B. Additional Experimental Results

B.1. Estimating the Q Function

In Section 5 we briefly discussed the possibility of avoiding explicitly estimating the Q function. All the terms in (26) can be computed directly, with the exception of the Q function. One approach therefore is to train an additional function approximator targeting the Q function directly. This can then be used to estimate the discounted sum of rewards ahead given a particular action and belief state (when $\beta = 0$) without directly using the Monte Carlo rollouts. However, estimating the Q function increases the computational cost, increases the number of hyperparameters that need tuning, and can lead to instabilities and biased training by over reliance on imperfect function approximators, especially in high-dimensional environments. Therefore, as in many on-policy RL algorithms, an alternative is to use Monte Carlo estimates of the Q function, computed directly from a sampled trajectory (c.f. (27)-(29)).

However, somewhat unexpectedly, this second approach can lead to the systemic failure of A2D in particular *environments*. This can be shown by expanding the definition of $Q^{\pi_\psi}(a, b)$:

$$Q^{\pi_\psi}(a,b) = \mathop{\mathbb{E}}_{p(s,s'|a,b)}\left[\mathop{\mathbb{E}}_{d^{\pi_\psi}(b'|s')}\left[r(s,a,s') + \mathop{\gamma\mathbb{E}}_{\pi_\psi(a'|b')}\left[Q^{\pi_\psi}(a',b')\right]\right]\right], \tag{B.1}$$

where s' and b' are the state and belief state after taking action a in state s and belief state b. Since sampling from $p(s, s'|a, b)$ and $d^{\pi_\psi}(b'|s')$ is intractable, directly using the trajectories is equivalent to using a singled sample value throughout this expression *and* the gradient estimator in (27). Re-using just a single value of s inside and outside of this expectation biases the gradient estimator, as the estimate of Q is *not* conditionally independent of the current (unobserved) state given the belief state. Intuitively, using Monte Carlo rollouts essentially allows the expert to "cheat" by learning using exclusively the true state and reward signal over a *single* time step of a trajectory.

When the Q function is estimated directly, the expectation in (B.1) is estimated directly by the learned Q function, thereby amortizing this inference by learning across many different sampled trajectories. Therefore, from a theoretical perspective, estimating the Q function is important for A2D to be guaranteed to function. However, we find that this bias is only significant in specific environments, and hence, in many environments, explicitly estimating the Q function can be avoided. This reduces the computational cost of the algorithm, and reduces the number of hyperparameters and network architectures that need tuning. Furthermore, and most importantly, this eliminates the direct dependence on faithfully approximating the Q function, which, in environments with high-dimensional observations and actions, can be prohibitively difficult.

To explore this behavior, and verify this theoretical insight, we introduce three variants of the Tiger Door problem, shown in Figure B.1. The first variant, "Tiger Door 1," shown in Figure B.1a, actually corresponds to a gridworld embedding of the original Tiger Door problem (Littman et al., 1995). "Tiger Door 2" & "Tiger Door 3," shown in Figures B.1b and B.1c, then separate the goal by one and two squares respectively.

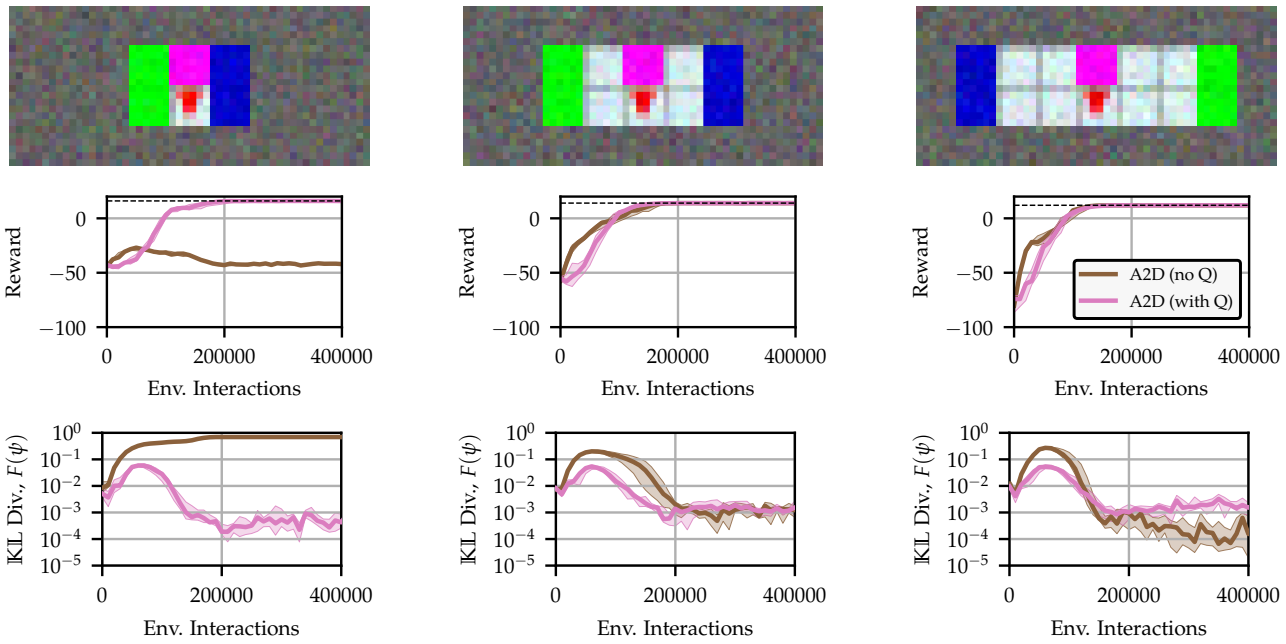
The analysis above predicts that A2D should not be able to solve Tiger Door 1 without direct estimation of the Q function. This is because the expert can reach the goal with certainty in a single action, which ends the episode. This means the expert can always maximize reward by proceeding directly to the goal, and as the episode ends, the gradient signal is dominated by the bias from the single step. This causes the expert to put additional mass on directly proceeding to the goal, even though the goal is not visible to the agent. We note that this is also the most extreme example of this bias, and we believe this environment to be somewhat of an unusual corner-case.

However, in Tiger Doors 2 and 3, the episode does not end immediately after proceeding directly towards the goal. Therefore, the value of proceeding directly towards the goal is diminished, as the marginalization over state provided by GAE and the value function reduces the estimated advantage value. The gradient computed in these scenarios is therefore dramatically less biased, to the point where directly estimating the Q function not required.

The predicted behavior is indeed observed when applying A2D to each Tiger Door variant, shown in Figure B.1. We see that in Tiger Door 1, the correct policy is only recovered when the Q function is explicitly estimated. When the Q function is

*Equal contribution [1]Department of Engineering Science, University of Oxford [2]Department of Computer Science, University of British Columbia [3]Inverted AI [4]Alberta Machine Learning Intelligence Institute (AMII) [5]Montréal Institute for Learning Algorithms (MILA). Correspondence to: Andrew Warrington <andreww@robots.ox.ac.uk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.



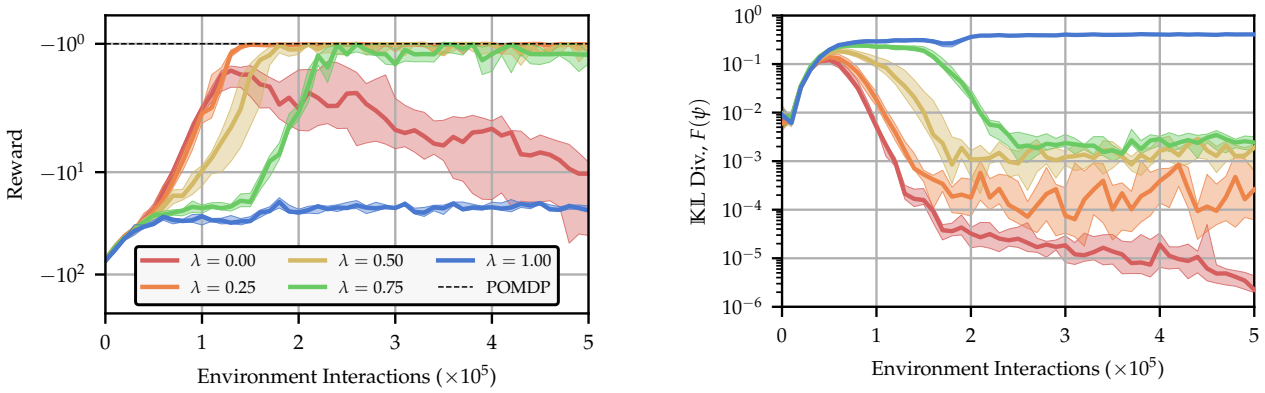
(a) Tiger Door 1. (b) Tiger Door 2. (c) Tiger Door 3.

Figure B.1: Results investigating requirement of directly estimating the Q function, as initially introduced in Section 5 and discussed further in Section B.1. Median and quartiles across 20 random seeds are shown. The Q function is learned targeting the expected discounted sum of rewards ahead conditioned on a particular (belief) state-action pair. A value function is also learned in this way, and is used in conjunction with the Q function to directly estimate the advantage in (27). Hence the A2D gradient is computed without direct use of Monte Carlo rollouts. When no Q function is being used, the advantage is computed using GAE (c.f. Equations (27)-(29)), with $\lambda = 0.5$. We instantly anneal $\beta = 0$. Figure B.1a: Training curves for Tiger Door 1 (Littman et al., 1995). As predicted by the discussion in Section B.1, A2D does not converge to the correct policy if a Q function is not simultaneously learned. This deficiency is instrumented by the high $\mathbb{KL}$ divergence throughout training and a discrepancy between the expected reward of the expert and agent. If a Q function is learned, the desired partially observed behavior is recovered. Figure B.1b and B.1c: By separating the goal by at least one square means the desired behavior is recovered regardless of whether a Q function is used. This is because the bias has been reduced through the use of GAE and the introduction of additional random variables.

not estimated, the expert directly optimizes just the reward under the MDP, earning itself a reward of 18, but rendering an implicit policy that performs poorly, earning a reward of -42. In Tiger Doors 2 & 3, the correct trainee policy is recovered regardless of whether a Q function is explicitly learned. Interestingly, we observe that the policy divergence, $F(\psi)$, is often lower during training when using the Q function. This further reinforces that estimating the Q function more directly optimizes the reward of the trainee. We note, however, that the *final* divergence achieved by the Q function is often higher than that obtained without Q. This is likely due to the systemic bias introduced by using function approximation. Note that for all of these experiments we use the compact representation.

We also explore, in Figure B.2, the affect that the GAE parameter (Schulman et al., 2015b), λ, has on A2D training. Inspecting (29) indicates that GAE provides the ability to diminish the unmodelled dependence on s_t, and hence reduce the bias in the estimator by attenuating future reward from the Monte Carlo rollouts and replacing this reward with the correctly amortized value, integrating over the true state, estimated by the value function (which in the limit of $\beta = 0$ is only conditioned on b_t). This suggests that $\lambda = 0$, corresponding to the expected temporal difference reward, is as close to the theoretically ideal Q function based estimator in (26) as is possible. The dependency on s_t (as denoted in (29)) is maximally reduced, to the point where it only affects the gradient signal for a single step (further reinforcing why Tiger Door 1 fails, but Tiger Doors 2 & 3 succeed). In contrast, using $\lambda = 1$ maximizes the bias, by not attenuating any Monte Carlo reward signal.

We observe this behavior in Figure B.2. We see that $\lambda = 1$ does not converge to the optimal solution, as the bias term



(a) Reward.

(b) Divergence.

Figure B.2: Results showing the affect of the GAE parameter λ on A2D, applied to the Tiger Door 2 environment. The reward is normalized such that the optimal reward under the POMDP is -10^0. As predicted, we see that lower λ values yield faster convergence and monotonically lower policy divergences. However, as this is equivalent to TD0, the RL is unstable (obscured in this plot are short, sharp drops in the reward and rises in the divergence). Eventually, all traces begin to diverge from the optimal policy. For any λ value less than unity, convergence is stable (and the short, sharp drops do not exist). Finally, and again as predicted, we see that learning does not converge when $\lambda = 1$, with reward remaining flat and low, and the divergence remaining high.

dominates, effectively halting learning. Lower λ values allow more state information to be integrated out, and hence the partially observed policy can improve. This is seen by faster and more stable convergence for lower λ values. This can also be observed in Figure B.2b, where lower λ values achieve a lower policy divergence. This implies that the expert is less able to leverage state information to learn a policy that cannot be represented by the agent. However, reducing λ must unfortunately be balanced against the limiting behavior of GAE, which corresponds to a TD0 estimate of the return. In this regime, bias introduced by function approximation can make the convergence of RL unreliable. Indeed, even in idealized settings, it can be shown such estimators diverge without further modifications, such parameter averaging (Maei et al., 2009).

Therefore, the hyperparameter λ takes on additional importance when tuning A2D using the biased Monte Carlo gradient estimator. If the coefficient λ is too close to zero, then the effects of bootstrapping error can lead learning to stall, unstable solutions, or even divergence, as is often observed in RL, and may reduce the effectiveness of GAE and RL by overly relying on function approximators. However, this lower λ value reduces the bias in the estimator, and hence provides faster convergence, more stable convergence, and achieves a lower final policy divergence (c.f. $\lambda = 0.00$ in Figure B.2). If λ is too close to unity, there may too much bias in the gradient estimate. This bias may force, either, A2D to not converge outright if $\lambda = 1.00$, or, cause A2D to drift away slightly from the optimal solution after convergence as the expert aggregates this slight bias into the solution. In practice, we find that this second failure mode only occurs once learning has already converged to the optimal solution, and so the optimal policy can simply be taken prior to any divergence. Further analysis of this effect, both theoretically, such as defining and bounding the precise nature of this bias, and practically, such as adaptively adjusting λ to control the bias-variance trade-off, are interesting directions of future work.

We note that Frozen Lake subtly exhibits the bias in the Monte Carlo gradient estimator if the value of λ is too high. First, the trainee quickly converges to the optimal partially observing policy (and so the environment *is* solved). Then, after *many* more optimization steps, the probability that the agent steps onto the ice can rise slightly. This causes the divergence to rise slightly and the expected stochastic reward to fall slightly before stabilizing. The rise is small enough that the deterministic policy evaluation remains unchanged. However, as predicted by the analysis above, this behavior can be eradicated by reducing the value of λ. However, in RL generally, lowering λ can stall, or even halt, learning from the outset by overreliance on biased function approximators. This can cause the optimization to become stuck in local minima. As eluded to above, we find that we can control this behavior by slightly reducing the value of λ from its initial value during the optimization. This minimizes the dependency on function approximators early in training and retains the fast and reliable convergence to the optimal policy, and then attenuates any bias after learning has converged. While we believe this behavior only presents in a small number of very specific environments and can be eradicated through tuning of hyperparameters, we propose that further investigation of adaptively controlling λ during the optimization is a promising and practical future research

$$= \mathop{\mathbb{E}}_{d^{\pi_\psi}(s,b)} \left[\mathop{\mathbb{E}}_{\pi_\theta(a|s)} \left[Q^{\pi_\psi}(a,b) \nabla_\theta \log \pi_\theta(a|s) \right] \right], \tag{C.69}$$

□

The A2D gradient estimator given in (26) then adds an importance weight to the inner expectation, as we rollout under π_ψ. This allows us to instead weight actions sampled under the current trainee policy, π_ψ, without biasing the gradient estimator. We can then cast this estimator in terms of advantage, where the Q function with the value function subtracted as a baseline to reduce the variance of the estimator.

Lemma 6 (A2D Advantage-based gradient estimator, c.f. Section 5, Equation (27))**.** *We can construct a gradient estimator from* (C.69) *that uses the advantage by subtracting the value function as a* baseline *(Bertsekas, 2019; Sutton, 1992; Williams, 1992):*

$$\nabla_\theta J_\psi(\theta) = \mathop{\mathbb{E}}_{d^{\pi_\psi}(s,b)} \left[\mathop{\mathbb{E}}_{\pi_\theta(a|s)} \left[Q^{\pi_\psi}(a,b) \nabla_\theta \log \pi_\theta(a|s) \right] \right], \tag{C.70}$$

$$= \mathop{\mathbb{E}}_{d^{\pi_\psi}(s,b)} \left[\mathbb{E}_{\pi_\theta(a|s)} \left[(Q^{\pi_\psi}(a,b) - V^{\pi_\psi}(b)) \nabla_\theta \log \pi_\theta(a|s) \right] \right]. \tag{C.71}$$

Proof. It is sufficient to show that:

$$\mathop{\mathbb{E}}_{d^{\pi_\psi}(s,b)} \left[\mathop{\mathbb{E}}_{\pi_\theta(a|s)} \left[V^{\hat{\pi}_\theta}(b) \nabla_\theta \log \pi_\theta(a|s) \right] \right] = 0, \tag{C.72}$$

which can be shown easily as:

$$\mathbb{E}_{d^{\pi_\psi}(s,b)} \left[\mathbb{E}_{\pi_\theta(a|s)} \left[V^{\hat{\pi}_\theta}(b) \nabla_\theta \log \pi_\theta(a|s) \right] \right] = \mathbb{E}_{d^{\pi_\psi}(s,b)} \left[V^{\hat{\pi}_\theta}(b) \mathbb{E}_{\pi_\theta(a|s)} \left[\nabla_\theta \log \pi_\theta(a|s) \right] \right] \tag{C.73}$$

$$= \mathbb{E}_{d^{\pi_\psi}(s,b)} \left[V^{\hat{\pi}_\theta}(b) \int_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a|s) \mathrm{d}a \right], \tag{C.74}$$

$$= \mathbb{E}_{d^{\pi_\psi}(s,b)} \left[V^{\hat{\pi}_\theta}(b) \nabla_\theta \int_{a \in \mathcal{A}} \pi_\theta(a|s) \mathrm{d}a \right], \tag{C.75}$$

$$= \mathbb{E}_{d^{\pi_\psi}(s,b)} \left[V^{\hat{\pi}_\theta}(b) \nabla_\theta 1 \right] = 0, \tag{C.76}$$

Noting that this is an example of the *baseline* trick used throughout RL (Bertsekas, 2019; Sutton, 1992; Williams, 1992). □

This allows us to construct a gradient estimator using the advantage, which in conventional RL, is observed to reduce the variance of the gradient estimator compared to directly using the Q values.

We are now able to prove an exact form of the A2D update. This proof is similar to Theorem 2, however, no longer assumes identifiability of the POMDP-MDP process pair by instead updating the expert at each iteration.

C.3.2. Theorem 3

Theorem 3 (Convergence of Exact A2D, reproduced from Section 5)**.** *Under exact intermediate updates to the expert policy (see* (C.78)*), the following iteration converges to an optimal partially observed policy $\pi_{\psi^*}(a|b) \in \Pi_\phi$, provided Assumption 1 holds:*

$$\psi_{k+1} = \arg\min_{\psi \in \Psi} \mathop{\mathbb{E}}_{d^{\pi_{\psi_k}}(s,b)} \left[\mathbb{KL} \left[\pi_{\theta_k^*}(a|s) \,||\, \pi_\psi(a|b) \right] \right], \tag{C.77}$$

$$\textit{where} \quad \hat{\theta}_k^* = \arg\max_{\theta \in \Theta} \mathop{\mathbb{E}}_{d^{\pi_{\psi_k}}(b) \hat{\pi}_\theta(a|b)} \left[Q^{\hat{\pi}_\theta}(a,b) \right]. \tag{C.78}$$

Proof. We will again, for ease of exposition assume that a unique optimal policy exists, as in Theorem 2. We again reinforce that this is not a *requirement*. Extending this proof to include multiple optimal partially observable policies only requires that we reason about the $\mathbb{KL}$ divergence between π_{ψ_k} and π_{ϕ^*} at each step in the proof, instead of showing that the optimal parameters are equal. This alteration is technically simple, but is algebraically and notationally onerous. Similar to Theorem

C.3.3. DISCUSSION

In this section we presented a derivation of exact A2D, where the expert is defined through the exact internal maximization step defined in (C.78). We include these derivations to show the fundamental limitations of imitation learning and thus A2D under ideal settings. Exactly performing this maximization is difficult unto itself, and therefore the A2D algorithm presented in Algorithm 1 simply assumes that this maximization is performed sufficiently accurately to produce meaningful progress in policy space. Although we note that empirically A2D is robust to inexact updates, we defer the challenging task of formally and precisely quantifying the convergence properties of A2D under inexact internal updates to future work.

D. Experimental Configurations

D.1. Gridworld

We implemented both gridworld environments by adapting the *MiniGrid* environment provided by Chevalier-Boisvert et al. (2018). For both gridworld experiments, the image is rendered as a 42×42 RGB image. The agent has four actions available, moving in each of the compass directions. Each movement incurs a reward of -2, hitting the weak patch of ice or tiger incurs a reward of -100, and reaching the goal incurs a reward of 20. Pushing the button in Tiger Door is free, but effectively costs 4 due to its position, or 2 in the Q function experiments. Policy performance is evaluated every five steps by sampling 2000 interactions under the stochastic policy. A discount factor of $\gamma = 0.995$ was used and an upper limit of $T = 200$ is placed on the time horizon.

For experts and agents/trainees that use the compact representation, the policy is a two layer MLP that accepts a compact vector as input, with 64 hidden units in each layer, outputting the log-probabilities of each action. The value function uses the same architecture and input, but outputs a number representing the reward ahead. The value function is learned by minimizing the mean squared error in the discounted reward-to-go. 32 batches are constructed from the rollout and are used to update the value function using ADAM (Kingma & Ba, 2014) with a learning rate of 7×10^{-4}, for 25 epochs. Q functions are only used here with compact representations, and so we can simply append a one-hot encoding of the action to the (flat) input vector. The Q function is then learned in the same way as the value function, except for with a slightly lower learning rate of 3×10^{-4}. Policies and value functions conditioned on images use a two layer convolutional encoder, each with 32 filters, and a single output layer mapping to a flat hidden state with 50 hidden units. Image-based policies and value functions learn separate image encoders in the gridworld examples, whereas in the CARLA examples, a shared encoder is used. This output is then used as input into a two layer MLP, each with 64 hidden units, outputting the log-probabilities of each action or the expected discounted reward ahead. L2 regularization is applied to all networks, with a coefficient of 0.001.

We learn the MDP expert (*RL (MDP)*) by applying TRPO with batch sizes of $2,000$ and a trust region of 0.01. An entropy regularizer is applied directly to the advantages computed, with coefficient 1. We set $\lambda = 0.95$ in the GAE calculation (Schulman et al., 2015b). This trust region, regularization and λ value are used throughout, unless otherwise stated. Reinforcement learning in the POMDP (*RL*) uses separate policies and value functions conditioned on the most recent image. In asymmetric reinforcement learning (*RL (Asym)*) the policy is conditioned on the image, but the value function takes the compact and omniscient state representation (s_t) as input. Policies and value functions are then learned using the same process as before.

The policy learned by *RL (MDP)* is then used as the expert in AIL (*AIL*), where $2,000$ samples are collected at each iteration and appended to a rolling buffer of $5,000$ samples. The KL-divergence between the expert and trainee action distributions is minimized by performing stochastic gradient descent, using ADAM with a learning rate of 3×10^{-4}, using a batch size of 64 for two whole epochs per iteration. We find that the MDP converges within approximately $80,000$ environment interactions, and so we begin the AIL line at this value. β is annealed to zero after the first time step (as recommended by Ross et al. (2011)).

For experiments using a pretrained encoder (*Pre-Enc*), we roll out for $10,000$ environment interactions under a trained MDP expert from *RL (MDP)* to generate the data buffer. The encoder, that takes images as input and targets the true state vector, is learned by regressing the predictions on to the true state. We perform 100 training epochs with a learning rate of 3×10^{-4}. We start this curve at the $80,000$ interactions required to train the expert from which the encoder is learned. We use an asymmetric value function conditioned on the true state. The encoder is then frozen and a two-layer, 64 hidden unit MLP policy head is learned using TRPO. We found that a lower trust region size of 0.005 was required for Tiger Door to stably converge. We confirmed separately for both pretrained encoders and AIL that the encoder class can represent and learn the required policies and transforms, and both converge to the solution of the MDP when conditioned on *omniscient* image-based input.

For A2D, expert and trainee policies are initialized from scratch, and are learned using the broadly the same settings as *RL (MDP)* and *AIL*. In A2D, we decay β with coefficient 0.8 at each iteration, although faster β decays did not hurt performance. Slower β decays can lead to higher and longer divergences during training, and can lead to the agent becoming trapped in local optima. We use a higher entropy regularization coefficient, equal to 10, finding that this increased regularization helped A2D avoid falling into local minima, although this can be further ameliorated by setting $\beta = 0$ throughout, as we do in the CARLA experiments. We found for Frozen Lake that a lower $\lambda = 0.9$ value of yielded more stable convergence and a

lower final policy divergence (we refer the reader to Section B.1 for more information). Value and Q functions are learned by individually targeting the sum of rewards ahead (i.e. is not back-propagated through any mixture). We note that choosing to parameterize the mixture value function as the weighted sum of individual value functions is an assumption. However, we note that we require $\beta \to 0$ for the gradient to be unbiased. In this limit the mixture is equal to just the value function of the agent. Therefore, explicitly parameterizing the value function in this way *ensures* that state information is removed from the estimation. Exploring different ways of parameterizing the value function is a potential topic for future research.

In Section B.1 we use a λ value of 0.5 in GAE (Schulman et al., 2015b) (when not sweeping over λ). We used an entropy regularizer of 0.02 is applied directly to the surrogate loss. We also use TRPO with a trust region KL-divergence of 0.001.

D.2. CARLA Experiments

We implemented our autonomous vehicle experiment using CARLA (Dosovitskiy et al., 2017). This scenario represents a car driving forward at the speed limit, while avoiding a pedestrian which may run out from behind a vehicle 50% of the time, at a variable speed. There are a total of 10 waypoints, indicating the path the vehicle should take as produced by an external path planner. We enforce that the scenario will end prematurely if one of the following occurs: a time limit of 90 time-steps is reached, a collision with a static object, a lane invasion occurs, if a waypoint is not gathered within 35 time-steps, or, the car's path is not within a certain distance of the nearest waypoint. We found that inclusion of these premature endings was crucial for efficient learning. The reward surface for this problem is generated using a PID controller which is computed using an example nominal trajectory. The reward at any given time-step is defined as the product of the absolute difference between the agents actions and the optimal actions by a low-level PID controller to guide the vehicle to the next waypoint, and is bounded to lie in $[0, 1]$.

For the expert policy used both in AIL and A2D, we use a two layer MLP with 64 hidden units and ReLU activations. The agent and trainee policies use a shared image encoder (Laskin et al., 2020a;b; Yarats et al., 2021), followed by the same MLP architecture as the expert policy to generate actions. The RL algorithm used in both the expert and agent RL updates is PPO (Schulman et al., 2017) with generalized advantage estimation (GAE) (Schulman et al., 2015b). We detach the encoder during the policy update and learn the encoder during the value function update (Laskin et al., 2020a;b; Yarats et al., 2021). In A2D we use the MLP defined above for the expert policy. The trainee policy and value functions use a common encoder, updated during the trainees value update and frozen during the policy update, and the MLP defined above as the policy head or value head network. For all algorithms we used a batch size of 64 in both the PPO policy update, value function update, and the imitation learning update. As in the previous experiments, in the imitation learning step, we iterate through all data seen and stored in the replay buffer. We found that starting the β parameter at zero produced faster convergence.

We performed a coarse-grained hyperparameter search using the Bayesian optimization routine provided by the experimental control and logging software *Weights & Biases* (Biewald, 2020). This allows us to automate hyperparameter search and distribute experimental results for more complex experiments in a reproducible manner. Each method was provided approximately the same amount of search time, evaluating at least 60 different hyperparameter settings. The optimal settings were then improved manually over the course of approximately 5 further tests. We score each method and hyperparameter setting using a running average of the reward over the previous 25 evaluation steps, and used early stopping if a run consistently performed poorly.

Each algorithm uses different learning rates and combinations of environment steps between updates. For example, we found that all AIL algorithms performed best when taking 10 steps between updates, RL in the expert tended to work better by taking more steps in between updates (≈ 400) with a larger step-size $\approx 4 \times 10^{-4}$, where the agents RL updates favored fewer steps ≈ 75 with smaller steps 7×10^{-5}. For all algorithms 4 parallel environments were run concurrently, as this was observed to improve performance across all algorithms. This was especially the case for the RL methods, which relied on more samples to accurately compute the advantage.

We note that there is a point of diminishing returns for PPO specifically (Engstrom et al., 2020), where policy learning degrades as the number of examples per update increases. Even though the advantage becomes progressively more accurate with increasing sample size, the mini-batch gradient decent procedure in PPO eventually leads to off-policy behavior that can be detrimental to learning. We also found that pre-generating a number of trajectories and pretraining the value function tended to improve performance for both A2D, as well as the compact expert RL algorithm. For A2D specifically, this ensured that the replay buffer for imitation learning was reasonably large prior to learning in the expert. This ensures that for any given update, the agent tends to be close to the expert policy, ensuring that the "off-policy" RL update is not too severely destabilized through importance weighting. To further improve this, we also introduced delayed policy updates,

which further reduced the divergence between expert and the agent in A2D. In both A2D and the RL setups, this also helped ensure that the value function is always converging faster than the policy, ensuring that the error in the resulting advantage estimates are low.

E. Additional Related Work

We now present a comprehensive review of existing literature not already covered. Exploiting asymmetric learning to accelerate learning has been explored in numerous previous work under a number of different frameworks, application domains, and levels of theoretical analysis.

The notion of using fully observed states unavailable at deployment time is often referred to as exploiting "privileged information" (Vapnik & Vashist, 2009; Lambert et al., 2018). For clarity, we refer to the expert as having access to privileged information, and the agent as only having access to a partial observation. We note that the use of the term "expert" does not imply that this policy is necessarily optimal under the MDP. Indeed, in A2D, the expert is co-trained with the agent, such that the expert is approximately a uniform random distribution at the start of the learning procedure. The term privileged information is more general than simply providing the world state, and may include additional loss terms or non-trivial transforms of the world state that expedite learning the agent. In this work, we exclusively consider the most general scenario where the privileged information is the full world state. However, there is nothing precluding defining an extended state space to include hand-designed features extracted from the state, or, using additional, hand crafted reward shaping terms when learning (or adapting) the expert.

E.1. Encodings

The first use-case we examine is probably the simplest, and the most widely studied. Asymmetric information is used to learn an encoding of the observation that reduces the dimensionality while retaining information. Standard reinforcement learning approaches are then employed freezing this encoding. Two slight variations on this theme exist. In the first approach, an MDP policy is learned to generate rollouts conditioned on omniscient information, and an encoder is learned on state-observation pairs visited during these rollouts (Finn et al., 2016; Levine et al., 2016). Either the encoder acts to directly recover the underlying states, or simply learns a lower-dimensional embedding where performing reinforcement learning is more straightforward.

Andrychowicz et al. (2020) explore learning to manipulate objects using a mechanical hand using *both* state information from the robot (joint poses, fingertip positions etc) and RGB images. This particular application is an interesting hybrid approach dictated by the domain. State information pertaining to the manipulator is easily obtained, but state information about the pose of the object being manipulated is unavailable and must be recovered using the images. A controller is learned in simulation (MDP), while simultaneously (and separately from the MDP) a separate perception network is learned that maps the image to the pose of the object being manipulated. State information and pose encoding are then concatenated and used as the state vector on which the policy acts. While the pose of the object is unobserved, it is readily recoverable from a single frame (or stack of frames), and hence the partial observation is predominantly a high-dimensional and bijective embedding of the true state. If the true position of the hand was not available, this would be less certain as the object and other parts of the manipulator obfuscates much of the manipulator from any of the three viewpoints (more viewpoints would of course reverse this to being a bijection). The use of a recurrent policy further improves the recovery of state as only the innovation in state needs to be recovered.

E.2. Asymmetric values

Another well-explored use-case is to instead exploit asymmetric information for to improve learning a value or Q- function (Könönen, 2004; Pinto et al., 2017; Andrychowicz et al., 2020). This is achieved by conditioning either the value function or Q-function on different information than the policy that is either more informative, or lower dimensional representations, and can help guide learning (Könönen, 2004; Pinto et al., 2017). Learning the value or Q function in a lower-dimensional setting enables this function to be learned more stably and with fewer samples, and hence can track the current policy more effectively. Since the value and Q-function are not used at test time, there is no requirement for privileged information to be available when deployed. Pinto et al. (2017) introduce this in a robotics context, using an asymmetric value function, conditioned on the true underlying state of a robotic manipulator, to learn a partially observing agent conditioned only on a third-person monocular view of the arm. Similar ideas were explored previously by Könönen (2004) in relation to semi-centralized multi-agent systems, where each agent only partially observes the world state, but a central controller is able to observe the whole state. The state used by the central controller is used to evaluate the value of a particular world state, whilest each agent only acts on partial information.

E.3. Behavioral Cloning & Imitation Learning

Behavioral cloning and imitation learning (Kang et al., 2018; Ross et al., 2011), introduced in Main Section 2.3, is, in our opinion, an under-explored avenue for expediting learning in noisy and high-dimensional partially observed processes. The main observation is that this process separates learning to act and learning to perceive (Chen et al., 2020). The fully observing expert learns to act, without the presence of extraneous patterns or noise. The agent then learns to perceive such that it can replicate the actions of the expert. A major benefit of cloning approaches is that perception is reduced to a supervised learning task, with lower variance than the underlying RL task.

Pinto et al. (2017) briefly assess using asymmetric DAgger as a baseline. It is observed that the agent learns quickly, but actually converges to a worse solution than the asymmetric actor-critic solution. This difference is attributed to the experts access to (zero variance) state information otherwise unavailable to the partially observing agent. Our work builds on this observation, seeking to mitigate such weaknesses. Surprisingly, and to the best of our knowledge, no work (including Pinto et al. (2017)) has provided and in-depth analysis of this method, or directly built off this idea.

Chen et al. (2020) showed that large performance gains can be found in an autonomous vehicles scenario by using IL through the use of an asymmetric expert, specifically for learning to drive in the autonomous vehicle simulator CARLA (Dosovitskiy et al., 2017). Chen et al. (2020) train an expert from trajectories, created by human drivers, using behavioral cloning conditioned on an encoded aerial rendering of the environment including privileged information unavailable to the agent at deployment time. The aerial rendering facilitates extensive data augmentation schemes that would otherwise be difficult, or impossible, to implement in a symmetric setting. The agent is then learned using DAgger-based imitation learning. However, this general approach implicitly makes assumptions about the performance of the expert, as well as the underlying identifiability (as we define in Section 4) between the underlying fully and partially observed Markov decision processes.

Other works combine RL and IL to gain performance beyond that of the expert by considering that the expert is sub-optimal (Choudhury et al., 2018; Sun et al., 2018; Weihs et al., 2020), where the performance differential is either as a result of asymmetry, or, the expert simply not being optimal. These works, most often, train a policy that exploits knowledge of the performance differential between the expert and agent, or, the difference in policies. The weight applied to the sample in IL is increased for policies that are similar, or, where the performance gap is small. The example is then down-weighted when it is believed that the expert provides poor supervision in that state. However, these works do not consider updating the expert, and instead focus on ameliorating the drawbacks of AIL using derived statistics. In our work, we seek to define a method for updating an expert directly.

E.4. Co-learning Expert and Agent

Work that is maybe thematically most similar to ours investigates co-training of the agent and expert. This builds on the AIL approach, where instead of assuming an optimal expert exists, the expert and agent policies are learned simultaneously, where either an additional training phase as added to "align" the expert and agent (Salter et al., 2019; Song et al., 2019), architectural modification (Kamienny et al., 2020), or both (Schwab et al., 2019). An alternative method for deriving such an aligning gradient is to introduce an auxiliary loss regularizing the representation used by the agent to be predictive of the the underlying state, or, a best-possible belief representation (Nguyen et al., 2020).

Salter et al. (2019) trains separate policies for agent and expert using spatial attention, where the expert is conditioned on the state of the system, and the agent is conditioned on a monocular viewpoint. By inspecting the attention map of expert and agent, it is simple to establish what parts of the state or image the policy is using to act. An auxiliary (negative) reward term is added to the reward function that penalizes differences in the attention maps, such that the agent and expert are regularized to use the same underlying features. This auxiliary loss term transfers information from the MDP to the POMDP. The main drawbacks of this approach however are its inherent reliance on an attention mechanism, and tuning the hyperparameters dictating the weight of having a performant agent, expert and the level of alignment between the attention mechanisms. Further, using a attention as the transfer mechanism between the agent and expert somewhat introduces an additional layer of complexity and obfuscation of the actual underlying mechanism of information transfer.

Song et al. (2019) present an algorithm, CoPiEr, that co-trains two policies, conditioned on different information (any combination of fully or partially observing). CoPiEr rolls out under both policies separately, and then selects the rollouts from the policy that performs the best. These samples are then used in either an RL or IL (or hybrid of the two) style update. In this sense, the better performing policy (with ostensibly "better" rollouts) provides high-quality supervision to the policy with lower quality rollouts. MDP to POMDP transfer or privileged information is not considered. Most significantly,

imitation learning is proposed as a method of transferring from one policy to another, or, RL augmented with an IL loss to provide better supervision while retaining RLs capability to explore policy space.

Schwab et al. (2019) on the other hand extend Pinto et al. (2017) by introducing multitask reinforcement learning themes. A "task" is uniquely described by the set of variables that the policy is conditioned on, such as images from different view points, true state information and proprioceptive information. An input-specific encoder encodes each observation before mixing the encoded input features and passing these to a head network which outputs the actions. Instead of aligning attention mechanisms (as per Salter et al. (2019)), Schwab et al. (2019) the head network is shared between tasks providing alignment between the single-input policies. At test time, only those observations that are available need to be supplied to the policy, respecting the partial observability requirement at test time. This approach does not explicitly use an expert, instead using a greater range of more informative information channels to efficiently learn the policy head, while simultaneously co-training the channel-specific encoders.

Finally, the work of Kamienny et al. (2020) present privileged information dropout (PI-D). The general approach of information dropout (Achille & Soatto, 2018) is to lean a model while randomly perturbing the internal state of the model, effectively destroying some information. The hypothesis is that this forces the model to learn more robust and redundant features that can survive this corruption. Kamienny et al. (2020) use this theme by embedding both partial observation and state, where the state embedding is then used to corrupt (through multiplicative dropout) the internal state of the agent. The PI expert is then able to mask uninformative patterns in the observations (using the auxiliary state information), facilitating more efficient learning. The PI can then be easily marginalized out by not applying the dropout term. Importantly however, reinforcement learning is still performed in the partially observing agent, a characteristic we wish to avoid due to the high-variance nature of this learning.

References

Achille, A. and Soatto, S. Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2897–2905, 2018.

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. Optimality and approximation with policy gradient methods in Markov decision processes. In *Proceedings of Thirty Third Conference on Learning Theory*. PMLR, 2020.

Andrychowicz, O. A. M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation. *International Journal of Robotics Research*, 39(1):3–20, 2020.

Arora, S., Choudhury, S., and Scherer, S. Hindsight is only 50/50: Unsuitability of mdp based approximate pomdp solvers for multi-resolution information gathering. *arXiv preprint arXiv:1804.02573*, 2018.

Bertsekas, D. P. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335, 2011.

Bertsekas, D. P. *Reinforcement learning and optimal control*. Athena Scientific Belmont, MA, 2019.

Bertsekas, D. P. and Tsitsiklis, J. N. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

Biewald, L. Experiment Tracking with Weights and Biases, 2020. Software available from wandb.com.

Chen, D., Zhou, B., Koltun, V., and Krähenbühl, P. Learning by cheating. In *Conference on Robot Learning*, pp. 66–75. PMLR, 2020.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for OpenAI gym. https://github.com/maximecb/gym-minigrid, 2018.

Choudhury, S., Bhardwaj, M., Arora, S., Kapoor, A., Ranade, G., Scherer, S., and Dey, D. Data-driven planning via imitation learning. *The International Journal of Robotics Research*, 37(13-14):1632–1672, 2018.

Deisenroth, M. P., Neumann, G., Peters, J., et al. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403, 2013.

Doshi-Velez, F., Pfau, D., Wood, F., and Roy, N. Bayesian nonparametric methods for partially-observable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):394–407, 2013.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. Deep spatial autoencoders for visuomotor learning. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:512–519, 2016.

Igl, M., Zintgraf, L., Le, T. A., Wood, F., and Whiteson, S. Deep variational reinforcement learning for POMDPs. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2117–2126. PMLR, 2018.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Kamienny, P.-A., Arulkumaran, K., Behbahani, F., Boehmer, W., and Whiteson, S. Privileged information dropout in reinforcement learning. *arXiv:2005.09220*, 2020.

Kang, B., Jie, Z., and Feng, J. Policy optimization with demonstrations. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*. PMLR, 2018.

Kato, S., Takeuchi, E., Ishiguro, Y., Ninomiya, Y., Takeda, K., and Hamada, T. An Open Approach to Autonomous Vehicles. *IEEE Micro*, 35(6):60–68, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Könönen, V. Asymmetric multiagent reinforcement learning. *Web Intelligence and Agent Systems: An international journal*, 2(2):105–121, 2004.

Lambert, J., Sener, O., and Savarese, S. Deep learning under privileged information using heteroscedastic dropout. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8886–8895, 2018.

Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. Dart: Noise injection for robust imitation learning. *arXiv preprint arXiv:1703.09327*, 2017.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119*, 2020. arXiv:2004.04136.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17:1–40, 2016.

Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. Learning policies for partially observable environments: Scaling up. *Seventh International Conference on Machine Learning*, pp. 362–370, 1995.

Maei, H. R., Szepesvari, C., Bhatnagar, S., Precup, D., Silver, D., and Sutton, R. S. Convergent temporal-difference learning with arbitrary smooth function approximation. In *NIPS*, pp. 1204–1212, 2009.

Meng, Z., Li, J., Zhao, Y., and Gong, Y. Conditional teacher-student learning. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6445–6449. IEEE, 2019.

Murphy, K. P. A survey of POMDP solution techniques. *environment*, 2:X3, 2000.

Nguyen, H., Daley, B., Song, X., Amato, C., and Platt, R. Belief-grounded networks for accelerated robot learning under partial observability. *arXiv preprint arXiv:2010.09170*, 2020.

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

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

Ross, S., Gordon, G. J., and Bagnell, J. A. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research*, 15:627–635, 2011.

Salter, S., Rao, D., Wulfmeier, M., Hadsell, R., and Posner, I. Attention-privileged reinforcement learning. *arXiv preprint arXiv:1911.08363*, 2019.

Sasaki, F. and Yamashina, R. Behavioral cloning from noisy demonstrations. In *International Conference on Learning Representations*, 2021.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Schwab, D., Springenberg, J. T., Martins, M. F., Neunert, M., Lampe, T., Abdolmaleki, A., Hertweck, T., Hafner, R., Nori, F., and Riedmiller, M. A. Simultaneously learning vision and feature-based control policies for real-world ball-in-a-cup. In *Robotics: Science and Systems XV*, 2019.

Song, J., Lanka, R., Yue, Y., and Ono, M. Co-training for policy learning. *35th Conference on Uncertainty in Artificial Intelligence*, 2019.

Spaan, M. T. J. *Partially Observable Markov Decision Processes*, pp. 387–414. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-27645-3.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

Sun, W., Bagnell, J. A., and Boots, B. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *6th International Conference on Learning Representations*, pp. 1–14, 2018.

Sutton, R. *Reinforcement Learning*. The Springer International Series in Engineering and Computer Science. Springer US, 1992. ISBN 9780792392347.

Vapnik, V. and Vashist, A. A new learning paradigm: Learning using privileged information. *Neural networks*, 22 (5-6):544–557, 2009.

Weihs, L., Jain, U., Salvador, J., Lazebnik, S., Kembhavi, A., and Schwing, A. Bridging the imitation gap by adaptive insubordination. *arXiv preprint arXiv:2007.12173*, 2020.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Wymann, B., Espie, C. G., Dimitrakakis, C., Coulom, R., and Sumner, A. TORCS: The Open Racing Car Simulator, 2014.

Yarats, D., Kostrikov, I., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.