Learning Social Learning

Kamal Ndousse[1], Douglas Eck[2], Sergey Levine[2,3], Natasha Jaques[2,3]
[1]OpenAI, San Francisco, CA, USA
[2]Google Research - Brain team, Mountain View, CA, USA
[3]UC Berkeley, Berkeley, CA, USA

Abstract

Social learning is a key component of human and animal intelligence. By taking cues from the behavior of experts in their environment, social learners can acquire sophisticated behavior and rapidly adapt to new circumstances. This paper investigates whether independent reinforcement learning (RL) agents in a multi-agent environment can use social learning to improve their performance using cues from other agents. We find that in most circumstances, vanilla model-free RL agents do not use social learning, even in environments in which individual exploration is expensive. We analyze the reasons for this deficiency, and show that by introducing a model-based auxiliary loss we are able to train agents to leverage cues from experts to solve hard exploration tasks. The generalized social learning policy learned by these agents allows them to not only outperform the experts with which they trained, but also achieve better zero-shot transfer performance than solo learners when deployed to novel environments with experts. In contrast, agents that have not learned to rely on social learning generalize poorly and do not succeed in the transfer task. Further, we find that by mixing multi-agent and solo training, we can obtain agents that use social learning to out-perform agents trained alone, even when experts are not available. This demonstrates that social learning has helped improve agents' representation of the task itself. Our results indicate that social learning can enable RL agents to not only improve performance on the task at hand, but improve generalization to novel environments.

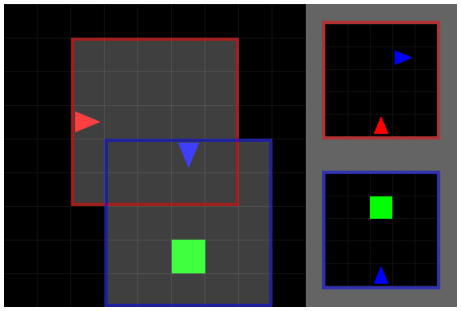
Introduction

Social learning—learning by observing the behavior of other agents in the same environment—enables both humans and animals to discover useful behaviors that would be difficult to obtain through individual exploration, and adapt rapidly to new circumstances (Henrich and McElreath 2003; Laland 2004). For example, fish are able to locate safe sources of food in new environments by observing where other members of their species congregate (Laland 2004). Beyond simple imitation, social learning allows individual humans to "stand on the shoulders of giants" and more easily develop sophisticated behaviors or innovations that would be impossible to discover from scratch within one lifetime. In fact, social learning may be a central feature of humans' unprecedented cognitive and technological development (Boyd, Richerson, and Henrich 2011; Humphrey 1976).

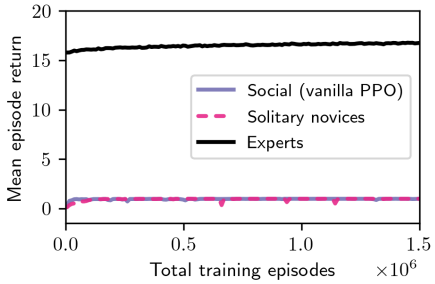
What if Reinforcement Learning (RL) agents could benefit from social learning? In many real-world environments where we would like to deploy RL agents (such as autonomous driving or robotics), individual exploration can be costly, inefficient, error-prone, or unsafe. However, these environments are filled with human experts who know how to perform tasks that we would like RL agents to learn, such as driving a car, or loading a dishwasher. Instead of requiring humans to explicitly train them, or to collect demonstration trajectories for imitation learning, RL agents could passively observe the behavior of people that happen to be present in the environment going about their daily tasks. Agents could follow social cues to adapt quickly to novel tasks and circumstances without resorting to clumsy or costly exploration. This ability would be particularly beneficial, given that current deep RL approaches often fail to generalize to new environments; performance deteriorates rapidly with even slight modifications to the training environment (Cobbe et al. 2019; Farebrother, Machado, and Bowling 2018; Packer et al. 2018).

We investigate whether model-free RL agents can benefit from cues in the behavior of experts present in the same multi-agent, partially observable environment (as illustrated in Figure 1a). In contrast to imitation learning, the novices have no direct access to expert trajectories, and the experts selfishly pursue their own goals with no incentive to teach the novices. We find that even in these simple environments, standard model-free RL agents often fail to benefit from the presence of experts (Figure 1b). To increase the potential for social learning, we develop a novel environment in which individual exploration is costly, and prestige cues help agents identify experts. However, even under these circumstances we find that model-free RL agents do not engage in social learning.

To understand this effect, we show that in sparse reward environments, model-free RL struggles to benefit from demonstrations that do not directly increase reward. To ameliorate this issue, we propose a new Social Learning with Auxiliary Prediction Loss (SociAPL) technique. Our empirical results show that SociAPL agents are able to benefit from the cues of experts and learn a generalized social learning policy, which allows them to achieve higher performance than agents trained alone, agents trained in the presence of experts without the



(a) The novice agent (red) is trying to navigate to the green goal, which is outside its partially observed view of the environment (outlined in red, shown in top right). When the expert (blue) turns toward the goal, this can act as a cue to help guide the novice.



(b) Vanilla model-free RL shows no performance benefit when agents are trained in the presence of experts, and cannot reach expert performance, indicating agents are not effective at social learning.

Figure 1: Although cues in expert behavior could help agents more rapidly complete an exploration task (a), we find that vanilla model-free RL agents do not appear to benefit from the presence of experts in their environment.

auxiliary loss, and even the experts themselves.

Crucially, we show that social learning improves generalization. When evaluated in novel environments with different experts, SociAPL agents use cues from the new experts to achieve high zero-shot transfer performance. They significantly out-perform agents trained alone and agents that do not use social learning, both of which struggle in the transfer task. To the best of our knowledge, this paper is the first to investigate the effect of social learning on the ability of RL agents to generalize to new environments.

To prevent social learners from becoming reliant on the presence of experts, we interleave training with experts and training alone. Social learners trained in this manner make use of what they learned from experts to improve their performance in the solo environment, even out-performing agents that were only ever trained solo. This shows that agents can acquire skills through social learning that they could not discover alone, and which are beneficial even when experts are no longer present.

The contributions of this work are: i) a new environment that promotes social learning through prestige cues and penalizing individual exploration; ii) an analysis of why model-free RL struggles to benefit from cues in expert behavior in sparse reward environments; iii) the SociAPL algorithm which we empirically demonstrate enables social learning and improved performance over solo training; and iv) zero-shot transfer results showing that agents which engage in social learning generalize effectively to new environments when other agents cannot.

Related Work

There is a rich body of work on imitation learning, which focuses on training an agent to closely approximate the behavior of a curated set of expert trajectories (Pomerleau 1989; Schaal 1999; Billard et al. 2008; Argall et al. 2009). For example, Behavioral Cloning (e.g. Bain and Sammut (1995); Torabi, Warnell, and Stone (2018)) uses supervised learning to imitate how the expert maps observations to actions. Because imitation learning can be brittle if the agent encounters a state that was not contained in the expert dataset, it can also be combined with RL (e.g. Guo et al. (2019); Lerer and Peysakhovich (2019)), or make use of multiple experts (e.g. Cheng, Kolobov, and Agarwal (2020)). Third-person imitation learning (e.g. Shon et al. (2006); Calinon, Guenter, and Billard (2007); Stadie, Abbeel, and Sutskever (2017)) enables agents to learn to copy an expert's policy when observing the expert from a third-person perspective.

Our work shares some similarities with third-person imitation learning, in that our agents are never given access to expert's observations and therefore only have a third-person view. However, our work is distinct from imitation learning. We do not assume access to a curated set of expert trajectories. Rather, our agents co-exist in a multi-agent environment with other agents with varying degrees of expertise, who are not always within their partially observed field of view. To gain access to other agents' expertise, novices may need to learn to actively follow them. The experts can selfishly pursue their own reward, and we do not assume experts are incentivized to teach other agents (as in e.g. Omidshafiei et al. (2019); Christiano et al. (2017)). Thus, we make fewer assumptions, and seek to design agents that can be applied to a broader range of naturalistic settings in which other agents (such as humans) are already present, but may not be willing to explicitly train our learning agent. In these settings, it is up to the novice agent to learn to associate cues in the expert's behavior with their own ability to obtain reward from the environment. However, we do not require novices to explicitly model other agents (as in e.g. Albrecht and Stone (2018); Jaques et al. (2018)). Finally, unlike in imitation learning, we do not force our agents to copy the experts' policy; in fact, we show that our agents eventually learn to out-perform the experts with which they learned.

Our work is distinct from Inverse RL (IRL) (e.g. Ng, Russell et al. (2000); Ramachandran and Amir (2007); Ziebart et al. (2008); Hadfield-Menell et al. (2016)), because our agents share a single environment with the experts which has a consistent reward function, and thus do not need to infer

the reward function from expert trajectories.[1] However, we do consider learning from sub-optimal experts, which was studied by Jacq et al. (2019) in the context of IRL.

Why should we expect social learning from (potentially sub-optimal) agents in the environment to be beneficial? Rendell et al. (2010) explored this question through a non-stationary multi-armed bandit tournament, in which agents had the option to copy the arm pulled by another agent. They found a strong correlation between the number of copy actions taken and success in the tournament, with the winning agent playing 'copy' on more than 90% of turns. Copying was so effective because each agent is attempting to play its best known strategy, so learning from others can be much more effective than random individual exploration.

The insight that copying can be a highly effective strategy was applied to the context of autonomous driving, where Landolfi and Dragan (2018) trained cars which copied the policy of human drivers on the road when the agent had a high degree of uncertainty. However, they assume access to privileged information about other cars' states and policies. Sun et al. (2019) make similar assumptions, but use cues from other cars to update agents' beliefs; e.g. their belief about the presence of pedestrians in occluded regions.

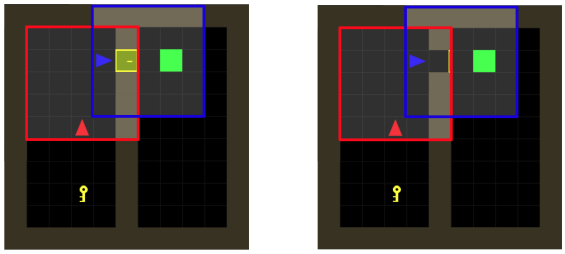
Most closely related to our work is that of (Borsa et al. 2019), who train model-free RL agents in the presence of a hand-coded (scripted) expert policy. In the partially observed exploration tasks they consider, they find that that novice agents learning from experts outperform solitary novices only when experts visibly make use of information that is hidden from the novices. In our work, experts do not have access to privileged information about the environment; rather they are distinguished by their task skill. We analyze why it is difficult for RL agents to benefit from expert demonstrations in sparse reward environments, and propose a method to solve it. Unlike Borsa et al. (2019), we show for the first time that social learning allows agents to transfer effectively to unseen environments with new experts, resulting in better generalization than solo learners.

Finally, our approach uses a model-based auxiliary loss, which predicts the next state given the current state, to better enable agents to learn transition dynamics from trajectories in which they received no reward. The idea of using auxiliary losses in deep RL has been explored in a variety of works (e.g. Ke et al. (2019); Weber et al. (2017); Shelhamer et al. (2016); Jaderberg et al. (2016)). Model-based RL (MBRL) has also been applied to the multi-agent context (e.g. Krupnik, Mordatch, and Tamar (2020)).

Background

We focus on Multi-Agent Partially Observable Markov Decision Process (MA-POMDP) environments defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{I}, N \rangle$, where N is the number of agents. The shared environment state is $s \in \mathcal{S}$. $\mathcal{I}$ is an inspection function that maps s to each agent's partially observed view of the world, s^k. At each timestep t, each agent k chooses a discrete action $a_t^k \in \mathcal{A}$. Agents act simultaneously and there is no notion of turn-taking. Let $\mathcal{A}^N$ be the joint action space, and $\vec{a}_t$ be the vector containing the actions of all agents for timestep t. The transition function depends on the joint action space: $\mathcal{T} : \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \rightarrow [0, 1]$. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the same across agents, but each agent receives its own individual reward $r_t^k = \mathcal{R}(s_t, a_t^k)$.

In this setting, each agent k is attempting to selfishly maximize its own, individual reward by learning a policy π^k that optimizes the total expected discounted future reward: $J(\pi^k) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}^k \mid s_0 \right]$, given a starting state s_0 and a discount factor $\gamma \in [0, 1]$. Note that agents are trained independently, cannot directly observe other agents' observations, actions, or rewards, and do not share parameters. To simplify notation, when we discuss the learning objectives for a single novice we forego the superscript notation.



(a) *Door Key* environment (b) Novel demonstration state $\tilde{s}$

Figure 2: An example environment in which the blue expert provides a demonstration of a difficult-to-reach state $\tilde{s}$.

Model-Free, Sparse-Reward Social Learning

To understand how experts present in the same environment could provide cues that enhance learning—and how model-free RL could fail to benefit from those cues—consider the example environment pictured in Figure 2. In this sparse-reward, hard exploration environment, the agents only receive a reward for reaching the green goal; the rest of the time, their reward is 0. To reach the goal, they must first pick up a key, and use it to pass through the door. Assume there is an infinite supply of keys, and after an agent passes through the door it closes. An expert agent can provide a demonstration of a novel state $\tilde{s}$ that is difficult to produce through random exploration: that it is possible to open the door. Ideally, we would like novice agents to learn from this demonstration by updating their internal representation to model $\tilde{s}$.

However, if the novice does not receive any reward as a result of observing $\tilde{s}$, model-free RL receives little benefit from the expert's behavior. Assume that the novice agent is learning policy π_θ with a policy-gradient objective on a trajectory including the demonstrated state $\tilde{s}_k, k \in (0, T-1)$:

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) R_t \tag{1}$$

where $R_t = \sum_{t'=t+1}^{T} \gamma^{t'-t-1} r_{t'}$ is the total reward received over the course of the trajectory. If the agent receives 0 reward during the episode in which the demonstration occurred

[1] Note that novices and experts do not share rewards; if an expert receives a reward for completing the task the novice does not benefit.

(e.g. it does not reach the goal), we can see that $\forall t \in (0, T), R_t = \sum_{t'=t+1}^{T} \gamma^{t'-t-1} r_{t'}(0) = 0$. Therefore $\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(0) = 0$, and the novice receives no gradients from the expert's demonstration which allow it to update its policy.

Temporal Difference (TD) Learning could temporarily mitigate this issue, but as we show in detail in the appendix, this ability quickly deteriorates. Consider Q-learning, in which $Q(a, s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \,|\, a, s\right]$ models the total expected reward from taking action a in state s. As the agent continues to receive 0 rewards during training, all Q values will be driven toward zero. Even if the agent observes a useful novel state such as $\tilde{s}$, as $Q(a, s) \to 0, \forall a \in \mathcal{A}, s \in \mathcal{S}$, the Q-learning objective becomes:

$$Q(\tilde{s}, a) = r + \gamma \max_{a'} Q(s', a') = 0 + \gamma 0 = 0 \tag{2}$$

Thus, the objective forces the value of $Q(\tilde{s}, a)$ to be zero. In this case, modeling transitions into or out of state $\tilde{s}$ is not required in order to produce an output of zero, since all other Q-values are already zero.

In both cases, we can see that before a model-free RL agent receives a reward from the environment, it will have difficulty modeling novel states or transition dynamics. This makes learning from the cues of experts particularly difficult. Learning to model the expert's policy $\pi^E(a^E|s^E)$ would likely help the novice improve performance on the task. However, the novice does not have explicit access to the expert's states or actions. From its perspective, the other agent's policy is simply a part of the environment transition dynamics. While the true state transition function $s_{t+1} = \mathcal{T}(s_t, a_t^N, a_t^E)$ depends on both the novice's own action a_t^N, and the expert's policy (since $a_t^E = \pi^E(s_t^E)$), the novice is only able to observe $p(s_{t+1}^N|s_t^N, a_t^N)$. Therefore, the novice can only obtain knowledge of the expert's policy through correctly modeling the state transitions it observes. Since, as we have argued above, the novice will struggle to model state transitions in the absence of external reward, it will also have difficulty modeling another agent's policy.

SociAPL

To mitigate this issue, we propose augmenting the novice agent with a model-based prediction loss. Specifically, we append additional layers θ_A to the policy network's encoding of the current state, $f_\theta(s_t)$, as shown in Figure 3. We then introduce an unsupervised mean absolute error (MAE) auxiliary loss to train the network to predict the next state s_{t+1} given the current state s_t and the agent's current action a_t:

$$\hat{s}_{t+1} = f_{\theta_A}(a_t, s_t) \tag{3}$$

$$J = \frac{1}{T} \sum_{t=0}^{T} |s_{t+1} - \hat{s}_{t+1}| \tag{4}$$

This architecture allows gradients from the auxiliary loss to contribute to improving $f_\theta(s_t)$. Figure 4 shows example state predictions generated by the auxiliary layers for a SociAPL agent, demonstrating that this architecture enables effectively learning of the transition dynamics.

We can now see that if the novel demonstration state is in a trajectory, $\tilde{s}_k \in (0, T)$, the term $|\tilde{s}_k - \hat{s}_k|$ will be part of the objective. It will not be 0 unless the agent learns to perfectly predict the novel demonstration state. Therefore, cues from the expert will provide gradients that allow the novice to improve its representation of the world, even if it does not receive any reward from the demonstration. This architecture also implicitly improves the agent's ability to model other agents' policies, since it must correctly predict other agents' actions in order to accurately predict the next state. It is able to do this without ever being given explicit access to the other agent's states or actions. We call our approach Social learning with Auxiliary Predictive Loss (SociAPL), and hypothesize that it will improve agents' ability to learn from the cues of experts.

Optimization and architecture

To optimize our agents, we test both TD-learning (deep Q-learning) and policy-gradient methods, and find that Proximal Policy Optimization (PPO) (Schulman et al. 2017) provides better performance and stability, and is most able to benefit from social learning. We use Generalized Advantage Estimation (GAE) (Schulman et al. 2016) to train the PPO value function.

As shown in Figure 3, our agents use convolution layers to learn directly from a pixel representation of the state s_t. Because the environments under investigation are partially observed and non-Markov, we use a recurrent policy parameterized by a Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber 1997) to model the history of observations in each episode. LSTM hidden states are stored in the experience replay buffer. Following the recommendations of Andrychowicz et al. (2020), we recalculate the stored state advantage values used to compute the value function loss between mini-batch gradient updates. In addition, we recalculate and update the stored hidden states as suggested by Kapturowski et al. (2018) for off-policy RL. A detailed description of the network architecture and hyperparameters used in our experiments are given in an appendix.

Baselines and ablations

Our experiments compare training SociAPL agents (which learn in the presence of experts in the same environment) to agents which use the same architecture and prediction loss as in Figure 3, but train alone. We call these agents Solo APL. We also compare to two ablations of the model. The first uses the same architecture with no auxiliary loss, and can be considered a vanilla PPO implementation. The second replaces the next-state-prediction objective of Equation 4 with a simple autoencoder reconstruction objective that predicts the *current* state, $\hat{s}_t = f_{\theta_A}(a_t, s_t)$. The loss is thus $J = \frac{1}{T} \sum_{t=0}^{T} |s_t - \hat{s}_t|$. In this case the network must represent the current state, but does not need to learn about the transition dynamics. We refer to this Social Auxiliary Reconstruction Loss agent as SociARL.

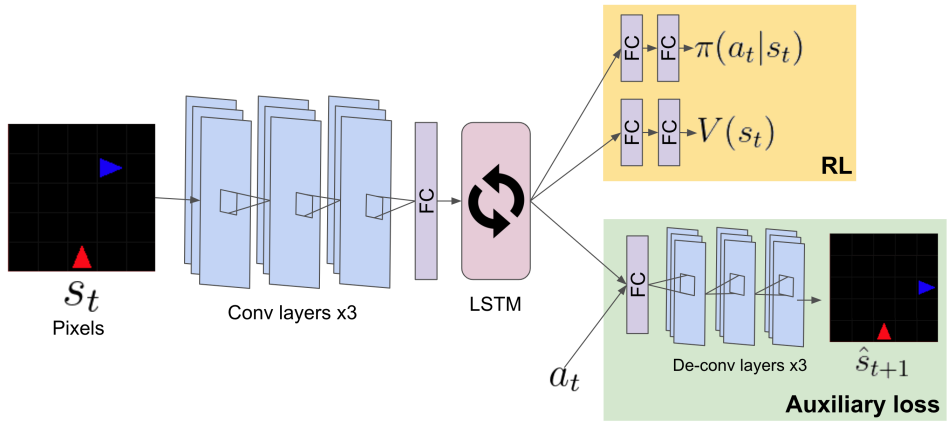


Figure 3: SociAPL deep neural network architecture. Convolution (conv) layers extract information about the state from pixels, which is fed into a Fully Connected (FC) layer, and a recurrent LSTM. The yellow shaded box shows the components of the model which learn the RL policy π and value function $V(s)$. The green shaded box shows the components of the model dedicated to computing the auxiliary loss, which predicts the next state given the current state.

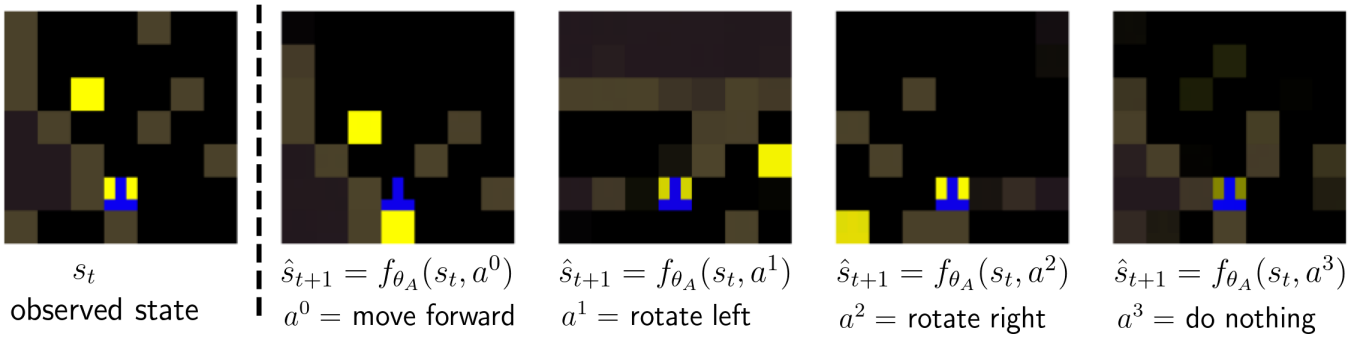


Figure 4: Examples of future states $\hat{s}_{t+1}$ predicted by the network given state s_t ("observed state"), and each of the possible movement actions a_t. Most predictions are highly accurate, indicating the network has learned to effectively model transition dynamics. The transition for the 'do nothing' action is less accurate because it is infrequently chosen by the agent.

Social Learning Environments

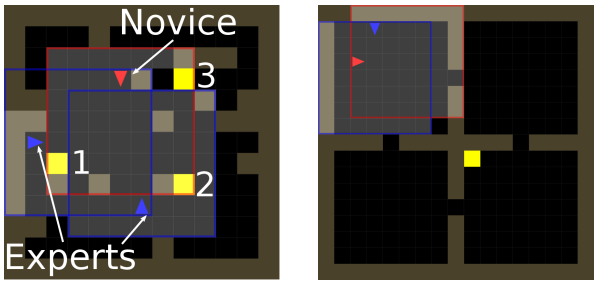
Humans and animals are most likely to rely on social learning when individual learning is too difficult or unsafe (Henrich and McElreath 2003; Laland 2004). Further, individuals prefer to learn from others that they perceive to be highly successful or competent, which is known as *prestige bias* (Jiménez and Mesoudi 2019). Cues or signals associated with prestige have shown to be important to both human and animal social learning (Barkow et al. 1975; Horner et al. 2010).

Motivated by these two ideas, we introduce a novel environment specifically designed to encourage social learning by making individual exploration difficult and expensive, and introducing prestige cues. In Goal Cycle (Figure 5a), agents must navigate between several goal tiles, and are rewarded for navigating between the goals in a certain order and penalized for deviating from that order. The goal tiles are placed randomly and are visually indistinguishable, so it is not possible for an agent to identify the correct traversal order without potentially incurring an exploration penalty. When the penalty is large, this becomes a hard exploration task. Each agent changes color as it collects rewards over the course of each episode, making color a cue to agents' success or prestige.

Agent observations consist of egocentric images of nearby tiles, and they can see themselves in their partial views. As an agent accrues rewards over a single episode, it changes from red to blue. The agent's color resets to red if it incurs a penalty. Thus, upon reaching a goal, an agent can determine whether its traversal order was correct by observing its own color. The colour of other agents acts as a signal of their competence (or prestige cue), helping identify other agents with the most expertise. The prestige c_{t+1} depends on the previous prestige c_t and reward r_t:

$$c_{t+1} = \begin{cases} \alpha_c c_t + r_t, & r_t \geq 0 \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where the decay constant α_c determines how quickly prestige decays absent new rewards. For our experiments we used $\alpha = 0.99$. Agent colors in goal cycle are determined using the squashed prestige value to interpolate between the red



(a) Goal Cycle (3 goals) (b) Four Rooms

Figure 5: Goal Cycle (a) is a 13x13 environment in which the positions of goal tiles (yellow) and obstacles (brown) are randomized at the start of each episode. Agents receive rewards for traversing the goal tiles in a certain order (0 $\rightarrow$ 1 $\rightarrow$ 2 $\rightarrow$ 0 $\rightarrow$ ...) and penalties for deviations from the correct order. However, the correct order is not visible. Agents always receive a reward from the first goal tile they encounter in an episode. We test whether agents trained in Goal Cycle can generalize to Four Rooms (b), a larger, 17x17 environment with walls hiding the location of a single goal.

and blue RGB color vectors.

$$\text{COLOR}_t = \text{BLUE} \cdot \tilde{c}_t + \text{RED} \cdot (1 - \tilde{c}_t), \quad (6)$$

where $\tilde{c}_t = \text{sigmoid}(c_t)$.

Agents persist in the environment for a fixed duration. During each episode skilled agents discover the goal locations and correct traversal order while incurring minimal penalties, then achieve high rewards by cycling between the goals for the remainder of the episode. It is not possible to identify the correct cycle by observing the goal tiles directly; however, this information can be inferred by observing other agents. In practice, since the behavior of other agents can be unpredictable and potentially non-stationary, agents more easily learn to solve the task directly through trial and error. But by adjusting the penalty for navigating to goals in the wrong order, we can disincentivize individual exploration and thereby encourage social learning. Thus, we can directly control how strongly agents are incentivized to attend to one another. In all the Goal Cycle variants discussed here, agents receive a reward of +1 for navigating to the first goal they encounter in an episode, and +1 for any navigating to any subsequent goals if in the correct order. They receive a penalty of -1.5 for navigating to the incorrect goal. The Goal Cycle environments are available in the open-source Marlgrid project (Ndousse 2020).

Training Expert Agents

The Goal Cycle environments are challenging to master because there are many more incorrect than correct goal traversal sequences. When the penalty is large, the returns for most traversal orders are negative, so mistake-prone agents learn to avoid goal tiles altogether (after the first goal in an episode, which always gives a reward of +1). However, we can obtain agents that perform well in high penalty environments using a curriculum: we train them initially in low-penalty environments and gradually increase the penalty magnitude.

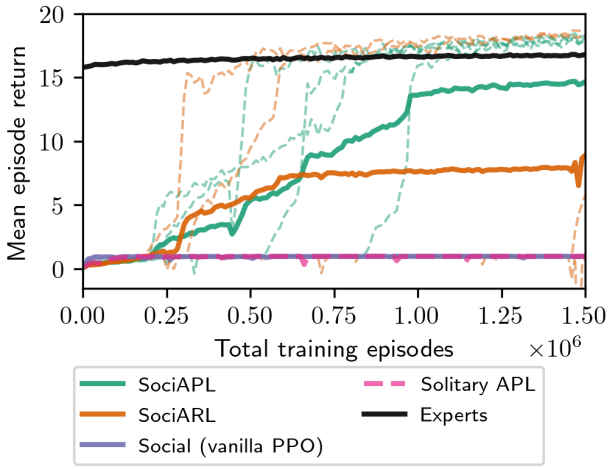


Figure 6: For social learning in the presence of experts, only agents with an auxiliary loss that enables them to model expert's cues (SociAPL, SociARL) succeed in using social learning to improve performance. Faded dotted lines show the performance of individual random seeds and bolded lines show the mean of 5 seeds. The final performance is bimodal, with some novice seeds achieving expert-level performance and others failing to learn entirely. Since the normality assumption is violated, we refrain from using confidence intervals. None of the seeds for solo agents or social vanilla PPO are able to solve the task. In contrast, 4 of 5 SociAPL seeds are able to exceed the performance of the experts present in their environment.

Transfer Environments

We test the zero-shot transfer performance of agents pre-trained in 3-Goal Cycle in two new environments. The first is a Goal Cycle environment with 4 goals. This variant is significantly harder, because while there are two possible cycles between three goals, there are six ways to cycle between four goals. Thus, even optimal agents must incur higher penalties to identify the correct traversal order in each episode.

We also test transfer to the classic Four Rooms environment (shown in Figure 5b). Here agents must locate a single goal tile within a time limit. The reward for navigating to the goal is +1 at the beginning of each episode and decreases linearly until reaching 0 when the episode ends. The goal is placed randomly in one of the sixteen corner tiles. This 17x17 environment is significantly larger, and agents are unlikely to have encountered walls while training in Goal Cycle.

Results

Figure 6 compares Goal Cycle performance of agents trained alone (solo) to agents trained in the presence of experts (social). Solo APL agents are unable to discover the strategy of reaching the goals in the correct order; instead, all random seeds converged to a strategy of stopping after reaching a single goal (with a maximum return of 1). Social vanilla PPO agents failed in the same way, with all seeds receiving a maximum reward of 1. This supports our hypothesis that

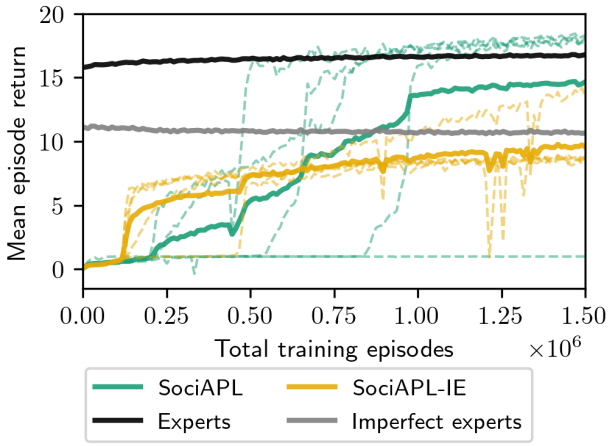


Figure 7: The effect of expert skill on social learning. With the proposed APL, novice agents benefit from social learning in the presence of near-optimal experts (SociAPL) or imperfect experts (SociAPL-IE), surpassing the performance of solo agents in both cases. However, SociAPL-IE agents achieve lower performance, and only 1/5 random seeds exceed the performance of the imperfect experts.

model-free RL has difficulty benefiting from the cues of experts in sparse reward environments.

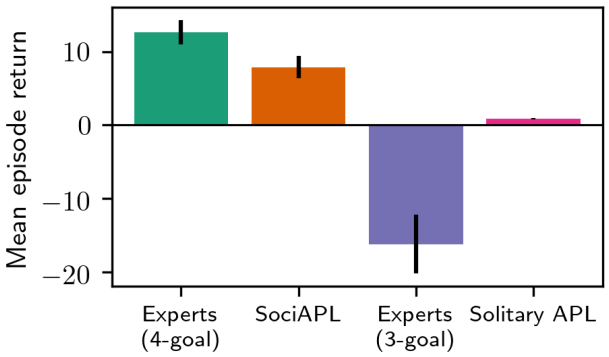
In contrast, social agents trained with an auxiliary loss to predict the next state (SociAPL) or reconstruct the current state (SociARL) were both able to achieve higher performance. While the reconstruction loss helps SociARL agents learn from experts, still higher performance is obtained with the SociAPL model-based prediction loss. This is likely because in contrast with the reconstruction loss, the model-based loss helps agents learn about both the transition dynamics and the other agent's policy. The majority of SociAPL random seeds are actually able to exceed the performance of the expert with which they are trained. Note that the solo agents use the same auxiliary prediction loss as the SociAPL agents, showing that good performance depends on learning effectively from expert cues.

Learning from Sub-Optimal Experts

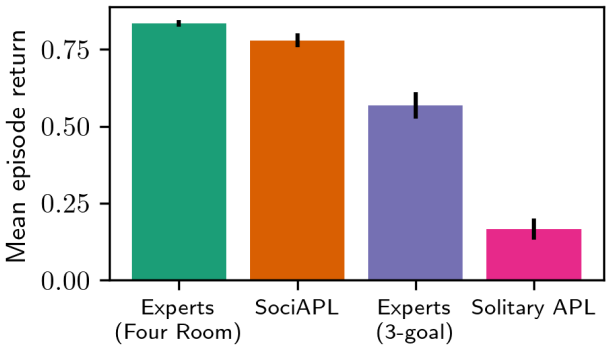
In real-world environments, not all of the agents that it is possible to observe will have optimal performance. Therefore, we test whether it is still possible to learn from imperfect experts (SociAPL-IE), and present the results in Figure 7. While the performance of the expert does affect the novices' final performance, even when learning from imperfect experts novices can use social learning to exceed the performance of agents trained alone. However, we find that novices trained with optimal experts more frequently learn to exceed the performance of the experts with which they are trained.

Transfer to New Environments

A central thesis of this work is that social learning can help agents adapt more rapidly to novel environments. This could be particularly useful because deep RL often fails to generalize



(a) Goal Cycle (4 goals, with experts)



(b) Four Rooms (with expert)

Figure 8: Zero-shot transfer performance in novel environments with expert demonstrators. Each bar shows the mean of the best three out of five seeds, and each seed was evaluated at five consecutive model checkpoints (128 episodes per checkpoint) starting after 1.5 million training episodes. The experts and 3-goal solo learners were also trained with APL. Error bars show 95% confidence intervals for the estimated means. Agents which have learned to rely on social learning benefit from the cues of novel experts in the new environment. They easily out-perform agents trained alone as well as the experts from which they originally learned. Videos: tinyurl.com/SociAPL

to even slight modifications of the training environment. Therefore, we investigate how well agents pre-trained in 3-Goal Cycle can generalize to two new environments: 4-Goal Cycle and Four Rooms.

Figure 8a shows the zero-shot transfer performance of agents trained in 3-Goal Cycle to 4-Goal Cycle and a new set of experts. Agents trained alone (which have not learned to rely on social learning) perform poorly. In contrast, SociAPL agents show evidence of having learned a generalized social learning policy, in that they are able to learn from the cues of a new set of experts and maintain high performance in the new environment. Note that agents trained as experts in the 3-goal environment receive large negative scores, repeatedly incurring a penalty for navigating to the goals in the incorrect order. This illustrates the brittleness of an RL policy which

is overfit to the training environment.

Figure 8b shows the zero-shot transfer performance of the same agents in the Four Rooms environment, which is significantly larger and has sparser rewards. Despite these differences, expert agents from the 3-Goal Cycle environment perform moderately well. However, the SociAPL agents not only significantly out-perform agents trained alone, but are actually able to outperform the 3-Goal Cycle experts with which they were trained. This is because they can benefit from the cues of Four Rooms experts to more closely approximate the optimal policy. Once again, we see that a generalized social learning policy is better able to transfer to a new environment than an expert overfit to the training environment. Taken together, these results suggest social learning can enable RL agents to benefit from the cues of experts present in their environments, enhancing their ability to adapt to changes between the training and test environments.

Removing the Experts

A potential downside to learning to rely on the cues of experts is that agents could fail to perform well when experts are removed from the environment. For novice agents always trained with experts, there appears to be a trade-off between social learning and individual learning. As shown in Figure 9, novice SociAPL agents initially learn to solve the Goal Cycle task with individual exploration, but eventually they overfit to the presence of experts and become reliant on expert cues to the detriment of solo performance. Observing the agent's behavior reveals that it has learned to follow the cues of experts when they are present in the environment, but refrain from individual exploration when experts are not present. Given the high cost of individual exploration in this environment, this a safe but conservative strategy.

We find that we can improve the solo performance of SociAPL by changing the distribution of training tasks to include episodes in which the experts are not present. The dashed lines in Figure 9 show the transfer performance of a checkpoint transferred from entirely social environments to a mix of solo and social environments as training proceeds. The agent is able to retain good performance in solo 3-goal environments as well as 4-goal environments with experts, indicating that it is learning to opportunistically take advantage of expert cues while building individual expertise in the 3-goal environment. In fact, the performance of SociAPL in the solo environment is higher than agents exclusively trained in the solo environment (as shown in Figure 6). This demonstrates that not only does social learning enable agents to discover skills that they could not learn by themselves, but that they are able to retain this knowledge to improve their individual performance even when experts are not present.

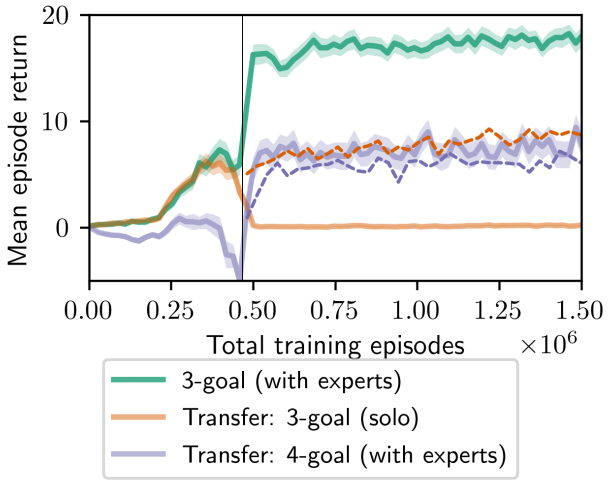


Figure 9: **Solid lines** Transfer performance throughout training for one of the SociAPL seeds from Figure 6. The green line shows performance in environments sampled from the training distribution, and the solid purple and orange lines show the zero-shot transfer performance in solo 3-Goal Cycle, and 4-Goal Cycle with experts, respectively. The agent initially relies mostly on individual exploration as indicated by good zero-shot transfer performance in the solo environment. Zero-shot performance in the 4-Goal Cycle environment with experts is initially poor, since the individual 3-Goal policy is ineffective in the 4-Goal variant. After about 500k episodes the agent becomes reliant on cues from the 3-Goal experts. This allows it to perform well in 4-Goal, but its performance suffers in 3-Goal. **Dashed lines** Transfer performance of the same agent if the distribution of training environments is changed from 100% social to 75% solo and 25% social after about 500k episodes. As training proceeds the agent retains the capacity to solve the solo three-goal environment while learning to use cues from expert behavior when they are available. The performance of this agent in the solo environment actually exceeds that of agents trained exclusively in the solo environment.

Conclusions

In this work we have investigated whether model-free deep RL is able to benefit from the presence of expert agents in the environment. We find that in sparse reward environments, model-free agents fail to use expert cues to obtain optimal performance. However, by adding a model-based auxiliary loss which requires modeling transition dynamics in the absence of any rewards, we are able to train SociAPL agents to use social learning. When deployed to novel environments, these agents retain their ability to benefit from the cues of experts, and perform well in zero-shot transfer tasks with experts when agents trained alone cannot. Further, by mixing social and solo training, we obtain social learning agents that actually have higher performance in the solo environment than agents trained exclusively in the solo environment. Our results demonstrate that social learning not only enables agents to learn complex behavior that they do not discover when trained alone, but that it can allow agents to achieve good performance when transferred to novel environments.

Limitations and Future Work

Our social learning experiments focus on exploratory navigation tasks, so agents learn to follow experts as an effective

social learning strategy. Compelling directions for future work include extending this work to other domains such as manipulation, to scenarios in which expert agents pursue different goals than the novices, and to scenarios with multiple experts that employ a variety of strategies. We demonstrate that training in a mixture of social and solitary environments can permit novice SociAPL agents to develop effective strategies for both social and individual task variants, and notably that the resulting individual skill far exceeds that of a solitary novice. However, in this work we do not thoroughly explore different strategies for augmenting solitary with social experience. Further research could clarify the circumstances in which adding social experiences could aid solitary task performance, and see the development of algorithms to facilitate this for arbitrary tasks.

Acknowledgements

We are grateful to the OpenAI Scholars program for facilitating the collaboration that led to this paper. We would also like to thank Ken Stanley, Joel Lehman, Bowen Baker, Yi Wu, Aleksandra Faust, Max Kleiman-Weiner, and DJ Strouse for helpful discussions related to this work, and Alethea Power for providing compute resources that facilitated our experiments.

References

Albrecht, S. V.; and Stone, P. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence* 258: 66–95.

Andrychowicz, M.; Raichuk, A.; Stańczyk, P.; Orsini, M.; Girgin, S.; Marinier, R.; Hussenot, L.; Geist, M.; Pietquin, O.; Michalski, M.; Gelly, S.; and Bachem, O. 2020. What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study. *arXiv preprint arXiv:2006.05990* .

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57(5): 469–483.

Bain, M.; and Sammut, C. 1995. A Framework for Behavioural Cloning. In *Machine Intelligence 15*, 103–129.

Barkow, J. H.; Akiwowo, A. A.; Barua, T. K.; Chance, M.; Chapple, E. D.; Chattopadhyay, G. P.; Freedman, D. G.; Geddes, W.; Goswami, B.; Isichei, P.; et al. 1975. Prestige and culture: a biosocial interpretation [and comments and replies]. *Current Anthropology* 16(4): 553–572.

Biewald, L. 2020. Experiment Tracking with Weights and Biases. URL https://www.wandb.com/. Software available from wandb.com.

Billard, A.; Calinon, S.; Dillmann, R.; and Schaal, S. 2008. Survey: Robot programming by demonstration. *Handbook of robotics* 59(BOOK_CHAP).

Borsa, D.; Heess, N.; Piot, B.; Liu, S.; Hasenclever, L.; Munos, R.; and Pietquin, O. 2019. Observational learning by reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 1117–1124. International Foundation for Autonomous Agents and Multiagent Systems.

Boyd, R.; Richerson, P. J.; and Henrich, J. 2011. The cultural niche: Why social learning is essential for human adaptation. *Proceedings of the National Academy of Sciences* 108(Supplement_2): 10918–10925.

Calinon, S.; Guenter, F.; and Billard, A. 2007. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37(2): 286–298.

Cheng, C.-A.; Kolobov, A.; and Agarwal, A. 2020. Policy Improvement from Multiple Experts. *arXiv preprint arXiv:2007.00795* .

Chevalier-Boisvert, M.; Willems, L.; and Pal, S. 2018. Minimalistic Gridworld Environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid.

Christiano, P. F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 4299–4307.

Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; and Schulman, J. 2019. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, 1282–1289.

Farebrother, J.; Machado, M. C.; and Bowling, M. 2018. Generalization and regularization in DQN. *arXiv preprint arXiv:1810.00123* .

Guo, X.; Chang, S.; Yu, M.; Tesauro, G.; and Campbell, M. 2019. Hybrid reinforcement learning with expert state sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3739–3746.

Hadfield-Menell, D.; Russell, S. J.; Abbeel, P.; and Dragan, A. 2016. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, 3909–3917.

Henrich, J.; and McElreath, R. 2003. The evolution of cultural evolution. *Evolutionary Anthropology: Issues, News, and Reviews* 12(3): 123–135.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780.

Horner, V.; Proctor, D.; Bonnie, K. E.; Whiten, A.; and de Waal, F. B. 2010. Prestige affects cultural learning in chimpanzees. *PloS one* 5(5): e10625.

Humphrey, N. K. 1976. The social function of intellect. In *Growing points in ethology*, 303–317. Cambridge University Press.

Jacq, A.; Geist, M.; Paiva, A.; and Pietquin, O. 2019. Learning from a Learner. In *International Conference on Machine Learning*, 2990–2999.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* .

Jaques, N.; Lazaridou, A.; Hughes, E.; Gulcehre, C.; Ortega, P. A.; Strouse, D.; Leibo, J. Z.; and de Freitas, N. 2018.

Intrinsic social motivation via causal influence in multi-agent RL. *arXiv preprint arXiv:1810.08647* .

Jiménez, Á. V.; and Mesoudi, A. 2019. Prestige-biased social learning: current evidence and outstanding questions. *Palgrave Communications* 5(1): 1–12.

Kapturowski, S.; Ostrovski, G.; Quan, J.; Munos, R.; and Dabney, W. 2018. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*.

Ke, N. R.; Singh, A.; Touati, A.; Goyal, A.; Bengio, Y.; Parikh, D.; and Batra, D. 2019. Learning dynamics model in reinforcement learning by incorporating the long term future. *arXiv preprint arXiv:1903.01599* .

Kingma, D.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* .

Krupnik, O.; Mordatch, I.; and Tamar, A. 2020. Multi-Agent Reinforcement Learning with Multi-Step Generative Models. In *Conference on Robot Learning*, 776–790.

Laland, K. N. 2004. Social learning strategies. *Animal Learning & Behavior* 32(1): 4–14.

Landolfi, N. C.; and Dragan, A. D. 2018. Social Cohesion in Autonomous Driving. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8118–8125. IEEE.

Lerer, A.; and Peysakhovich, A. 2019. Learning existing social conventions via observationally augmented self-play. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 107–114.

Ndousse, K. 2020. Marlgrid. https://github.com/kandouss/marlgrid.

Ng, A. Y.; Russell, S. J.; et al. 2000. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, 2.

Omidshafiei, S.; Kim, D.-K.; Liu, M.; Tesauro, G.; Riemer, M.; Amato, C.; Campbell, M.; and How, J. P. 2019. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6128–6136.

Packer, C.; Gao, K.; Kos, J.; Krähenbühl, P.; Koltun, V.; and Song, D. 2018. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282* .

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Pomerleau, D. A. 1989. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, 305–313.

Ramachandran, D.; and Amir, E. 2007. Bayesian Inverse Reinforcement Learning. In *IJCAI*, volume 7, 2586–2591.

Rendell, L.; Boyd, R.; Cownden, D.; Enquist, M.; Eriksson, K.; Feldman, M. W.; Fogarty, L.; Ghirlanda, S.; Lillicrap, T.; and Laland, K. N. 2010. Why copy others? Insights from the social learning strategies tournament. *Science* 328(5975): 208–213.

Schaal, S. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences* 3(6): 233–242.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations (ICLR)*.

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

Shelhamer, E.; Mahmoudieh, P.; Argus, M.; and Darrell, T. 2016. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307* .

Shon, A.; Grochow, K.; Hertzmann, A.; and Rao, R. P. 2006. Learning shared latent structure for image synthesis and robotic imitation. In *Advances in neural information processing systems*, 1233–1240.

Stadie, B. C.; Abbeel, P.; and Sutskever, I. 2017. Third-person imitation learning. *arXiv preprint arXiv:1703.01703* .

Sun, L.; Zhan, W.; Chan, C.-Y.; and Tomizuka, M. 2019. Behavior planning of autonomous cars with social perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, 207–213. IEEE.

Torabi, F.; Warnell, G.; and Stone, P. 2018. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954* .

Weber, T.; Racanière, S.; Reichert, D. P.; Buesing, L.; Guez, A.; Rezende, D. J.; Badia, A. P.; Vinyals, O.; Heess, N.; Li, Y.; et al. 2017. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203* .

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, 1433–1438. Chicago, IL, USA.

Appendix

Additional results

Figure 10 shows the training curve for the next-state auxiliary prediction loss of Equation 4, for the five SociAPL agent seeds plotted in Figure 6. The curve shows that the agent is effectively learning to predict the next state with low mean absolute error. However, because the agent's policy is changing at the same time, the prediction problem is non-stationary, which means that the loss does not always decrease. If the agent discovers a new behavior, the model will be required to predict new state transitions not previously experienced.

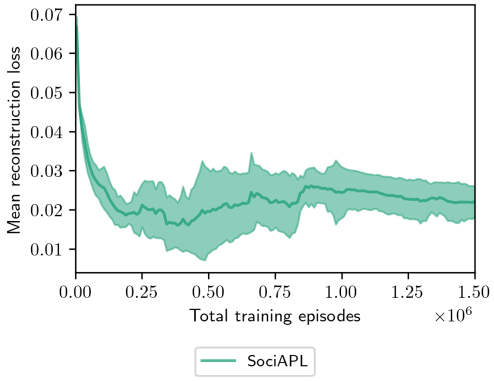


Figure 10: Next-state auxiliary prediction loss (in Mean Absolute Error (MAE) over the course of training the SociAPL agents shown in Figure 6.

Q-learning learning in sparse reward environments

Before a Temporal Difference (TD) learning agent has received any reward, it will be difficult for it to learn to model transition dynamics. Consider as an example deep Q-learning, in which the Q-function is parameterized by a neural network which encodes the state using a function $f_\theta(s)$. Assume the network is randomly initialized such that all Q-values are small, random values; *i.e.* $\forall a \in \mathcal{A}, s \in \mathcal{S}, Q(a,s) = \epsilon \sim \mathcal{N}(0, 0.1)$. Assume that the agent has not yet learned to navigate to the goal, and has received zero rewards so far during training. Therefore, when the agent observes the experience $(s, a, r = 0, s')$, the Q-learning objective is:

$$J(\theta) = (r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2 \quad (7)$$

$$= (0 + \gamma\epsilon_1 - \epsilon_2)^2 \quad (8)$$

In effect, this induces a correlation between $Q(s', a')$ and $Q(s, a)$, and consequently $f_\theta(s')$ and $f_\theta(s)$, as a result of observing the state transition $s \to s'$. However, as the agent continues to receive zero reward, all Q-values will be driven toward zero. Once this occurs, even if the agent observes a useful novel state such as $\tilde{s}$, our equation becomes:

$$(r + \gamma \max_{a'} Q(s', a') - Q(\tilde{s}, a))^2 = (0 - Q(\tilde{s}, a))^2 \quad (9)$$

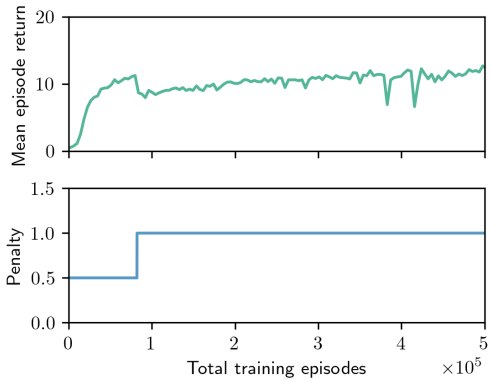


Figure 11: We use a penalty curriculum to obtain experts in environments where exploration is expensive. In the scenario visualized here, an agent trained for 81920 episodes in Goal Cycle environments with penalty of 0.5, then continued with a penalty of 1.0

such that the objective forces the value of $Q(\tilde{s}, a)$ to be zero. In this case, modeling transitions into or out of state $\tilde{s}$ is not required in order to produce an output of zero, since all other Q-values are already zero.

Environment details

The environments used in this paper were originally based on Minigrid (Chevalier-Boisvert, Willems, and Pal 2018). The partial states constituting agent observations are 27×27 RGB images corresponding to 7×7 grid tiles. There are 7 possible actions, though only three actions (rotate left, rotate right, move forward) are relevant in the experiments discussed in this paper. Agents are unable to see or move through the obstructive tiles that clutter their environments, and obstructed regions of their partial views appear as purple. However, agents can both see and move through goal tiles as well as other agents.

When the penalty for individual exploration in Goal Cycle environments is large, agents are unable to learn effective strategies. We used a penalty curriculum to obtain experts for such tasks as shown in Figure 11.

Network architecture details

The value, policy, and auxiliary task networks share three convolution layers, a fully connected layer, and an LSTM layer. Values and policies are computed with two fully connected layers, and the prediction-based auxiliary branch has a fully connected layer followed by transposed convolution layers that mirror the input convolution layers. The convolution and transposed convolution use leaky ReLU activation functions; all other layers use tanh activation functions.

- Shared input layers:
  - Conv $(3, 32)$ 3x3 filters, stride 3, padding 0,
  - Conv $(32, 64)$ 3x3 filters, stride 1, padding 0,
  - Conv $(64, 64)$ 3x3 filters, stride 1, padding 0,
  - FC $(576, 192)$,

  - LSTM $(192, 192)$
- Value MLP:
  - FC $(192, 64)$,
  - FC $(64, 64)$,
  - FC $(64, 1)$
- Policy MLP:
  - FC $(192, 64)$,
  - FC $(64, 64)$,
  - FC $(64, 7)$
- Auxiliary state prediction layers:
  - FC $(192 + 7, 576)$,
  - DeConv $(64, 64)$ 3x3 filters, stride 1, padding 0,
  - DeConv $(64, 32)$ 3x3 filters, stride 1, padding 0,
  - DeConv $(32, 3)$ 3x3 filters, stride 3, padding 0

Altogether there are 668555 parameters. We experimented with smaller (357291-parameter) networks but did not observe a significant performance difference. The networks were sized to roughly saturate the available (desktop) compute resources.

Hyperparameters

Each agent uses a single Adam optimizer (Kingma and Ba 2014) to update its parameters. Each of the novice agents was trained with a learning rate of $1e-4$. For SociAPL, the expert agents were trained with a learning rate of $1e-5$. Weights for all agents in the generalization experiments as well as the imperfect experts in SociAPL-IE were kept frozen and not updated.

Each parameter update consists of 20 sequential mini-batch updates with the same batch of rollouts (128 episodes). Each mini-batch consists of a uniform random sample of trajectories from that batch. Hidden states are stored alongside the trajectories, and the initial hidden state for each mini-batch trajectory is retrieved from these stored values. Hidden states and advantage values for the entire batch are re-calculated every 2 mini-batches. The mini-batch iteration ceases if $\mathrm{KL}(\pi, \pi^{rollout})$ exceeds a target of 0.01. If for any mini-batch the estimated divergence exceeds a hard limit of 0.03, the update terminates and all changes to the network parameters and optimizer state are reverted.

batch size	128 episodes
mini-batches per batch	20
mini-batch num trajectories	512
mini-batch trajectory length	16
hidden state/advantage update interval	2 minibatches
return discount γ	0.993
GAE-λ	0.97
PPO clip ratio	0.2
KL target	0.01
KL hard limit	0.03

For each mini-batch iteration, the loss used to update agent parameters is

$$L^{\text{total}} = L^{\pi}(\theta) + c_V \cdot L^{V}(\theta) + c_{\text{aux}} \cdot L^{\text{aux}}(\theta) - c_{\text{ent}} \cdot L^{\text{ent}}(\theta),$$

where the policy loss L^{π} is computed with PPO-clip (Schulman et al. 2017) and GAE (Schulman et al. 2016), the value loss L^{V} is the mean squared error of the values estimated for each step in the trajectory, $L^{\text{ent}}(\theta)$ is the policy entropy bonus, and $L^{\text{aux}}(\theta)$ is the auxiliary prediction loss (Equation 4). The loss scaling coefficients used in our experiments are $c_V = 0.1$, $c_{\text{ent}} = 1e-5$, and $c_{\text{aux}} = 3$.

The prestige decay constant α_c used for the Goal Cycle environments (i.e. Equation 5) was 0.99.

In general we sought hyperparameters that enable stable training. We experimented with mini-batch sizes varying from 32 to 1025 trajectories and found training to be more stable with larger mini-batches. Training was less stable with learning rates higher than $1e-4$.

We randomized seeds for both the network parameter initialization and environment generation for each trial of each experiment.

Compute

The experiments in this paper were performed primarily on a desktop computer with an AMD Ryzen 3950x CPU and two Nvidia GTX 1080TI GPUs, as well as g4dn.8xlarge instances provisioned on Amazon AWS. Either system can run two or three trials simultaneously, each consisting of three agents training together in a shared environment. Collecting experience and updating parameters were comparably time consuming, and a single 1.5M episode (375M step) 3-agent training run took about 30 hours. We used Ubuntu 18.04 with python3.8 and all neural networks are implemented in PyTorch v1.6 (Paszke et al. 2019). Training metrics were logged with Weights and Biases (Biewald 2020).