BRIDGING THE IMITATION GAP BY ADAPTIVE INSUBORDINATION

Luca Weihs[*,1], **Unnat Jain**[*,2,†], **Jordi Salvador**[1], **Svetlana Lazebnik**[2]
Aniruddha Kembhavi[1], **Alexander Schwing**[2]
[1]Allen Institute for AI, [2]University of Illinois at Urbana-Champaign

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

ABSTRACT

When expert supervision is available, practitioners often use imitation learning with varying degrees of success. We show that when an expert has access to privileged information that is unavailable to the student, this information is marginalized in the student policy 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 skills. To better address these tasks and alleviate the imitation gap we propose 'Adaptive Insubordination' (ADVISOR), which dynamically weights imitation and reward-based reinforcement learning losses during training, enabling switching between imitation and exploration. On a suite of challenging didactic and MINIGRID tasks, we show that 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 succeeds in complex tasks with sparse rewards (Chevalier-Boisvert et al., 2018a; Peng et al., 2018; Nair et al., 2018), and when the observations are high-dimensional, *e.g.*, in visual 3D environments (Kolve et al., 2019; Savva et al., 2019). In such tasks, obtaining a high quality policy purely from reward-based RL is often challenging, requiring extensive reward shaping and careful tuning as reward variance remains high. In contrast, IL leverages an expert which is generally less impacted by the environment's random state. However, designing an expert often relies on privileged information that is unavailable at inference time. For instance, it is straightforward to create a navigational expert when privileged with access to a connectivity graph of the environment (using shortest-path algorithms) (*e.g.*, Gupta et al., 2017b) or an instruction-following expert which leverages an available semantic map (*e.g.*, Shridhar et al., 2020; Das et al., 2018b). Similarly, game experts may have the privilege of seeing rollouts (Silver et al., 2016) or vision-based driving experts may have access to ground-truth layout (Chen et al., 2020b). Such graphs, maps, rollouts, or layouts aren't available to the student or at inference time.

How does use of a privileged expert influence the student policy? We show that training an agent to imitate such an expert results in a policy which marginalizes out the privileged information. This can result in a student policy which is sub-optimal, and even near-uniform, over a large collection of states. We call this discrepancy between the expert policy and the student policy the *imitation gap*. To overcome this imitation gap, prior work often uses stage-wise training: IL is used to 'warm start' learning and subsequent reward-based RL algorithms, such as proximal policy optimization (PPO), are used for fine-tuning (Lowe et al., 2020). While this strategy is often successful, the following example shows that it can fail dramatically.

Example 1 (Poisoned Doors)**.** Suppose an agent is presented with $N \geq 3$ doors $d_1, \ldots, d_N$. As illustrated in Fig. 1 (for $N = 4$), opening d_1 requires entering an unknown but fixed code of length M. Successful code entry results in a guaranteed reward of 1, otherwise the reward is 0. Since the

[*]denotes equal contribution by LW and UJ; [†] work partially done as an intern at Allen Institute for AI

code is unknown to the agent, it would have to learn the code. 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 for all $i \in \{2, \ldots, N\} \setminus j$ the reward behind d_i is -2. Without knowledge of 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 open a door among $d_2, \ldots, d_N$ uniformly at random obtaining an expected return of $-2 \cdot (N-3)/(N-1)$. Training with reward-based RL after this 'warm start' 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 bridge the imitation gap, we introduce **Ad**aptive **I**nsub**or**dination (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 expert'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 expert's action with high confidence and is otherwise small. As we show empirically, ADVISOR combines the benefits of IL and RL while avoiding the pitfalls of either method alone.

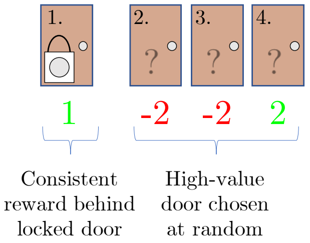

1.
2.
3.
4.
?
?
?
1
-2
-2
2
Consistent reward behind locked door
High-value door chosen at random

Figure 1: PoisonedDoors

We evaluate the benefits of employing ADVISOR across ten tasks including the Poisoned Doors discussed above, a 2D gridworld, and a suite of tasks based on the MiniGrid environment (Chevalier-Boisvert et al., 2018a;b). Across all tasks, ADVISOR outperforms popular IL and RL baselines as well as combinations of these methods. We also demonstrate that ADVISOR can learn to ignore corruption in expert supervision. ADVISOR can be easily incorporated into existing RL pipelines. The code to do the same is included in the supplement and will be made publicly available.

2 Related Work

A series of solutions (*e.g.*, Mnih et al., 2015; van Hasselt et al., 2016; Bellemare et al., 2016; Schaul et al., 2016) have made off-policy deep Q-learning methods stable for complex environments like Atari Games. These advances have been further improved upon through policy gradient methods (Schulman et al., 2015a; Mnih et al., 2016; Levine et al., 2016; Wang et al., 2017; Silver et al., 2016). Particularly, Trust Region Policy Optimization (TRPO) (Schulman et al., 2015a) improves sample-efficiency by safely integrating larger gradient steps, but is incompatible with architectures with shared parameters between policy and value approximators. Proximal Policy Optimization (PPO) (Schulman et al., 2017) employs a clipped variant of TRPO's surrogate objective and is widely adopted in the deep RL community. We also use it 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, leading to methods that imitate expert information (Subramanian et al., 2016). A simple approach to imitation learning (IL) is Behaviour Cloning (BC), a supervised classification loss between the policy of the learner and expert agents (Sammut et al., 1992; Bain & Sammut, 1995). BC suffers from compounding of errors due to covariate shift, namely if the learning agent makes a single mistake at inference time then it can rapidly enter settings where it has never received relevant supervision and thus fails (Ross & Bagnell, 2010). Data Aggregation (DAgger) (Ross et al., 2011) is the go-to online sampling framework that trains a sequence of learner policies by querying the expert at states beyond those that would be reached by following only expert actions. IL is further enhanced, *e.g.*, via hierarchies (Le et al., 2018), by improving over the expert (Chang et al., 2015), bypassing any intermediate reward function inference (Ho & Ermon, 2016), and/or learning from experts that differ from the learner (Gupta et al., 2017a; Jiang, 2019; Gangwani & Peng, 2020). 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 (Silver et al., 2016), robotic and motor skills (Pomerleau, 1991; Kober & Peters, 2009; Peters & Schaal, 2008; Rajeswaran et al., 2018), navigation in visually realistic environments (Gupta et al., 2017b; Das et al., 2018a; Jain et al., 2019), and web & language based tasks (He et al., 2016; Das et al., 2017; Shi et al., 2017; Wang et al., 2018).

Recent methods mix expert demonstrations with the agent's own roll-outs instead of using a sequential combination of IL followed by RL. DQfD (Hester et al., 2018) initializes the replay buffer with expert episodes and adds roll-outs of (a pretrained) agent. They weight experiences based on the previous temporal difference errors (Schaul et al., 2016) and use a supervised loss to learn from the expert. For continuous action spaces, DDPGfD (Vecerík et al., 2017) is an analogous incorporation of IL into DDPG (Lillicrap et al., 2016). POfD (Kang et al., 2018) improves performance 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).

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.

3 ADVISOR

We first introduce notation to define the imitation gap and illustrate how it leads to 'policy averaging.' Next, using the construct of an auxiliary policy, we propose ADVISOR to bridge this gap. Finally, we show how to estimate the auxiliary policy in practice using deep nets.

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, 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 (Mnih et al., 2015; 2016), 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* $f : \mathcal{S} \to \mathcal{O}_f$ 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-partial policy* and the set of feasible f-partial 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-partial policy is optimal among all other f-partial policies, we say it is *f-optimal*. We call $o \in \mathcal{O}_f$ a *partial-observation* and for any f-partial 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 an expert policy which is able to successfully complete the task $\mathcal{T}$. This expert policy may have access to the whole environment state and thus may be optimal among all policies. Alternatively, the expert policy may, like the student, only make decisions given partial information (*e.g.*, a human who sees exactly the same inputs as the student). For flexibility we will define the expert policy as $\pi^{\text{exp}}_{f^{\text{exp}}}$, denoting it is a f^{exp}-partial policy for some filtration function f^{exp}. For simplicity, we will assume that $\pi^{\text{exp}}_{f^{\text{exp}}}$ is f^{exp}-optimal. Subsequently, we will drop the subscript f^{exp} unless we wish to explicitly discuss multiple experts simultaneously.

In IL (Osa et al., 2018; Ross et al., 2011), π_f is trained to mimic π^{exp} by minimizing the (expected) cross-entropy between π_f and π^{exp} over a set of sampled states $s \in \mathcal{S}$:

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

where $CE(\pi^{\text{exp}}, \pi_f)(S) = -\pi^{\text{exp}}(S) \odot \log \pi_f(S)$, $\odot$ denotes the usual dot-product, and S is a random variable taking value $s \in \mathcal{S}$ with probability measure $\mu : \mathcal{S} \to [0, 1]$. Often $\mu(s)$ is chosen to

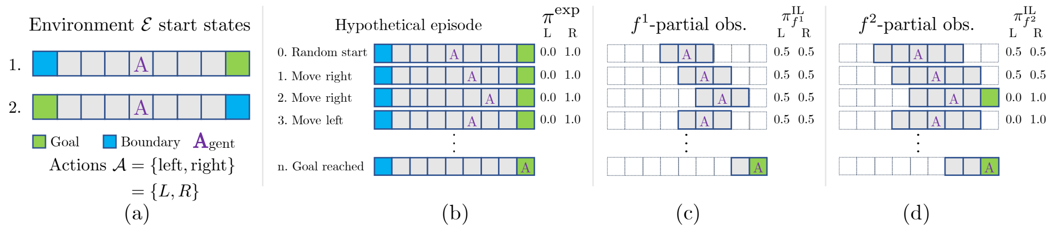


Figure 2: **Effect of partial observability in a 1-dimensional gridworld environment.** (a) The two start states and actions space for 1D-Lighthouse with $N = 4$. (b) A trajectory of the agent following a hypothetical random policy. At every trajectory step we display output probabilities as per the shortest-path expert (π^{exp}) 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 $\pi^{\text{IL}}_{f^1}, \pi^{\text{IL}}_{f^2}$, learned by imitating π^{exp}, show that imitation results in sub-optimal policies *i.e.* $\pi^{\text{IL}}_{f^1}, \pi^{\text{IL}}_{f^2} \neq \pi^{\text{exp}}$.

equal the frequency with which an exploration policy (*e.g.*, random actions or π^{exp}) visits state s in a randomly initialized episode. When it exists, we denote the policy minimizing Eq. (1) as $\pi_f^{\mu,\pi^{\text{exp}}}$. When μ and π^{exp} are unambiguous, we write $\pi^{\text{IL}}_f = \pi_f^{\mu,\pi^{\text{exp}}}$.

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

Example 2 (1D-Lighthouse)**.** We illustrate imitation gap using a gridworld spanning $[-N, \dots, N]$. The two start states correspond to the goal at $-N$ or N, while the agent is always initialized at 0 (see Fig. 2a). Clearly, given access to the full state, π^{exp} would map states to an 'always left' or 'always right' probability distribution if the goal is on the left or right, respectively. We now consider an intuitive family of filtration functions f^i which restrict the agent's visibility to i steps on either side of its current location, though it doesn't affect the agent's memory of its previous actions. It is straightforward to show that an agent following the f^1-optimal policy will begin to move deterministically towards any corner, w.l.o.g. assume right. Depending on whether the goal is visible from location $N-1$ the agent either continues towards it, or turns left and deterministically repeats the same action (left) until the goal (at $-N$) is reached. Similarly, an agent following an f^i-optimal policy goes to location $N-i$, finishing the episode in an expected $0.5 \cdot N + 0.5 \cdot (3N - 2i)$ steps. However, what is $\pi^{\text{IL}}_{f^i}$, *i.e.*, the best f^i-partial policy that can be learnt by imitating π^{exp}? It is straightforward to show that an agent following policy $\pi^{\text{IL}}_{f^i}$ will take left (and right) 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 in half of these events the expert will instruct the agent to go right and in the other half to go left. The cross-entropy loss will cause $\pi^{\text{IL}}_{f^i}$ to be uniform in all such states. We defer a rigorous treatment of this example to the Appendix A.1. Furthermore, in Sec. 4 and Fig. 5, we train f^i-partial policies with f^j-optimal experts for a 2D variant of this example. We empirically verify that a student learns a better policy while imitating teachers with a similar filtration than "more intelligent" ones.

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^{exp} 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 $\Pi^{feas.}$ contains all f-partial policies. Then, for any $s \in \mathcal{S}$ with $o = f(s)$, we have that $\pi^{IL}_f(o) = \mathbb{E}_\mu[\pi^{exp}(S) \mid f(S) = o]$.*

Proofs are deferred to 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) (Lowe et al., 2020; Das et al., 2018b). Obviously training first with IL and then via pure RL techniques 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 expert policy only in settings in which the expert policy can, in principle, be exactly reproduced by the student. Otherwise the student should learn via 'standard' RL methods. To this end, 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 to imitate an expert 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.[1] This loss formulation is general and spans all policy gradient methods, including A2C and PPO. The imitation loss is the standard cross-entropy loss $\mathbb{E}_\mu[CE(\pi^{\text{exp}}(S), \pi_f(S;\theta))]$. Concretely, ADVISOR loss is:

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

Our goal is to find a *weight function* $w : \mathcal{S} \times \Theta \to [0, 1]$ 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^{exp} and π_f^{IL} at a state s and a mapping from this distance to weights.

We now define $d^0(\pi, \pi_f)(s)$, a distance estimate between a policy π and an f-partial 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.*, the KL-divergence (which we use in our experiments). 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.[2] 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 $d^0(\pi^{\text{exp}}, \pi_f^{\text{IL}})(s)$.

Unfortunately it is, in general, impossible to compute $d^0(\pi^{\text{exp}}, \pi_f^{\text{IL}})(s)$ 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 will define in the next section.

Given π_f^{aux} we obtain the estimator $d^0(\pi^{\text{exp}}, \pi_f^{\text{aux}})$ of $d^0(\pi^{\text{exp}}, \pi_f^{\text{IL}})$. Additionally, we make use of the monotonically decreasing function $m_{\alpha,\beta} : \mathbb{R}_{\geq 0} \to [0, 1]$, where $\alpha, \beta \geq 0$. We define our weight function $w(s)$ for $s \in \mathcal{S}$ as:

$$w(s) = m_{\alpha,\beta}(d^0(\pi^{\text{exp}}, \pi_f^{\text{aux}})(s)) \quad \text{with} \tag{3}$$

$$m_{\alpha,\beta}(x) = e^{-\alpha x} \cdot 1_{[x \leq \beta]} \tag{4}$$

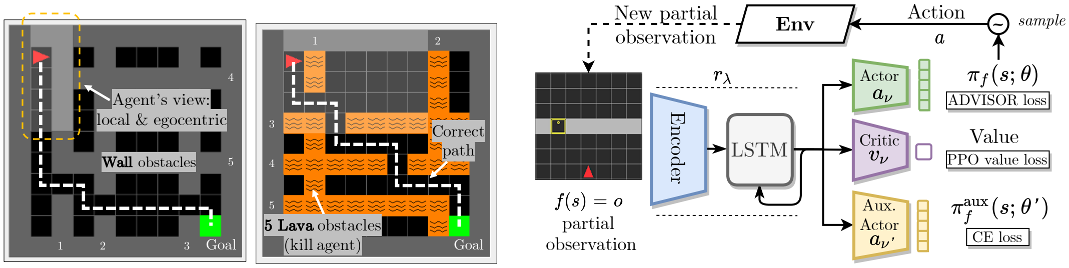
Together Eq. 2, 3, 4 define ADVISOR.

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 $d^0(\pi^{\text{exp}}, \pi_f^{\text{IL}})(s)$ using the plug-in estimator $d^0(\pi^{\text{exp}}, \pi_f^{\text{aux}})(s)$. While plug-in estimators are intuitive and simple to define, they need not be

[1] For readability, we implicitly make three key simplifications. First, computing the expectation $\mathbb{E}_\mu[\ldots]$ 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.

[2] Measures 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.*, $\mathbb{E}_\mu[d(\pi(S), \pi_f(S)) \mid f(S) = o]$.



(a) WallCrossing (b) LavaCrossing (c) Model overview

Figure 3: **MiniGrid base tasks and model overview.** (a) WC: Navigation with wall obstacles, with additional expert and environmental challenges. We test up-to 25×25 grids with 10 walls. (b) LC: Safe navigation, avoiding lethal lava rivers. We test up-to 15×15 grids with 10 lava rivers. (c) An auxiliary actor is added and trained only using IL. The 'main' actor policy is trained using the ADVISOR loss defined in Section 3.2, 3.3.

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. 3c we provide an overview of how we compute the estimator π_f^{aux} via deep nets. As is common practice (Mnih et al., 2016; Heess et al., 2017; Jaderberg et al., 2017; Pathak et al., 2017; Mirowski et al., 2017; Chevalier-Boisvert et al., 2018a; Chen et al., 2020a; Jain et al., 2020; Weihs et al., 2020), 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. Instead, 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.*, $\pi_f^{\text{aux}} = a_{\nu'} \circ r_\lambda$. As both actors share the representation r_λ they benefit from any mutual learning. While our instantiation covers most use-cases, ADVISOR can be extended to estimating π_f^{IL} via two separate networks, *i.e.*, $\theta' = (\nu', \lambda')$.

4 Experiments

We rigorously compare ADVISOR to IL methods, RL methods and their popularly-adopted (yet ad hoc) combinations. In particular, we evaluate 14 methods. We do this over ten tasks – realizations of Ex. 1 & Ex. 2, and eight navigational tasks of varying complexity within the fast, versatile MiniGrid environment (Chevalier-Boisvert et al., 2018a;b). Furthermore, for robustness, we train 50 hyperparameter variants for complex tasks. For all tasks, we find ADVISOR-based methods outperform or match performance of all baselines.

4.1 Tasks

Succinct descriptions of our tasks follow. We defer details and description of experts to Appendix A.5.
PoisonedDoors (PD). As defined in Ex. 1 in Sec. 1 with $N = 4$, $M = 10$, see Fig. 1.
WallCrossing/LavaCrossing (WC/LC). As illustrated in Fig. 3a, 3b, an agent is tasked to navigate to the goal using local observations. In doing so, it must avoid walls or deadly (*i.e.*, episode-ending) rivers of lava. Evidently, imitating a shortest-path expert is easy, requiring no exploration beyond expert-visited states. Hence, we consider more challenging variants of WC and LC tasks.
Switch. The agent is initialized in a WC or LC environment with the "lights turned off." The agent can use an additional *switch* action to get unaffected observations, whereas the shortest-path expert can navigate in the dark. Hence, the expert doesn't supervise taking the new action. For both WC and LC base tasks, we experiment with two switches: (1) lights stay on after using the *switch* action (Once), or (2) light turn on only for a single timestep (Faulty).
Corrupt. To evaluate resilience of methods to a corrupted expert. In every episode the expert produces correct actions until it is within N_C steps of the target, after which it outputs random actions.
2D-Lighthouse (2D-LH). A harder, 2D variant of the gridworld task introduced in Ex. 2.

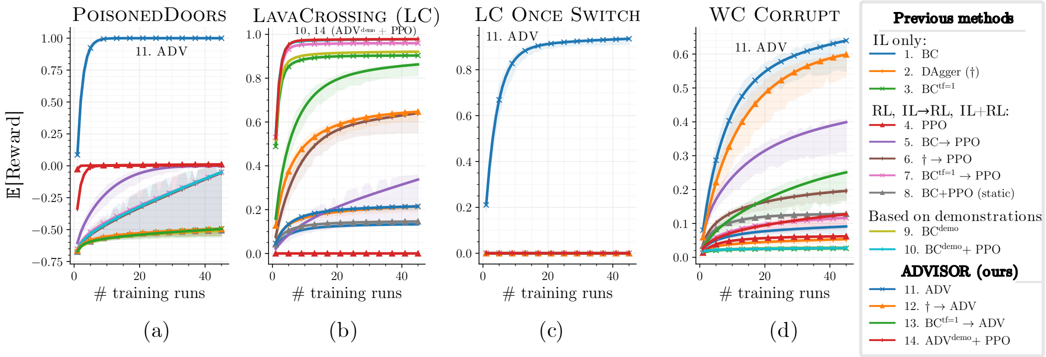


Figure 4: **Evaluation following (Dodge et al., 2019).** As described in Section 4.3, we plot expected validation reward of best-found model (y-axis) over an increasing budget of # training runs, each with random hyperparameter values (x-axis). Clearly, larger $\mathbb{E}$[Reward] with fewer # training runs is better. We mark the best performing method(s) at the top of each plot.

4.2 Baselines and ADVISOR-based Methods

Expert supervision comes in two forms: (a) as an expert policy, or (b) as a dataset of expert demonstrations. We study baselines and ADVISOR in both these forms. For (a), we include IL baselines with different levels of teacher-forcing (tf): tf=0, tf annealed from 1→0, and tf=1. This leads to Behaviour Cloning (BC), Data Aggregation (DAgger, †), and BC$^{\text{tf=1}}$, respectively. Also, we implement pure RL (PPO) which learns only on the sparse rewards. Furthermore, we implement popular sequential hybrids such as BC then PPO (BC→PPO), DAgger then PPO († $\rightarrow$ PPO), BC$^{\text{tf=1}}$ $\rightarrow$ PPO, and a parallel combination of BC $+$ PPO(static). This is a static variant of our adaptive combination ADVISOR (ADV). We introduce hybrids such as DAgger then ADVISOR († $\rightarrow$ ADV), and BC$^{\text{tf=1}}$ $\rightarrow$ ADV. For (b), agents imitate expert demonstrations and hence get no supervision beyond the states in the demonstrations. This leads to BC$^{\text{demo}}$ and its combination with PPO (BC$^{\text{demo}}$ $+$ PPO). We introduce a corresponding ADV$^{\text{demo}}$ $+$ PPO, applying ADVISOR on expert demonstrations while training PPO on on-policy rollouts. Further details of all methods are in Appendix A.6. For fairness, the same model architecture is shared across all methods (recall Fig. 3c, Sec. 3.3). We defer implementation details to Appendix A.7.

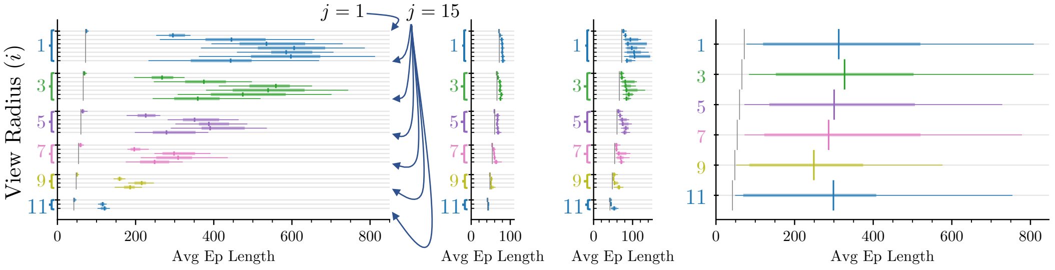
4.3 Evaluation

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. We avoid this by considering two strategies. For PD and all MiniGrid tasks, we follow recent best practices (Dodge et al., 2019).[3] Namely, we tune each method by randomly sampling a fixed number of hps and reporting, for each baseline, an estimate of $\mathbb{E}$[Val. reward of best model when allowed a budget of k random hps] for $1 \leq k \leq 45$. For this we must train 50 models per method, *i.e.*, 700 for each of these nine tasks. More details in Appendix A.8. 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 (details are in Appendix A.9).
Metrics. We record avg. rewards, episode lengths, and success rates. In the following, we report a subset of these recorded values. Additional plots can be found in Appendix A.10.

4.4 Results

Before delving into task-specific analysis, we state three overall trends. First, for tasks where best-performing baselines require querying the expert policy during training, ADV significantly outperforms all methods. These are the more difficult tasks which need exploration, such as PD and Switch. Second, for tasks where the best-performing baselines are demonstrations-based,

[3] See also reproducibility checklist in EMNLP'20 CfP: `https://2020.emnlp.org/call-for-papers`



(a) Behavior Cloning (b) ADV (c) † → PPO (d) PPO

Figure 5: **"Less intelligent" teachers.** Learning f^i-partial policies using f^j-optimal experts 2D-LH.

ADV$^{\text{demo}}$ $+$ PPO improves over or matches previous methods. Third, a head-on comparison of BC $+$ PPO(static) and ADV shows that our dynamic weighting approach is superior across all tasks.

PD. This environment was designed to be adversarial to standard imitation-learning approaches and so it is not surprising, see Fig. 4a, that models trained using standard IL techniques (DAgger, BC, BC$^{\text{tf}=1}$) perform poorly. Qualitatively, they attempt to open door 1 with a low probability and thus obtain an expected reward near $-2/3$. Baselines that learn from rewards, *e.g.*, PPO, can learn to avoid doors 2-4 but in practice cannot learn the combination to open the first door. This results in an average reward of 0. Notice that warm-starting with any form of IL is actively harmful: *e.g.*, it takes many hyperparameter evaluations before we consistently find a DAgger→PPO model that reproduces the performance of a plain PPO model. Finally, only our ADV method consistently produces high quality models (avg. reward approaching 1).

LC. The vanilla LC setting reveals that the imitation gap between the expert and the agent observations is nominal. Intuitively, the agent's egocentric partial observations are sufficient to learn to replicate the shortest-path expert actions: it need only follow alleys of safe ground leading to the narrow gaps in otherwise continuous obstacles. This is validated by the high performance of BC$^{\text{tf=1}}$ (and BC$^{\text{demo}}$) which learn only from expert trajectories and have no hope of bridging an imitation gap. Methods learning from demonstrations and RL (*e.g.*, ADV$^{\text{demo}}$+PPO and BC$^{\text{demo}}$ →PPO) perform only marginally better. Thus, when the imitation gap is small and BC from demonstrations is highly successful, we conclude that the gains from using ADVISOR-based methods may be marginal.

LC SWITCH. IL and warm-started methods receive no supervision to explore the *switch* action and thus learning in the dark resulting in poor policies, see Fig 4c. Also, early episode termination when agents encounter lava prevents PPO success due to sparse rewards. ADVISOR leverages it's RL loss to learn to 'switch' on lights after which it successfully imitates the expert.

WC CORRUPT. In Fig. 4d we investigate ADVISOR's ability to learn to ignore a corrupted expert. While this is not what ADVISOR was designed for, it is interesting to see that ADV-based methods are able to accomplish this task and do significantly better than the best performing competitor (BC→PPO). This suggests that ADVISOR is robust to expert failure.

2D-LH. Here we vary the privilege of an expert and study learning from "less intelligent" teachers. Particularly, for each method, we train an f^i-partial policy using an f^j-optimal expert (except for PPO which uses no expert supervision) 25 times. Each policy is then evaluated on 200 random episodes and the average episode length (lower being better) is recorded. For all odd i, j with $1 \leq i \leq 11$, $1 \leq j \leq 15$, and $j \geq i$ we show boxplots of the 25 training runs. Grey vertical lines show optimal average episode lengths for f^i-partial policies.
For BC we find training of an f^i-partial policy with an f^j-expert to result in a near optimal policy when $i = j$ but even small increases in j result in dramatic decreases in performance. This emphasizes the imitation gap. Surprisingly, 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.
Additionally, we find that: (i) PPO can perform well but has high variance across runs due to the problem of sparse rewards, and (ii) for this task, both ADVISOR and DAgger→PPO can ameliorate the impact of the imitation gap but ADVISOR consistently outperforms in all settings.

5 Conclusion

We introduce 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 imitation gap, we introduce a principled adaptive weighting technique (ADVISOR), which we test on a suite of ten 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. With these lessons, in future work, it would be interesting to study ADVISOR for agents within visually-rich 3D environments.

Acknowledgements

This material is based upon work supported in part by the National Science Foundation under Grants No. 1563727, 1718221, 1637479, 165205, 1703166, Samsung, 3M, Sloan Fellowship, NVIDIA Artificial Intelligence Lab, Allen Institute for AI, Amazon, AWS Research Awards, and Siebel Scholars Award. We thank Iou-Jen Liu, Nan Jiang, and Tanmay Gangwani for feedback on this work.

References

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence*, 1995.

Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip S Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *AAAI*, 2016.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. Learning to search better than your teacher. In *ICML*, 2015.

Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. In *ECCV*, 2020a. first two authors contributed equally.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, 2020b.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *ICLR*, 2018a.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018b.

A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *CVPR*, 2018a.

A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural Modular Control for Embodied Question Answering. In *CoRL*, 2018b.

Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *ICCV*, 2017.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *EMNLP*, 2019.

Tanmay Gangwani and Jian Peng. State-only imitation with transition dynamics mismatch. In *ICLR*, 2020.

Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. In *ICLR*, 2017a.

S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. In *CVPR*, 2017b.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *NeurIPS*, 2016.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *AAAI*, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *ICLR*, 2017.

Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G. Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *CVPR*, 2019. first two authors contributed equally.

Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander G. Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *ECCV*, 2020. first two authors contributed equally.

Nan Jiang. On value functions and the agent-environment boundary. *arXiv preprint arXiv:1905.13341*, 2019.

Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *ICML*, 2018.

D. Kingma and J. Ba. A method for stochastic optimization. In *CVPR*, 2017.

Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *NeurIPS*, 2009.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2019.

Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudik, Yisong Yue, and Hal Daumé. Hierarchical imitation and reinforcement learning. In *ICML*, 2018.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Ryan Lowe, Abhinav Gupta, Jakob N. Foerster, Douwe Kiela, and Joelle Pineau. On the interaction between supervision and self-play in emergent communication. In *ICLR*, 2020.

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.

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.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *ICRA*, 2018.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 2018.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 2018.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 2008.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 1991.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS*, 2018.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In *Machine Learning Proceedings*, 1992.

Manolis Savva, Jitendra Malik, Devi Parikh, Dhruv Batra, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, and Vladlen Koltun. Habitat: A platform for embodied AI research. In *ICCV*, 2019.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized Experience Replay. In *ICLR*, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tianlin Tim Shi, Andrej Karpathy, Linxi Jim Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *ICML*, 2017.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, R. Mottaghi, Luke Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *CVPR*, 2020.

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.

Kaushik Subramanian, Charles L Isbell Jr, and Andrea L Thomaz. Exploration from demonstration for interactive reinforcement learning. In *AAMAS*, 2016.

Mark van der Laan and Susan 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.

A.W. van der Vaart. *Asymptotic Statistics*. Asymptotic Statistics. Cambridge University Press, 2000. ISBN 9780521784504. URL https://books.google.com/books?id=UEuQEM5RjWgC.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Manfred Otto Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *ArXiv*, abs/1707.08817, 2017.

Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. Video captioning via hierarchical reinforcement learning. In *CVPR*, 2018.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *ICLR*, 2017.

Larry Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387251456.

Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. Allenact: A framework for embodied ai research. *arXiv*, 2020.