}(\pi_f)(s) = E_{\mu}[(d_{\pi}^0(\pi_f)(S))^{\beta} \mid f(S) = f(s)]^{1/\beta}, \tag{8}$$

with $d_{\mu, \pi}^{\infty}(\pi_f)(s)$ equalling the essential supremum of $d_{\pi}^0(\pi_f)$ under the conditional distribution $P_{\mu}(\cdot \mid f(S) = f(s))$. As a special case note that

$$d_{\mu, \pi}^1(\pi_f)(s) = E_{\mu}[d_{\pi}^0(\pi_f)(S) \mid f(S) = f(s)].$$

Then, for all $\beta \geq 0$ and $s \in \mathcal{S}$ (almost surely μ), we have that $\pi(s) \neq \pi_f(f(s))$ if and only if $d_\pi^\beta(\pi_f)(s) > 0$.

Proof. This statement follows trivially from the definition of π^{IL} and the fact that $d(\pi, \pi') \geq 0$ with $d(\pi, \pi') = 0$ if and only if $\pi = \pi'$. $\square$

The above proposition shows that any d^β can be used to consistently detect differences between π^{teach} and π_f^{IL} , i.e., it can be used to detect the imitation gap. Notice also that for any $\beta > 0$ we have that $d_{\mu, \pi^{\text{teach}}}^\beta(\pi_f^{\text{IL}})(s) = d_{\mu, \pi^{\text{teach}}}^\beta(\pi_f^{\text{IL}})(s')$ whenever $f(s) = f(s')$.

As an alternative to using d^0 , we now describe how $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{IL}})(s)$ can be estimated in practice during training. Let π_f^{aux} be an estimator of π_f^{IL} as usual. To estimate $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{IL}})(s)$ we assume we have access to a function approximator $g_\psi : \mathcal{O}_f \rightarrow \mathbb{R}$ parameterized by $\psi \in \Psi$, e.g., a neural network. Then we estimate $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{IL}})(s)$ with $g_{\hat{\psi}}$ where $\hat{\psi}$ is taken to be the minimizer of the loss

$$\mathcal{L}_{\mu, \pi^{\text{teach}}, \pi_f^{\text{aux}}}(\psi) = E_\mu \left[\left(d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(f(S))) - g_\psi(f(S)) \right)^2 \right]. \quad (9)$$

The following proposition then shows that, assuming that $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}}) \in \{g_\psi \mid \psi \in \Psi\}$, $g_{\hat{\psi}}$ will equal $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})$ and thus $g_{\hat{\psi}}$ may be interpreted as a plug-in estimator of $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{IL}})$.

Proposition 3. For any $\psi \in \Psi$,

$$\mathcal{L}_{\mu, \pi^{\text{teach}}, \pi_f^{\text{aux}}}(\psi) = E_\mu [(d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(f(S)))^2] + c,$$

where $c = E_\mu [(d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(f(S))) - d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S))^2]$ is constant in ψ and this implies that if $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}}) \in \{g_\psi \mid \psi \in \Psi\}$ then $g_{\hat{\psi}} = d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})$.

Proof. In the following we let $O_f = f(S)$. We now have that

$$\begin{aligned} & E_\mu [(d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(O_f)) - g_\psi(O_f))^2] \\ &= E_\mu [((d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(O_f)) - d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S)) + (d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f)))^2] \\ &= E_\mu [(d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(O_f)) - d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S))^2] + E_\mu [(d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f))^2] \\ &\quad + 2 \cdot E_\mu [(d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(O_f)) - d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S)) \cdot (d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f))] \\ &= c + E_\mu [(d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f))^2] \\ &\quad + 2 \cdot E_\mu [(d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(O_f)) - d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S)) \cdot (d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f))]. \end{aligned}$$

Now as $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(s) = d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(s')$ for any s, s' with $f(s) = f(s')$ we have that $d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f)$ is constant conditional on O_f and thus

$$\begin{aligned} & E_\mu [(d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(O_f)) - d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S)) \cdot (d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f)) \mid O_f] \\ &= E_\mu [(d(\pi^{\text{teach}}(S), \pi_f^{\text{aux}}(O_f)) - d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S)) \mid O_f] \cdot E_\mu [d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f) \mid O_f] \\ &= E_\mu [d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) \mid O_f] \cdot E_\mu [d_{\mu, \pi^{\text{teach}}}^1(\pi_f^{\text{aux}})(S) - g_\psi(O_f) \mid O_f] \\ &= 0. \end{aligned}$$

Combining the above results and using the law of iterated expectations gives the desired result. $\square$

A.4 Future Directions in Improving Distance Estimators

In this section we highlight possible directions towards improving the estimation of $d_{\pi^{\text{teach}}}^0(\pi_f^{\text{IL}})(s)$ for $s \in \mathcal{S}$. As a comprehensive study of these directions is beyond the scope of this work, our aim in this section is intuition over formality. We will focus on d^0 here but similar ideas can be extended to other distance measures, e.g., those in Sec. A.3.

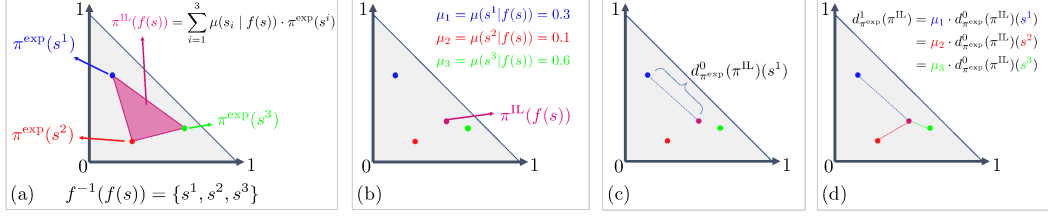


Figure A.1: **Concept Illustration.** Here we illustrate several of the concepts from our paper. Suppose our action space $\mathcal{A}$ contains three elements. Then for any $s \in \mathcal{S}$ and policy π , the value $\pi(s)$ can be represented as a single point in the 2-dimensional probability simplex $\{(x, y) \in \mathbb{R}^2 \mid x \geq 0, y \geq 0, x + y \leq 1\}$ shown as the grey area in (a). Suppose that the fiber $f^{-1}(f)$ contains the three unique states s^1, s^2 , and s^3 . In (a) we show the hypothetical values of π^{exp} when evaluated at these points. Proposition 1 says that $\pi^{\text{IL}}(s)$ lies in the convex hull of $\{\pi^{\text{teach}}(s^i)\}_{i=1}^3$ visualized as a magenta triangle in (a). Exactly where $\pi^{\text{IL}}(s)$ lies depends on the probability measure μ , in (b) we show how a particular instantiation of μ may result in a realization of $\pi^{\text{IL}}(s)$ (not to scale). (c) shows how $d_{\pi^{\text{teach}}}^1$ measures the distance between $\pi^{\text{teach}}(s^1)$ and $\pi^{\text{IL}}(s^1)$. Notice that it ignores s^2 and s^3 . In (d), we illustrate how $d_{\pi^{\text{teach}}}^0$ produces a “smoothed” measure of distance incorporating information about all s^i .

As discussed in the main paper, we estimate $d_{\pi^{\text{teach}}}^0(\pi_f^{\text{IL}})(s)$ by first estimating π_f^{IL} with π_f^{aux} and then forming the “plug-in” estimator $d_{\pi^{\text{teach}}}^0(\pi_f^{\text{aux}})(s)$. For brevity, we will write $d_{\pi^{\text{teach}}}^0(\pi_f^{\text{aux}})(s)$ as $\hat{d}$. While such plug-in estimators are easy to estimate and conceptually compelling, they need not be statistically efficient. Intuitively, the reason for this behavior is because we are spending too much effort in trying to create a high quality estimate π_f^{aux} of π_f^{IL} when we should be willing to sacrifice some of this quality in service of obtaining a better estimate of $d_{\pi^{\text{teach}}}^0(\pi_f^{\text{IL}})(s)$. Very general work in this area has brought about the targeted maximum-likelihood estimation (TMLE) [67] framework. Similar ideas may be fruitful in improving our estimator $\hat{d}$.

Another weakness of $\hat{d}$ discussed in the main paper is that it is not prospective. In the main paper we assume, for readability, that we have trained the estimator π_f^{aux} before we train our main policy. In practice, we train π_f^{aux} alongside our main policy. Thus the quality of π_f^{aux} will improve throughout training. To clarify, suppose that, for $t \in [0, 1]$, $\pi_{f,t}^{\text{aux}}$ is our estimate of π_f^{IL} after $(100 \cdot t)\%$ of training has completed. Now suppose that $(100 \cdot t)\%$ of training has completed and we wish to update our main policy using the ADVISOR loss given in Eq. (2). In our current approach we estimate $d_{\pi^{\text{teach}}}^0(\pi_f^{\text{IL}})(s)$ using $d_{\pi^{\text{teach}}}^0(\pi_{f,t}^{\text{aux}})(s)$ when, ideally, we would prefer to use $d_{\pi^{\text{teach}}}^0(\pi_{f,1}^{\text{aux}})(s)$ from the end of training. Of course we will not know the value of $d_{\pi^{\text{teach}}}^0(\pi_{f,1}^{\text{aux}})(s)$ until the end of training but we can, in principle, use time-series methods to estimate it. To this end, let q_ω be a time-series model with parameters $\omega \in \Omega$ (e.g., q_ω might be a recurrent neural network) and suppose that we have stored the model checkpoints $(\pi_{f,i/K}^{\text{aux}} \mid i/K \leq t)$. We can then train q_ω to perform forward prediction, for instance to minimize

$$\sum_{j=1}^{\lfloor t \cdot K \rfloor} \left(d_{\pi^{\text{teach}}}^0(\pi_{f,j/K}^{\text{aux}})(s) - q_\omega(s, (\pi_{f,i/K}^{\text{aux}}(s))_{i=1}^{j-1}) \right)^2,$$

and then use this trained q_ω to predict the value of $d_{\pi^{\text{teach}}}^0(\pi_{f,1}^{\text{aux}})(s)$. The advantage of this prospective estimator q_ω is that it can detect that the auxiliary policy will eventually succeed in exactly imitating the expert in a given state and thus allow for supervising the main policy with the expert cross entropy loss earlier in training. The downside of such a method: it is significantly more complicated to implement and requires running inference using saved model checkpoints.

A.5 Additional Task Details

In Figure 4 we gave a quick qualitative glimpse at the various tasks we use in our experiments. Here, we provide additional details for each of them along with information about observation space associated with each task. For training details for the tasks, please see Sec. A.10. Our experiments

were primarily run using the AllenAct learning framework [76], see `AllenAct.org` for details and tutorials.

A.5.1 PoisonedDoors (PD)

This environment is a reproduction of our example from Sec. 1. An agent is presented with $N = 4$ doors $d_1, \dots, d_4$. Door d_1 is locked, requiring a fixed $\{0, 1, 2\}^{10}$ code to open, but always results in a reward of 1 when opened. For some randomly chosen $j \in \{2, 3, 4\}$, opening door d_j results in a reward of 2 and for $i \notin \{1, j\}$, opening door d_i results in a reward of -2 . The agent must first choose a door after which, if it has chosen door 1, it must enter the combination (receiving a reward of 0 if it enters the incorrect combination) and, otherwise, the agent immediately receives its reward. See Fig. 1.

A.5.2 2D-Lighthouse (2D-LH)

2D variant of the exemplar grid-world task introduced in Ex. 2, aimed to empirically verify our analysis of the imitation gap. A reward awaits at a randomly chosen corner of a square grid of size $2N + 1$ and the agent can only see the local region, a square of size $2i + 1$ about itself (an f^i -restricted observation). Additionally, all f^i allow the agent access to it's previous action. As explained in Ex. 2, we experiment with optimizing f^i -policies when given supervision from f^j -optimal teachers (*i.e.*, experts that are optimal when restricted to f^j -restricted observations). See Fig. A.2 for an illustration.

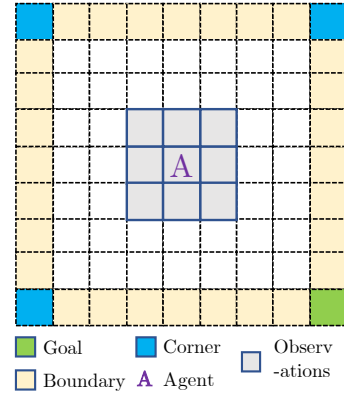


Figure A.2: 2D-LIGHTHOUSE

A.5.3 LavaCrossing (LC)

Initialized on the top-left corner the agent must navigate to the bottom-right goal location. There exists at least one path from start to end, navigating through obstacles. Refer to Fig. 4 where, for illustration, we show a simpler grid. Here the episode terminates if the agent steps on any of the lava obstacles. This LC environment has size 25×25 with 10 lava rivers ('S25, N10' as per the notation of [10]), which are placed vertically or horizontally across the grid. The expert is a shortest path agent with access to the entire environment's connectivity graph and is implemented via the `networkx` python library.

A.5.4 WallCrossing (WC)

Similar to LAVACROSSING in structure and expert, except that obstacles are walls instead of lava. Unlike lava (which immediately kills the agent upon touching), the agent may run into walls without consequence (other than wasting time). Our environment is of size 25×25 with 10 walls ('S25, N10').

A.5.5 WC/LC Switch

In this task the agent faces a more challenging filtration function. In addition to navigational actions, agents for this task have a 'switch' action. Using this switch action, the agents can switch-on the lights of an otherwise darkened environment which is implemented as an observation tensor of all zeros. In WC, even in the dark, an agent can reach the target by taking random actions with non-negligible probability. Achieving this in LC is nearly impossible as random actions will, with high probability, result in stepping into lava and thereby immediately end the episode.

We experiment with two variants of this 'switch' – ONCE and FAULTY. In the ONCE SWITCH variant, once the 'switch' action is taken, the lights remain on for the remainder of the episode. This is implemented as the unaffected observation tensor being available to the agent. In contrast, in the FAULTY SWITCH variant, taking the 'switch' action will only turn the lights on for a single timestep. This is implemented as observations being available for one timestep followed by zero tensors (unless the 'switch' action is executed again).

The expert for these tasks is the same as for WC and LC. Namely, the expert always takes actions along the shortest path from the agents current position to the goal and is unaffected by whether the light is on or off. For the expert-policy-based methods this translates to the learner agent getting perfect (navigational) supervision while struggling in the dark, with no cue for trying the switch action. For the expert-demonstrations-based methods this translates to the demonstrations being populated with blacked-out observations paired with perfect actions: such actions are, of course, difficult to imitate. As FAULTY is more difficult than ONCE (and LC more difficult than WC) we set grid sizes to reduce the difference in difficulty between tasks. In particular, we choose to set WC ONCE SWITCH on a (S25, N10) grid and the LC ONCE SWITCH on a (S15, N7) grid. Moreover, WC FAULTY SWITCH is set with a (S15, N7) grid and LC FAULTY SWITCH with a (S9, N4) grid.

A.5.6 WC/LC Corrupt

In the SWITCH task, we study agents with observations affected by a challenging filtration function. In this task we experiment with corrupting the expert’s actions. The expert policy flips over to a random policy when the expert is N_C steps away from the goal. For the expert-policy-based method this translates to the expert outputting uniformly random actions once it is within N_C steps from the target. For the expert-demonstrations-based methods this translates to the demonstrations consisting of some valid (observation, expert action) tuples, while the tuples close to the target have the expert action sampled from a uniform distribution over the action space. WC CORRUPT is a (S25, N10) grid with $N_C = 15$, while the LC CORRUPT is significantly harder, hence is a (S15, N7) grid with $N_C = 10$.

A.5.7 PointGoal Navigation

In PointGoal Navigation, a randomly spawned agent must navigate to a goal specified by a relative-displacement vector. The observation space is composed of rich egocentric RGB observations ($256 \times 256 \times 3$) with a limited field of view. The action space is $\{\text{move_ahead}, \text{rotate_right}, \text{rotate_left}, \text{stop}\}$. The task was formulated by [1] and implemented for the AIHABITAT simulator by [58]. Our reward structure, train/val/test splits, PointNav dataset, and implementation follow [58]. RL agents are trained using PPO following authors’ implementation⁷. The IL agent is trained with on-policy behavior cloning using the shortest-path action. A static combination of the PPO and BC losses (*i.e.* a simple sum of the PPO loss and IL cross entropy loss) is also used as a competing baseline for ADVISOR. Note that the agent observes a filtered egocentric observation while the shortest-path action is inferred from the entire environment state leading to a significant imitation gap. We train on the standard Gibson set of 76 scenes, and report metrics as an average over the val. set consisting of 14 unseen scenes in AIHABITAT. We use a budget of 50 million frames, *i.e.*, ~ 2 days of training on 4 NVIDIA TitanX GPUs, and 28 CPUs for each method.

A.5.8 ObjectGoal Navigation

In ObjectGoal Navigation within the RoboTHOR environment, a randomly spawned agent must navigate to a goal specified by an object category. In particular, the agent must search it’s environment to find an object of the given category and take a stop action (which ends the episode regardless of success) when that object is within 1m of the agent and visible. The observation space is composed of rich egocentric RGB observations ($300 \times 400 \times 3$) with a limited field of view. The action space is $\{\text{move_ahead}, \text{rotate_right}, \text{rotate_left}, \text{look_up}, \text{look_down}, \text{stop}\}$. The OBJECTNAV task within the RoboTHOR environment was proposed by [15], we use the version of this task corresponding to the 2021 RoboTHOR ObjectNav Challenge⁸ and use this challenge’s reward structure, dataset, train/val/test splits, and their baseline model architecture. This challenge provides implementations of PPO and DAGger where the DAGger agent is trained with supervision coming from a shortest-path expert. We implement our ADVISOR methodology (with no teacher forcing) as well as a baseline where we simply sum PPO and IL losses. We use a budget of 100 million frames, *i.e.*, ~ 2 -5 days of training, 8 NVIDIA TitanX GPUs, and 56 CPUs for each method. At every update step we use 60 rollouts of length 128 and perform 4 gradient steps with the rollout.

⁷<https://github.com/facebookresearch/habitat-lab>

⁸<https://ai2thor.allenai.org/robothor/cvpr-2021-challenge>

A.5.9 Cooperative Navigation

In Cooperative Navigation, there are three agents and three landmarks. The goal of the three agents is to cover the three landmarks. Agents are encouraged to move toward uncovered landmarks and get penalized when they collide with each other. Agents have limited visibility range. The agents can only observe other agents and landmarks within its visibility range (euclidean distance to the agent). The action space has five dimensions. The first dimension is no-op, and the other four dimensions represent the forward, backward, left, and right force applied to the agent. The RL agents are trained with MADDPG [41] with a permutation invariant critic [40]. The IL agents are trained using DAgger. The experts are pre-trained RL agents with no limits to their visibility range. Following [41, 40], we use a budget of 1.5 million environment steps. We use one NVIDIA GTX1080 and 2 CPUs to train these agents.

A.5.10 Observation spaces

2D-LH. Within our 2D-LH environment we wish to train our agent in the context of Proposition 1 so that the agent may learn any f -restricted policy. As the 2D-LH environment is quite simple, we are able to uniquely encode the state observed by an agent using a $4^4 \cdot 5^2 = 6400$ dimensional $\{0, 1\}$ -valued vector such that any f -restricted policy can be represented as a linear function applied to this observation (followed by a soft-max).⁹

PD. Within the PD environment the agent’s observed state is very simple: at every timestep the agent observes an element of $\{0, 1, 2, 3\}$ with 0 denoting that no door has yet been chosen, 1 denoting that the agent has chosen door d_1 but has not begun entering the code, 2 indicating that the agent has chosen door d_1 and has started entering the code, and 3 representing the final terminal state after a door has been opened or combination incorrectly entered.

MINIGRID. The MINIGRID environments [10] enable agents with an egocentric “visual” observation which, in practice, is an integer tensor of shape $7 \times 7 \times 3$, where the channels contain integer labels corresponding to the cell’s type, color, and state. Kindly see [10, 9] for details. For the above tasks, the cell types belong to the set of (empty, lava, wall, goal).

POINTNAV. Agents in the POINTNAV task observe, at every step, egocentric RGB observations ($256 \times 256 \times 3$) of their environment along with a relative displacement vector towards the goal (*i.e.* a 2d vector specifying the location of the goal relative the goal). See Figure 4 for an example of one such egocentric RGB image.

OBJECTNAV. Agents in the OBJECTNAV task observe, at every step, egocentric RGB observations ($300 \times 400 \times 3$) of their environment along with an object category (*e.g.* “BaseballBat”) specifying their goal. See Figure 4 for an example of one such egocentric RGB image. Note that agents in the OBJECTNAV task are generally also allowed access to egocentric depth frames, we do not use these depth frames in our experiments as their use slows simulation speed.

COOPNAV. At each step, each agent in COOPNAV task observes a 14-dimensional vector, which contains the absolute location and speed of itself, the relative locations to the three landmarks, and the relative location to other two agents.

A.6 ADVISOR can outperform BC in the no-imitation-gap setting

Recall the setting of our 2D-LH experiments in Section 4.4 where we train f^i -restricted policies (*i.e.*, an agent that can see i grid locations away) using f^j -optimal teachers. In particular, we train 25 policies on each i, j pair where for $1 \leq i \leq j \leq 15$ and i, j are both odd. Each trained policy is then evaluated on 200 random episodes and we record average performance across various metrics across these episodes. Complementing Fig. 6 from the main paper, Fig. A.3 shows the box plots of the trained policies average episode lengths, lower being better, when training with BC, BC $\rightarrow$ PPO, ADVISOR, and PPO (PPO does not use expert supervision so we simply report the performance of PPO trained f^i -restricted policies for each i).

As might be expected: ADVISOR has consistently low episode lengths across all i, j pairs suggesting that ADVISOR is able to mitigate the impact of the imitation gap. One question that is not well-

⁹As the softmax function prevents us from learning a truly deterministic policy we can only learn a policy arbitrarily close to such policies. In our setting, this distinction is irrelevant.

Method	% converged to near optimal performance							
	$i = 1$	3	5	7	9	11	13	15
ADV	1	1	1	1	1	1	1	1
BC	1	0.72	0.52	0.72	0.68	0.84	0.96	1
†	0.88	0.56	0.24	0.08	0.52	0.96	1	1

Table A.1: **Comparing efficiency of IL vs. ADVISOR in 2D-LH.** Here we report the percentage of runs (of 25 runs per (method, i) pair) that various methods converged to near-optimal performance (within 5% of optimal) with a budget of 300,000 training steps. Here i corresponds to an f^i -restricted (student) policy trained with expert supervision from an f^i -optimal teacher (*i.e.* the ‘no-imitation-gap’ setting).

answered by Fig. A.3 is that of the relative performance of ADVISOR and IL methods when *there is no imitation gap*, namely the $i = j$ case. As ADVISOR requires the training of an auxiliary policy in addition (but, in parallel) to a main policy, we test the sample efficiency of ADVISOR head-on with IL methods. Table A.1 records the percentage of runs in which ADV, BC, and † attain near optimal (within 5%) performance when trained in the no-imitation-gap setting (*i.e.* $i = j$) for different grid visibility i . We find that only ADVISOR consistently reaches near-optimal performance within the budget of 300,000 training steps. We suspect that this is due to the RL loss encouraging early exploration that results in the agent more frequently entering states where imitation learning is easier. This interpretation is supported by the observation that ADV, BC, and † all consistently reach near-optimal performance when i is very small (almost all states look identical so exploration can be of little help) and when i is quite large (the agent can see nearly the whole environment so there is no need to explore). While we do not expect this trend to hold in all cases, indeed there are likely many cases where pure-IL is more effective than ADV in the no-imitation-gap setting, it is encouraging to see that ADV can bring benefits even when there is no imitation gap.

A.7 Additional baseline details

A.7.1 Baselines details for 2D-LH, PD, and MINIGRID tasks

In Tab. A.2, we include details about the baselines considered in this work, including – purely RL (1), purely IL (2 – 4, 9), a sequential combination of IL/RL (6 – 8), static combinations of IL/RL (5, 10), a method that uses expert demonstrations to generate rewards for reward-based RL (*i.e.* GAIL, 11), and our dynamic combinations (12 – 15). Our imitation learning baselines include those which learn from both expert policy (*i.e.* an expert action is assumed available for any state) and expert demonstrations (offline dataset of pre-collected trajectories).

In our study of hyperparameter robustness (using the PD and MINIGRID tasks) the hyperparameters (hps) we consider for optimization have been chosen as those which, in preliminary experiments, had a substantial impact on model performance. This includes the learning rate (lr), portion of the training steps devoted to the first stage in methods with two stages (stage-split), and the temperature parameter in the ADVISOR weight function (α).¹⁰ Note that, the random environment seed also acts as an implicit hyperparameter. We sample hyperparameters uniformly at random from various sets. In particular, we sample lr from $[10^{-4}, 0.5)$ on a log-scale, stage-split from $[0.1, 0.9)$, and α from $\{4, 8, 16, 32\}$.

In the below we give additional details regarding the GAIL and $\text{ADV}^{\text{demo}} + \text{PPO}$ methods.

Generative adversarial imitation learning (GAIL). For a comprehensive overview of GAIL, please see [25]. Our implementation closely follows that of Ilya Kostrikov [36]. We found GAIL to be quite unstable without adopting several critical implementation details. In particular, we found it critical to (1) normalize rewards using a (momentum-based) running average of the standard deviation of past returns and (2) provide an extensive “warmup” period in which the discriminator network is pretrained. Because of the necessity of this “warmup period”, our GAIL baseline observes more

¹⁰See Sec. 3.2 for definition of the weight function for ADVISOR.

¹¹While implemented with supervision from expert policy, due to the teacher forcing being set to 1.0, this method can never explore beyond states (and supervision) in expert demonstrations.

#	Method	IL/RL	Expert supervision	Hps. searched
1	PPO	RL	Policy	lr
2	BC	IL	Policy	lr
3	$\dagger$	IL	Policy	lr, stage-split
4	$BC^{tf=1}$	IL	Policy ¹¹	lr
5	BC + PPO	IL&RL	Policy	lr
6	$BC \rightarrow PPO$	IL $\rightarrow$ RL	Policy	lr, stage-split
7	$\dagger \rightarrow PPO$	IL $\rightarrow$ RL	Policy	lr, stage-split
8	$BC^{tf=1} \rightarrow PPO$	IL $\rightarrow$ RL	Policy	lr, stage-split
9	BC^{demo}	IL	Demonstrations	lr
10	$BC^{demo} + PPO$	IL&RL	Demonstrations	lr
11	GAIL	IL&RL	Demonstrations	lr
12	ADV	IL&RL	Policy	lr, α
13	$\dagger \rightarrow ADV$	IL&RL	Policy	lr, α , stage-split
14	$BC^{tf=1} \rightarrow ADV$	IL&RL	Policy	lr, α , stage-split
15	$BC^{demo} + ADV$	IL&RL	Demonstrations	lr, α

Table A.2: **Baseline details.** IL/RL: Nature of learning, Expert supervision: the type of expert supervision leveraged by each method, Hps. searched: hps. that were randomly searched over, fairly done with the same budget (see Sec. A.9 for details).

expert supervision and is given a budget of substantially more gradient steps than all other methods. Because of this, our comparison against GAIL *disadvantages our ADVISOR method*. Despite this disadvantage, ADVISOR still outperforms.

The $ADV^{demo} + PPO$ method. As described in the main paper, the $ADV^{demo} + PPO$ method attempts to bring the benefits of our ADVISOR methodology to the setting where expert demonstrations are available but an expert policy (*i.e.*, an expert that can be evaluated at arbitrary states) is not. Attempting to compute the ADVISOR loss (recall Eq. (2)) on off-policy demonstrations is complicated however, as our RL loss assumes access to on-policy demonstrations. In theory, importance sampling methods, see, *e.g.*, [43], can be used to “reinterpret” expert demonstrations as though they were on-policy. But such methods are known to be somewhat unstable, non-trivial to implement, and may require information about the expert policy that we do not have access to. For these reasons, we choose to use a simple solution: when computing the ADVISOR loss on expert demonstrations we ignore the RL loss. Thus $ADV^{demo} + PPO$ works by looping between two phases:

- Collect an (on-policy) rollout using the agent’s policy, compute the PPO loss for this rollout and perform gradient descent on this loss to update the parameters.
- Sample a rollout from the expert demonstrations and, using this rollout, compute the demonstration-based ADVISOR loss

$$\mathcal{L}^{ADV-demo}(\theta) = \mathbb{E}_{demos.}[w(S) \cdot CE(\pi^{teach}(S), \pi_f(S; \theta))], \quad (10)$$

and perform gradient descent on this loss to update the parameters.

A.7.2 Baselines used in POINTNAV experiments

Our POINTNAV baselines are described in Appendix A.5.9. See also Table A.4.

A.7.3 Baselines details for OBJECTNAV experiments

Our OBJECTNAV baselines are described in Appendix A.5.8. See also Table A.4.

A.7.4 Baselines used in COOPNAV experiments

Our COOPNAV baselines are described in Appendix A.5.9. We follow the implementation of [40].

A.8 Architecture Details

2D-LH model. As discussed in Sec. A.5.10, we have designed the observation given to our agent so that a simple linear layer followed by a soft-max function is sufficient to capture any f -restricted policy. As such, our main and auxiliary actor models for this task are simply linear functions mapping the input 6400-dimensional observation to a 4-dimensional output vector followed by a soft-max non-linearity. The critic is computed similarly but with a 1-dimensional output and no non-linearity.

PD model. Our PD model has three sequential components. The first embedding layer maps a given observation, a value in $\{0, 1, 2, 3\}$, to an 128-dimensional embedding. This 128-dimensional vector is then fed into a 1-layer LSTM (with a 128-dimensional hidden state) to produce an 128-output representation h . We then compute our main actor policy by applying a 128×7 linear layer followed by a soft-max non-linearity. The auxiliary actor is produced similarly but with separate parameters in its linear layer. Finally the critic’s value is generated by applying a 128×1 linear layer to h .

MINIGRID model. Here we detail each component of the model architecture illustrated in Fig. 3. The encoder (‘Enc.’) converts observation tensors (integer tensor of shape $7 \times 7 \times 3$) to a corresponding embedding tensor via three embedding sets (of length 8) corresponding to type, color, and state of the object. The observation tensor, which represents the ‘lights-out’ condition, has a unique (*i.e.*, different from the ones listed by [10]) type, color and state. This prevents any type, color or state from having more than one connotation. The output of the encoder is of size $7 \times 7 \times 24$. This tensor is flattened and fed into a (single-layered) LSTM with a 128-dimensional hidden space. The output of the LSTM is fed to the main actor, auxiliary actor, and the critic. All of these are single linear layers with output size of $|\mathcal{A}|$, $|\mathcal{A}|$ and 1, respectively (main and auxiliary actors are followed by soft-max non-linearities).

POINTNAV, OBJECTNAV, and COOPNAV model.

For the POINTNAV [58], OBJECTNAV [15], and COOPNAV [40] tasks, we (for fair comparison) use model architectures from prior work. For use with ADVISOR, these model architectures require an additional auxiliary policy head. We define this auxiliary policy head as a linear layer applied to the model’s final hidden representation followed by a softmax non-linearity.

A.9 Fair Hyperparameter Tuning

As discussed in the main paper, we attempt to ensuring that comparisons to baselines are fair. In particular, we hope to avoid introducing misleading bias in our results by extensively tuning the hyperparameters (hps) of our ADVISOR methodology while leaving other methods relatively un-tuned.

2D-LH: Tune by Tuning a Competing Method. The goal of our experiments with the 2D-LH environment are, principally, to highlight that increasing the imitation gap can have a substantial detrimental impact on the quality of policies learned by training IL. Because of this, we wish to give IL the greatest opportunity to succeed and thus we are not, as in our PD/MINIGRID experiments, attempting to understand its expected performance when we must search for good hyperparameters. To this end, we perform the following procedure for every $i, j \in \{1, 3, 5, \dots, 15\}$ with $i < j$.

For every learning rate $\lambda \in \{100 \text{ values evenly spaced in } [10^{-4}, 1] \text{ on a log-scale}\}$ we train a f^i -restricted policy to imitate a f^j -optimal teacher using BC. For each such trained policy, we roll out trajectories from the policy across 200 randomly sampled episodes (in the 2D-LH there is no distinction between training, validation, and test episodes as there are only four unique initial world settings). For each rollout, we compute the average cross entropy between the learned policy and the expert’s policy at every step. A “best” learning rate $\lambda^{i,j}$ is then chosen by selecting the learning rate resulting in the smallest cross entropy (after having smoothed the results with a locally-linear regression model [75]).

A final learning rate is then chosen as the average of the $\lambda^{i,j}$ and this learning rate is then used when training all methods to produce the plots in Fig. 6. As some baselines require additional hyperparameter choices, these other hyperparameters were chosen heuristically (post-hoc experiments suggest that results for the other methods are fairly robust to these other hyperparameters).

PD and MINIGRID tasks: Random Hyperparameter Evaluations. As described in the main paper, we follow the best practices suggested by Dodge et al. [16]. In particular, for our PD and

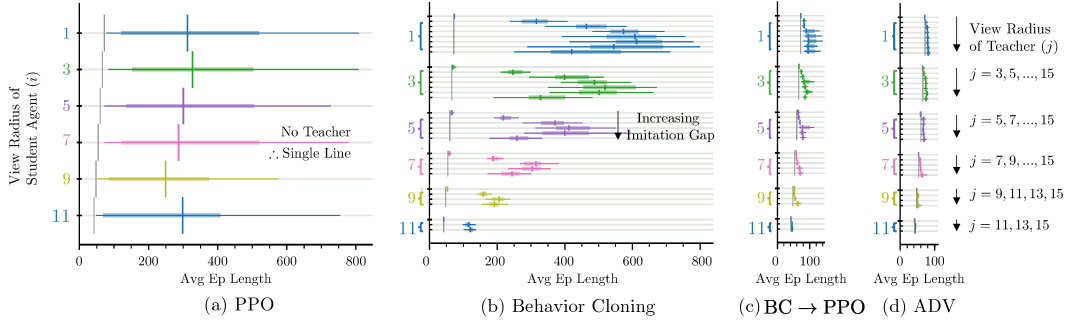


Figure A.3: **Size of the imitation gap directly impacts performance.** Training f^i -restricted students with f^j -optimal teachers (in 2D-LH).

MINIGRID tasks we train each of our baselines when sampling that method’s hyperparameters, see Table A.2 and recall Sec. A.7, at random 50 times. Our plots, *e.g.*, Fig. 5, then report an estimate of the expected (validation set) performance of each of our methods when choosing the best performing model from a fixed number of random hyperparameter evaluations. Unlike [16], we compute this estimate using a U-statistic [68, Chapter 12] which is unbiased. Shaded regions encapsulate the 25-to-75th quantiles of the bootstrap distribution of this statistic.

POINTNAV, OBJECTNAV, and COOPNAV tasks: use hyperparameters from in prior work. Due to computational constraints, our strategy for choosing hyperparameters for the POINTNAV, OBJECTNAV, and COOPNAV tasks was simply to follow prior work whenever possible. Of course, there was no prior work suggesting good hyperparameter values for the α, β parameters in our new ADVISOR loss. Following the intuitions we gained from our the 2D-LH, PD, and MINIGRID experiments, we fixed α, β to (10, 0.1) for POINTNAV, α, β to (20, 0.1) for OBJECTNAV, and α, β to (0.01, 0) for COOPNAV. For the OBJECTNAV task, we experimented with setting $\beta = 0$ and found that the change had essentially no impact on performance (validation-set SPL after ≈ 100 Mn training steps actually improved slightly from .1482 to .1499 when setting $\beta = 0$).

A.10 Training Implementation

As discussed previously, for our POINTNAV, OBJECTNAV, and COOPNAV experiments, we have used standard training implementation details (*e.g.* reward structure) from prior work. Thus, in the below, we provide additional details only for the 2D-LH, PD, and MINIGRID tasks.

A summary of the training hyperparameters and their values is included in Tab. A.3. Kindly see [62] for details on PPO and [61] for details on generalized advantage estimation (GAE).

Max. steps per episode. The maximum number of steps allowed in the 2D-LH task is 1000. Within the PD task, an agent can never take more than 11 steps in a single episode (1 action to select the door and then, at most, 10 more actions to input the combination if d_1 was selected) and thus we do not need to set a maximum number of allowed steps. The maximum steps allowed for an episode of WC/LC is set by [10, 9] to $4S^2$, where S is the grid size. We share the same limits for the challenging variants – SWITCH and CORRUPT. Details of task variants, their grid size, and number of obstacles are included in Sec. A.5.

Reward structure. Within the 2D-LH task, the agent receives one of three possible rewards after every step: when the agent finds the goal it receives a reward of 0.99, if it otherwise has reached the maximum number of steps (1000) it receives a -1 reward, and otherwise, if neither of the prior cases hold, it obtains a reward of -0.01 . See Sec. A.5.1 for a description of rewards for the PD task. For WC/LC, [10, 9] configure the environment to give a 0 reward unless the goal is reached. If the goal is reached, the reward is $1 - \frac{\text{episode length}}{\text{maximum steps}}$. We adopt the same reward structure for our SWITCH and CORRUPT variants as well.

Computing infrastructure. As mentioned in Sec. 4.3, for all tasks (except LH) we train 50 models (with randomly sampled hps) for each baseline. This amounts to 750 models per task or 6700 models in total. For each task, we utilize a g4dn.12xlarge instance on AWS consisting of 4 NVIDIA T4

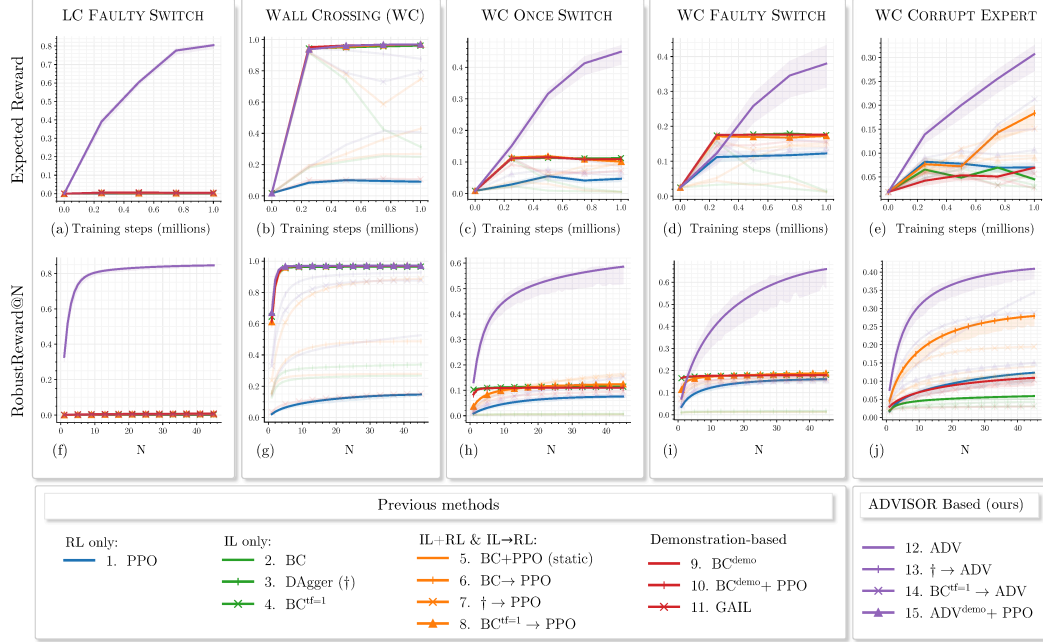


Figure A.4: **Additional results for MINIGRID tasks.** Here we include the results on the MINIGRID tasks missing from Figure 5.

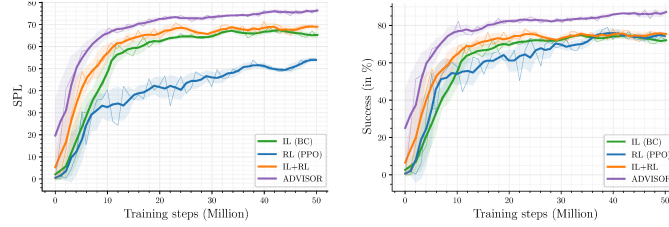


Figure A.5: **Learning curves for POINTNAV.** SPL (scaled by 100) and success rate (in %) are plotted vs. training steps, following the standard protocols. We evaluate checkpoints after every 1024k frames of experience. This is plotted as the thin line. The thick line and shading depicts the rolling mean (with a window size of 2) and corresponding standard deviation.

GPUs and 48 CPUs. We run through a queue of 750 models using ≈ 40 processes. For tasks set in the MINIGRID environments, models each require ≈ 1.2 GB GPU memory and all training completes in 18 to 36 hours. For the PD task, model memory footprints are smaller and training all models is significantly faster (< 8 hours).

A.11 Additional results

Here we record additional results that were summarized or deferred in Section 4.4. In particular,

- Figure A.3 complements Figure 6 from the main paper and provides results for additional baselines on the 2D-LH task. Notice that both the pipelined IL $\rightarrow$ PPO and ADVISOR methods greatly reduce the impact of the imitation gap (Figures A.3c and A.3d versus Figure A.3b) but our ADVISOR method is considerably more effective in doing so (Figure A.3c v.s. Figure A.3d).
- Figure A.4 shows the results on our remaining MINIGRID tasks missing from Figure 5. Notice that the trends here are very similar to those from Figure 5, ADVISOR-based methods have similar or better performance than our other baselines.

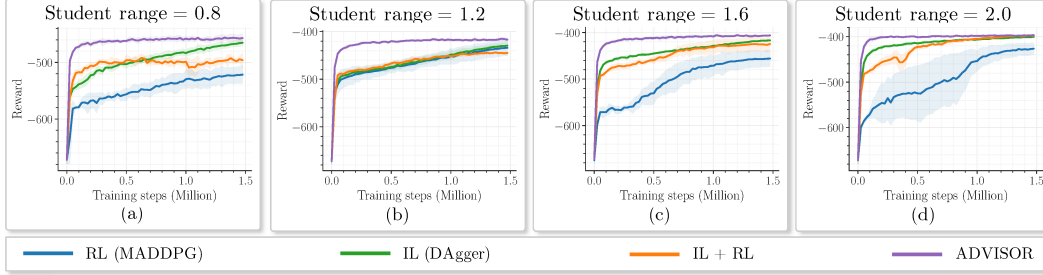


Figure A.6: **Learning curves for COOPNAV.** Rewards are plotted vs. training steps, following the standard protocols. For a full-range teacher, we train students with different (and limited) visibility range of 0.8, 1.2, 1.8, and 2.0. The networks are initialized with four different seeds and the mean and standard deviation are plotted. Checkpoints are evaluated at every 25k steps.

Hyperparameter	Value
<i>Structural</i>	
Cell type embedding length	8
Cell color embedding length	8
Cell state embedding length	8
RNN type	LSTM
RNN layers	1
RNN hidden size	128
# Layers in critic	1
# Layers in actor	1
<i>PPO</i>	
Clip parameter (ϵ) [62]	0.1
Decay on ϵ	Linear(1, 0)
Value loss coefficient	0.5
Discount factor (γ)	0.99
GAE parameter (λ)	1.0
<i>Training</i>	
Rollout timesteps	100
Rollouts per batch	10
# processes sampling rollouts	20
Epochs	4
Optimizer	Adam [33]
(β_1, β_2) for Adam	(0.9, 0.999)
Learning rate	searched
Gradient clip norm	0.5
Training steps (WC/LC & variants)	$1 \cdot 10^6$
Training steps (2D-LH & PD)	$3 \cdot 10^5$

Table A.3: Structural and training hyperparameters for 2D-LH, PD, and MINIGRID tasks.

- Table A.5 shows an extended version of Table 1 where, rather than grouping methods together, we display results for each method individually.
- Figure A.5 displays validation set performance of our POINTNAV baselines over training. Note that static combination of RL and IL losses improves individual RL/IL baselines. Our adaptive combination of these losses (ADVISOR) outperforms these baselines and is more sample efficient.
- Figure A.6 lays out the performance of agents on the COOPNAV task. In the main paper we include results for the limited visibility range of 1.6 for the student. Here, we include results for four visibility range. RL only baseline is least sample-efficient. Overall, we find ADVISOR is significantly more sample efficient – most of the learning is completed in just 0.2 million steps while the other baselines take over 1.5 million steps.

Hyperparameter	POINTNAV	OBJECTNAV
<i>Structural</i>		
RNN type		GRU
RNN layers		1
RNN hidden size		512
# Layers in critic		1
# Layers in actor		1
<i>PPO</i>		
Clip parameter (ϵ) [62]		0.1
Decay on ϵ		None
Value loss coefficient		0.5
Discount factor (γ)		0.99
GAE parameter (λ)		0.95
<i>Training</i>		
Rollout timesteps		128
Rollouts per batch	60	8
# processes sampling rollouts	60	16
Epochs		4
Optimizer		Adam [33]
(β_1, β_2) for Adam		(0.9, 0.999)
Learning rate	$3 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$
Gradient clip norm	0.5	0.1
Training steps	$100 \cdot 10^6$	$50 \cdot 10^6$

Table A.4: Structural and training hyperparameters for POINTNAV and OBJECTNAV.

Tasks → Training routines ↓	PD	LAVACROSSING				WALLCROSSING			
	-	Base Ver.	Corrupt Exp.	Faulty Switch	Once Switch	Base Ver.	Corrupt Exp.	Faulty Switch	Once Switch
PPO	0	0	0	0.01	0	0.09	0.07	0.12	0.05
BC	-0.6	0.1	0.02	0	0	0.25	0.05	0.01	0.01
Dagger (†)	-0.59	0.14	0.02	0	0	0.31	0.03	0.01	0.01
BC ^{tf=1}	-0.62	0.88	0.02	0.02	0	0.96	0.03	0.17	0.11
BC+PPO (static)	-0.59	0.12	0.08	0	0	0.27	0.09	0.01	0
BC → PPO	-0.17	0.15	0.32	0.02	0	0.43	0.18	0.14	0.09
† → PPO	-0.45	0.32	0.61	0.02	0	0.75	0.15	0.15	0.1
BC ^{tf=1} → PPO	-0.5	0.94	0.74	0.04	0	0.97	0.09	0.17	0.1
BC ^{demo}	-0.62	0.88	0.02	0.02	0	0.96	0.07	0.18	0.11
BC ^{demo} + PPO	-0.64	0.96	0.2	0.02	0	0.97	0.03	0.17	0.11
GAIL	-0.09	0	0	0.02	0	0.11	0.06	0.16	0.07
ADV	1	0.18	0.8	0.77	0.8	0.41	0.31	0.38	0.45
BC ^{tf=1} → ADV	-0.13	0.55	0.83	0.02	0	0.88	0.15	0.15	0.09
† → ADV	-0.1	0.47	0.73	0.01	0	0.79	0.21	0.13	0.07
ADV ^{demo} + PPO	0	0.96	0.94	0.03	0	0.97	0.11	0.14	0.06

Table A.5: Expected rewards for the POISONEDDOORS task and MINIGRID tasks. Here we show an expanded version of Table A.5 where results for all methods rather than grouped methods. For each of our 15 training routines we report the expected maximum validation set performance (when given a budget of 10 random hyperparameter evaluations) after training for $\approx 300k$ steps in the POISONEDDOORS environment and $\approx 1Mn$ steps in our 8 MINIGRID tasks. The maximum possible reward is 1 for the MINIGRID tasks.