TRANSFER AND EXPLORATION VIA THE INFORMATION BOTTLENECK

Anonymous authors
Paper under double-blind review

ABSTRACT

A central challenge in reinforcement learning is discovering effective policies for tasks where rewards are sparsely distributed. We postulate that in the absence of useful reward signals, an effective exploration strategy should seek out *decision states*. These states lie at critical junctions in the state space from where the agent can transition to new, potentially unexplored regions. We propose to learn about decision states from prior experience. By training a goal-conditioned model with an information bottleneck, we can identify decision states by examining where the model accesses the goal state through the bottleneck. We find that this simple mechanism effectively identifies decision states, even in partially observed settings. In effect, the model learns the sensory cues that correlate with potential subgoals. In new environments, this model can then identify novel subgoals for further exploration, guiding the agent through a sequence of potential decision states and through new regions of the state space. We demonstrate that this approach to exploration outperforms several state-of-the-art exploration methods in several challenging domains.

1 INTRODUCTION

Deep reinforcement learning has enjoyed many recent successes in domains where large amounts of training time and a dense reward function are provided. However, learning to quickly perform well in environments with sparse rewards remains a major challenge. Providing agents with useful signals to pursue in lieu of environmental reward becomes crucial in these scenarios. Here, we propose to incentive agents to learn about and exploit multi-goal task structure in order to efficiently explore in new environments. We do so by first training agents to develop useful habits as well as the knowledge of when to break them, and then using that knowledge to efficiently probe new environments for reward.

We focus on multi-goal environments and goal-conditioned policies (Foster and Dayan, 2002; Schaul et al., 2015; Plappert et al., 2018). In these scenarios, a goal G is sampled from $p(G)$ and the beginning of each episode and provided to the agent. The goal G provides the agent with information about the environment's reward structure for that episode. For example, in spatial navigation tasks, G might be the location or direction of a rewarding state. We denote the agent's policy $\pi_\theta(A \mid S, G)$, where S is the agent's state, A the agent's action, and θ the policy parameters.

We incentive agents to learn task structure by training policies that perform well under a variety of goals, while not overfitting to any individual goal. We achieve this by training agents that, in addition to maximizing reward, minimize the policy dependence on the individual goal, quantified by the conditional mutual information $I(A; G \mid S)$. This approach is inspired by the information bottleneck approach (Tishby et al., 1999) to training deep neural networks for supervised learning (Alemi et al., 2017; Chalk et al., 2016; Achille and Soatto, 2016; Kolchinsky et al., 2017), where classifiers are trained to achieve high accuracy while simultaneously encoding as little information about the input as possible. This form of "information dropout" has been shown to promote generalization performance (Achille and Soatto, 2016; Alemi et al., 2017). Here, we show that minimizing goal information promotes generalization in an RL setting as well. We refer to this approach as InfoBot (from information bottleneck).

This approach to learning task structure can also be interpreted as encouraging agents to develop *useful habits*. To see this, note that our regularizer can also be written as $I(A; G \mid S) =$

$\mathbb{E}_{\pi_\theta}\left[D_{\text{KL}}[\pi_\theta(A \mid S, G) \mid \pi_0(A \mid S)]\right]$, where $\pi_\theta(A \mid S, G)$ is the agent's multi-goal policy, $\mathbb{E}_{\pi_\theta}$ denotes an expectation over trajectories generated by π_θ, D_{KL} is the Kuhlback-Leibler divergence, and $\pi_0(A \mid S) = \sum_g p(g)\, \pi_\theta(A \mid S, g)$ is a "habit" policy with the goal marginalized out. While the agent never actually follows the habit policy π_0 directly, it can be viewed as what the agent might do in the absence of any knowledge about the goal. Thus, our approach encourages the agent to learn useful habits and to follow those habits closely, except where diverting from doing so leads to significantly higher reward. Humans too demonstrate an affinity for relying on habits when they can (Kool and Botvinick, 2018), which we take as encouraging support for this line of work (Hassabis et al., 2017).

We refer to states where diversions from habits occur as *decision states*, based on the intuition that they require the agent not to rely on their habit but instead to make a goal-dependent "decision." Our approach to exploration then involves encouraging the agent to seek out these decision states in new environments. The intuition behind this approach is that such decision states are natural subgoals for efficient exploration because they are boundaries between achieving different goals (van Dijk and Polani, 2011). By encouraging an agent to visit them, the agent is encouraged to 1) follow habitual trajectories that work across many goals and 2) uniformly explore across the many "branches" of decision-making. We encourage the visitation of decision states by first training an agent to recognize them by training with the information regularizer introduced above. Then, we freeze the agent's policy, and use $D_{\text{KL}}[\pi_\theta(A \mid S, G) \mid \pi_0(A \mid S)]$ as an exploration bonus for training a *new* policy. Crucially, this approach to exploration is tuned to the family of tasks the agent is trained on, and we show that it promotes more efficient exploration than other task-agnostic approaches to exploration (Houthooft et al., 2016; Pathak et al., 2017).

To summarize our contributions:

- We regularize RL agents in multi-goal settings with $I(A; G \mid S)$, an approach inspired by the information bottleneck and the cognitive science of habits, and show that it promotes *generalization* across tasks.
- We use policies as trained above to then provide an *exploration bonus* for training new policies in the form of $D_{\text{KL}}[\pi_\theta(A \mid S, G) \mid \pi_0(A \mid S)]$, which encourages the agent to seek out decision states. We demonstrate that this approach to exploration performs more effectively than other state-of-the-art methods, including a count-based bonus, VIME (Houthooft et al., 2016), and curiosity (Pathak et al., 2017).

2 Our Approach

Our objective is to train an agent on one set of tasks (environments) $T \sim p_{\text{train}}(T)$, but to have the agent perform well on another different, but related, set of tasks $T \sim p_{\text{test}}(T)$. To do so, we propose to maximize the following objective in the training environments:

$$\begin{aligned} J(\theta) &\equiv \mathbb{E}_{\pi_\theta}[r] - \beta I(A; G \mid S) \\ &= \mathbb{E}_{\pi_\theta}\left[r - \beta D_{\text{KL}}\left[\pi_\theta(A \mid S, G) \mid \pi_0(A \mid S)\right]\right], \end{aligned} \tag{1}$$

where $\mathbb{E}_{\pi_\theta}$ denotes an expectation over trajectories generated by the agent's policy, $\beta > 0$ is a tradeoff parameter, D_{KL} is the Kuhlback-Leibler divergence, and $\pi_0(A \mid S) \equiv \sum_g p(g)\, \pi_\theta(A \mid S, g)$ is a "habit" policy with the agent's goal marginalized out.

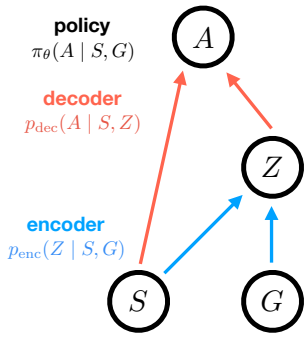


Figure 1: Policy architecture.

2.1 Tractable Bounds on Information

We parameterize the policy $\pi_\theta(A \mid S, G)$ using an encoder $p_{\text{enc}}(Z \mid S, G)$ and a decoder $p_{\text{dec}}(A \mid S, Z)$ such that $\pi_\theta(A \mid S, G) = \sum_z p_{\text{enc}}(z \mid S, G)\, p_{\text{dec}}(A \mid S, z)$ (see figure 1).[1] The encoder output Z is meant

[1] In practice, we estimate the marginalization over Z using a single sample throughout our experiments.

to represent the information about the present goal G that the agent believes is important to access in the present state S in order to perform well. The decoder takes this encoded goal information and the current state and produces a distribution over actions A.

We suppress the dependence of p_{enc} and p_{dec} on θ, but θ in the union of their parameters. Due to the data processing inequality (DPI) (Cover and Thomas, 2006), $I(Z; G \mid S) \geq I(A; G \mid S)$, so minimizing the goal information encoded by p_{enc} also minimizes $I(A; G \mid S)$.

Thus, we instead maximize this lower bound on $J(\theta)$:

$$\begin{aligned} J(\theta) &\geq \mathbb{E}_{\pi_\theta}[r] - \beta I(Z; G \mid S) \\ &= \mathbb{E}_{\pi_\theta}[r - \beta D_{\text{KL}}\left[p_{\text{enc}}(Z \mid S, G) \mid p(Z \mid S)\right]], \end{aligned} \tag{2}$$

where $p(Z \mid S) = \sum_g p(g)\, p_{\text{enc}}(Z \mid S, g)$ is the marginalized encoding.

In practice, performing this marginalization over the goal may often be prohibitive, since the agent might not have access to the goal distribution $p(G)$, or even if the agent does, there might be many or a continuous distribution of goals that makes the marginalization intractable. To avoid this marginalization, we replace $p(Z \mid S)$ with a variational approximation $q(Z \mid S)$ (Kingma and Welling, 2014; Alemi et al., 2017; Houthooft et al., 2016; Strouse et al., 2018). This again provides a lower bound on $J(\theta)$ since:

$$\begin{aligned} I(Z; G \mid S) &= \sum_{z,s,g} p(z, s, g) \log \frac{p(z \mid g, s)}{p(z \mid s)} \\ &= \sum_{z,s,g} p(z, s, g) \log p(z \mid g, s) - \sum_{z,s} p(s)\, p(z \mid s) \log p(z \mid s) \\ &\geq \sum_{z,s,g} p(z, s, g) \log p(z \mid g, s) - \sum_{z,s} p(s)\, p(z \mid s) \log q(z \mid s), \end{aligned} \tag{3}$$

where the inequality in the last line, in which we replace $p(z \mid s)$ with $q(z \mid s)$, follows from that $D_{\text{KL}}\left[p(Z \mid s) \mid q(Z \mid s)\right] \geq 0 \Rightarrow \sum_z p(z \mid s) \log p(z \mid s) \geq \sum_z p(z \mid s) \log q(z \mid s)$.

Thus, we arrive at the lower bound $\tilde{J}(\theta)$ that we maximize in practice:

$$J(\theta) \geq \tilde{J}(\theta) \equiv \mathbb{E}_{\pi_\theta}[r - \beta D_{\text{KL}}\left[p_{\text{enc}}(Z \mid S, G) \mid q(Z \mid S)\right]]. \tag{4}$$

In the experiments below, we fix $q(Z \mid S)$ to be unit Gaussian, however it could also be learned, in which case its parameters should be included in θ. Although our approach is compatible with any RL method, we maximize $\tilde{J}(\theta)$ on-policy from sampled trajectories using a score function estimator (Williams, 1992; Sutton et al., 1999a). As derived by Strouse et al. (2018), the resulting update at time step t, which we denote $\nabla_\theta \tilde{J}(t)$, is:

2.2 Optimization from sampled trajectories

$$\nabla_\theta \tilde{J}(t) = \tilde{R}_t \log(\pi_\theta(a_t \mid s_t, g_t)) - \beta \nabla_\theta D_{\text{KL}}\left[p_{\text{enc}}(Z \mid s_t, g_t) \mid q(Z \mid s_t)\right], \tag{5}$$

where $\tilde{R}_t \equiv \sum_{u=t}^{T} \gamma^{u-t} \tilde{r}_u$ is a modified return, $\tilde{r}_t \equiv r_t + \beta D_{\text{KL}}\left[p_{\text{enc}}(Z \mid s_t, g) \mid q(Z \mid s_t)\right]$ is a modified reward, T is the length of the current episode, and a_t, s_t, and g_t are the action, state, and goal at time t, respectively. The first term in the gradient comes from applying the REINFORCE update to the modified reward, and can be thought of as encouraging the agent to change the policy in the present state to revisit future states to the extent that they provide high external reward as well as low need for encoding goal information. The second term comes from directly optimizing the policy to not rely on goal information, and can be thought of as encouraging the agent to directly alter the policy to avoid encoding goal information in the present state. We stress that while we take a Monte Carlo policy gradient, or REINFORCE, approach here, our regularizer and thus our method are compatible with any RL algorithm.

2.3 Policy and exploration transfer

The end result of training with Eqn 5 is that the agent learns to rely on its (goal-independent) habits as much as possible, deviating only in decision states (as introduced in Section 1) where it makes

goal-dependent modifications. We demonstrate in Section 4 that this regularization alone already leads to generalization benefits (that is, increased performance on $T \sim p_{\text{test}}(T)$ after training on $T \sim p_{\text{train}}(T)$).

However, our main purpose in training with Eqn 5 is to train the agent to identify decision states for the purpose of providing an exploration bonus in new environments. That is, after training on $T \sim p_{\text{train}}(T)$, we freeze the agent's encoder $p_{\text{enc}}(Z \mid S, G)$ and marginal encoding $q(Z \mid S)$, discard the decoder $p_{\text{dec}}(A \mid S, Z)$, and use the encoders to provide $D_{\text{KL}}\left[p_{\text{enc}}(Z \mid S, G) \mid q(Z \mid S)\right]$ as a state and goal dependent exploration bonus for training a new policy $\pi_\phi(A \mid S, G)$ on $T \sim p_{\text{test}}(T)$. To ensure that the new agent does not pursue the exploration bonus solely (in lieu of reward), we also decay the bonus with continued visits by weighting with a count-based exploration bonus as well. That is, we divide the KL divergence by $\sqrt{c(S)}$, where $c(S)$ is the number of times that state has been visited during training, which is initialized to 1. Letting $r_e(t)$ be the environmental reward at time t, we thus train the agent to maximize the combined reward r_t:

$$r_t = r_e(t) + \frac{\beta}{\sqrt{c(s_t)}} D_{\text{KL}}\left[p_{\text{enc}}(Z \mid s_t, g_t) \mid q(Z \mid s_t)\right]. \tag{6}$$

Our approach is summarized in algorithm 1.

Algorithm 1 Transfer and Exploration via the Information Bottleneck

Require: A policy $\pi_\theta(A \mid S, G) = \sum_z p_{\text{enc}}(z \mid S, G)\, p_{\text{dec}}(A \mid S, z)$, parameterized by θ
Require: A variational approximation $q(Z \mid S)$ to the goal-marginalized encoder
Require: A regularization weight β
Require: Another policy $\pi_\phi(A \mid S, G)$, along with a RL algorithm $\mathcal{A}$ to train it
Require: A set of training tasks (environments) $p_{\text{train}}(T)$ and test tasks $p_{\text{test}}(T)$
Require: A goal sampling strategy $p(G \mid T)$ given a task T
  for episodes = 1 to N_{train} **do**
    Sample a task $T \sim p_{\text{train}}(T)$ and goal $G \sim p(G \mid T)$
    Produce trajectory τ on task T with goal G using policy $\pi_\theta(A \mid S, G)$
    Update policy parameters θ over τ using Eqn 5
  end for
  Optional: use π_θ directly on tasks sampled from $p_{\text{test}}(T)$
  for episodes = 1 to N_{test} **do**
    Sample a task $T \sim p_{\text{test}}(T)$ and goal $G \sim p(G \mid T)$
    Produce trajectory τ on task T with goal G using policy $\pi_\phi(A \mid S, G)$
    Update policy parameters ϕ using algorithm $\mathcal{A}$ to maximize the reward given by Eqn 6
  end for

3 Related work

van Dijk and Polani (2011) were the first to point out the connection between action-goal information and the structure of decision-making. They used information to identify decision states and use them as subgoals in an options framework (Sutton et al., 1999b). We build upon their approach by combining it with deep reinforcement learning to make it more scaleable, and also modify it by using it to provide an agent with an exploration bonus, rather than subgoals for options.

Our decision states are similar in spirit to the notion of "bottleneck states" used to define subgoals in hierarchical reinforcement learning. A bottleneck state is defined as one that lies on a wide variety of rewarding trajectories (McGovern and Barto, 2001; Stolle and Precup, 2002) or one that otherwise serves to separate regions of state space in a graph-theoretic sense (Menache et al., 2002; Şimşek et al., 2005; Kazemitabar and Beigy, 2009; Machado et al., 2017). The latter definition is purely based on environmental dynamics and does not incorporate reward structure, while both definitions can lead to an unnecessary profileration of subgoals. To see this, consider a T-maze in which the agent starts at the bottom and two possible goals exist at either end of the top of the T. All states in this setup are bottleneck states, and hence the notion is trivial. However, only the junction where the lower and upper line segments of the T meet are a decision state. Thus, we believe the notion of a decision state is a more parsimonious and accurate indication of good subgoals than is the above notions of a

bottleneck state. The success of our approach against state-of-the-art exploration methods (Section 4) supports this claim.

We use the terminology of information bottleneck (IB) in this paper because we limit (or bottleneck) the amount of goal information used by our agent's policy during training. However, the correspondence is not exact: while both our method and IB limit information into the model, we maximize *rewards* while IB maximizes *information* about a target to be predicted. The latter is thus a supervised learning algorithm. If we instead focused on imitation learning and replaced $\mathbb{E}[r]$ with $I(A^*; A \mid S)$ in Eqn 1, then our problem would correspond exactly to a variational information bottleneck (Alemi et al., 2017) between the goal G and correct action choice A^* (conditioned on S).

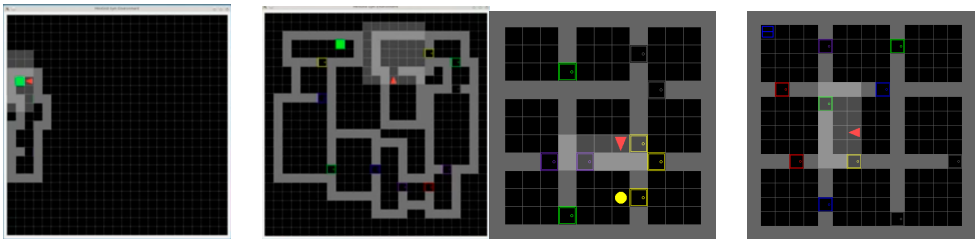
Whye Teh et al. (2017) trained a policy with the same KL divergence term as in Eqn 1 for the purposes of encouraging transfer across tasks. They did not, however, note the connection to variational information minimization and the information bottleneck, nor did they leverage the learned task structure for exploration. Parallel to our work, Strouse et al. (2018) also used Eqn 1 as a training objective, however their purpose was not to show better generalization and transfer, but instead to promote the sharing and hiding of information in a multi-agent setting.

Popular approaches to exploration in RL are typically based on: 1) injecting noise into action selection (e.g. epsilon-greedy, (Osband et al., 2016)), 2) encouraging "curiosity" by encouraging prediction errors of or decreased uncertainty about environmental dynamics (Schmidhuber, 1991; Houthooft et al., 2016; Pathak et al., 2017), or 3) count-based methods which incentivize seeking out rarely visited states (Strehl and Littman, 2008; Bellemare et al., 2016; Tang et al., 2016; Ostrovski et al., 2017). One limitation shared by all of these methods is that they have no way to leverage experience on previous tasks to improve exploration on new ones; that is, their methods of exploration are not tuned to the family of tasks the agent faces. Our transferrable exploration strategies approach in algorithm 1 however does exactly this. Another notable recent exception is Gupta et al. (2018), which took a meta-learning approach to transferrable exploration strategies.

4 Experimental Results

In this section, we demonstrate the following experimentally:

- The policy-goal information bottleneck leads to much better policy transfer than standard RL training procedures (direct policy transfer).
- Using decision states as an exploration bonus leads to better performance than a variety of standard task-agnostic exploration methods (transferable exploration strategies).



(a) MultiRoomN4S4 (b) MultiRoomN12S10 (c) FindObjS5 (d) FindObjS6

Figure 2: **MultiRoomNXSY and FindObjSY MiniGrid environments.** See text for details.

4.1 MiniGrid Environments

The first set of environments we consider are partially observable grid worlds generated with MiniGrid (Chevalier-Boisvert and Willems, 2018), an OpenAI Gym package (Brockman et al., 2016). We consider the **MultiRoomN**X**S**Y and a **FindObjS**Y task domains, as depicted in Figure 2. Both environments consist of a series of connected rooms, sometimes separated by doors that need opened. In both tasks, black squares are traversible, grey squares are walls, black squares with colored

borders are doors, the red triangle is the agent, and the shaded area is the agent's visible region. The MultiRoomNXSY the environment consists of X rooms, with size at most Y, connected in random orientations. The agent is placed in a distal room and must navigate to a green goal square in the most distant room from the agent. The agent receives an egocentric view of its surrounding, consisting of 3×3 pixels. The task increases in difficulty with X and Y. The FindObjSY environment consists of 9 connected rooms of size $Y - 2 \times Y - 2$ arranged in a grid. The agent is placed in the center room and must navigate to an object in a randomly chosen outer room (e.g. yellow circle in bottom room in Figure 2c and blue square in top left room in Figure 2d). The agent again receives an egocentric observation, this time consisting of 7×7 pixels, and again the difficulty of the task increases with Y. For more details of the environment, see Appendix C.

Solving these partially observable, sparsely rewarded tasks by random exploration is difficult because there is a vanishing probability of reaching the goal randomly as the environments become larger. Transferring knowledge from simpler to more complex versions of these tasks thus becomes essential. In the next two sections, we demonstrate that our approach yields 1) policies that directly transfer well from smaller to larger environments, and 2) exploration strategies that outperform other task-agnostic exploration approaches.

4.2 Direct Policy Generalization on MiniGrid Tasks

We first demonstrate that training an agent with a goal bottleneck alone already leads to more effective policy transfer. We train policies on smaller versions of the MiniGrid environments (MultiRoomN2S6 and FindObjS5 and S7), but evaluate them on larger versions (MultiRoomN10S4, N10S10, and N12S10, and FindObjS7 and S10) throughout training.

Figure 3 compares an agent trained with a goal bottleneck (first half of Algorithm 1) to a vanilla goal-conditioned A2C agent (Mnih et al., 2016) on MultiRoomNXSY generalization. As is clear, the goal-bottlenecked agent generalizes much better. The success rate is the number of times the agent solves a larger task with 10-12 rooms while it is being trained on a task with only 2 rooms. When generalizing to 10 small rooms, the agent learns to solve the task to near perfection, whereas the goal-conditioned A2C baseline only solves <50% of mazes (Figure 3a).

Table 1 compares the same two agents on FindObjSY generalization. In addition, this comparison includes an ablated version of our agent with $\beta = 0$, that is an agent with the same architecture as in Figure 1 but with the no information term in its training objective. This is to ensure that our method's success is not due to the architecture alone. As is evident, the goal-bottlenecked agent again generalizes much better.

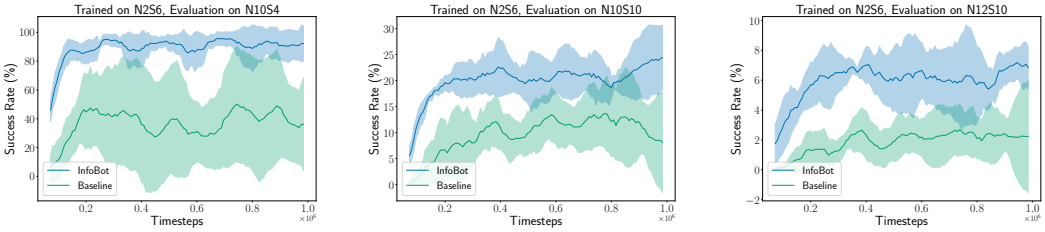
We analyzed the agent's behaviour to understand the intuition of why it generalizes well. In the MultiRoomNXSY environments, we find that the agent quickly discovers a wall following strategy. Since these environments are partially observable, this is indeed a good strategy that also generalizes well to larger mazes. In the FindObjSY environments, on the other hand, the agent sticks toward the center of the rooms, making a beeline from doorway to doorway. This is again a good strategy, because the agent's field of view in these experiments is large enough to see the entire room in which its in to determine if the goal object is present or not.

Method	**FindObjS7**	**FindObjS10**
Goal-conditioned A2C	56%	36%
InfoBot with $\beta = 0$	44%	24%
InfoBot	**81%**	**61%**

Table 1: **Policy generalization on FindObjSY.** Agents trained on FindObjS5, and evaluated on FindObjS7 and S10.

4.3 Transferable Exploration Strategies on MiniGrid Tasks

We now evaluate our approach to exploration (the second half of Algorithm 1). We train agents with a goal bottleneck on one set of environments (MultiRoomN2S6) where they learn the sensory cues that correspond to decision states. Then, we that knowledge to guide exploration one another set of environments (MultiRoomN3S4, MultiRoomN4S4, and MultiRoomN5S4).We compare to several



(a) (b) (c)

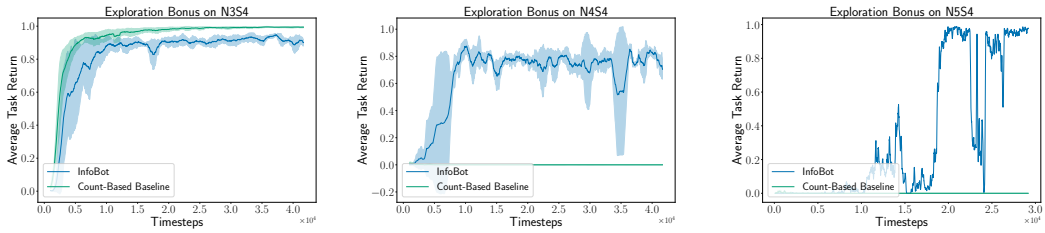
Figure 3: **Policy generalization on MultiRoomNXSY.** Success is measured by the percent of time the agent can find the goal in an unseen maze. Error bars are standard deviations across runs. Baseline is a vanilla goal-conditioned A2C agent.

Method	MultiRoomN3S4	MultiRoomN5S4
Goal-conditioned A2C	0%	0%
TRPO + VIME	54%	0%
Count based exploration	**95%**	0%
Curiosity-based exploration	**95%**	54%
InfoBot (decision state exploration bonus)	90%	**85%**

Table 2: **Transferable exploration strategies on MultiRoomNXSY.** InfoBot encoder trained on MultiRoomN2S6. All agents evaluated on MultiRoomN3S4 and N5S4. While several methods perform well with 3 rooms, InfoBot performs far better as the number of rooms increases to 5.

standard task-agnostic exploration methods, including count-based exploration (Eqn 6 without the D_{KL}, that a bonus of $\beta/\sqrt{c(s)}$), VIME (Houthooft et al., 2016), and curiosity-driven exploration (Pathak et al., 2017), as well as a goal-conditioned A2C baseline with no exploration bonuses. Results are shown in Table 2 and Figure 4.

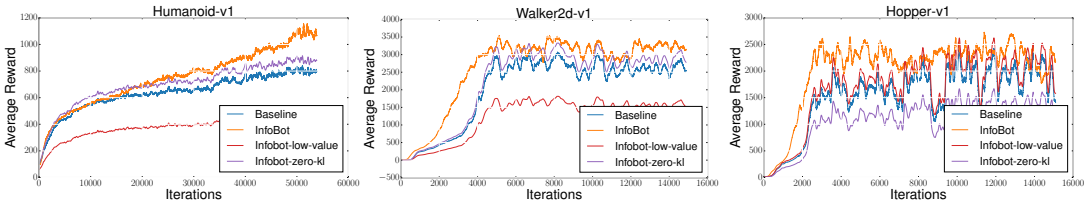
On a maze with three rooms, the count-based method and curiosity-driven exploration slightly outperform the proposed learned exploration strategy. However, as the number of rooms increases, the count-based method and VIME fail completely and the curiosity-based method degrades to only 54% success rate. This is in contrast to the proposed exploration strategy, which by learning the structure of the task, maintains a high success rate of 85%.



(a) (b) (c)

Figure 4: **Transferable exploration strategies on MultiRoomNXSY.** As the number of rooms increases (from left to right), a count-based exploration bonus alone cannot solve the task, whereas the proposed exploration bonus, by being tuned to task structure, enables success on these more difficult tasks.

4.4 TRANSFERABLE EXPLORATION STRATEGIES FOR CONTINUOUS CONTROL



(a) (b) (c)

Figure 5: **Transferable exploration strategies on Humanoid, Walker2D, and Hopper.** The "baseline" is PPO (Schulman et al., 2017). Experiments are run with 5 random seeds and averaged over the runs.

To show that the InfoBot architecture can also be applied to continuous control, we evaluated the performance of InfoBot on three continuous control tasks from OpenAI Gym (Brockman et al., 2016). Because InfoBot depends on the goal, in the control domains, we use high value states as an approximation to the goal state following Goyal et al. (2018). We maintain a buffer of high value states, and at each update, we sample a high value state from the buffer which acts as a proxy for the goal. We compared to proximal policy optimization (PPO) (Schulman et al., 2017), as well as two ablated versions of our model: 1) instead of taking high value states, we take low value states from the buffer as proxy to the goal ("InfoBot-low-value") and 2) we use the same InfoBot policy architecture but do not use the information regularizer (i.e. $\beta = 0$) ("InfoBot-zero-kl"). The results in Figure 5 show that InfoBot improves over all three alternatives.

4.5 TRANSFERABLE EXPLORATION STRATEGIES FOR ATARI

To show that the InfoBot formulation generalizes to more complex domains, we evaluated the performance of our model on Atari games (Bellemare et al., 2013) and compare to a standard A2C baseline (Mnih et al., 2016). In this domain, we again use high value states as an approximation to the goal state following Goyal et al. (2018). Figure 6 shows the performance of our InfoBot compared to the A2C baseline on four of the Atari games. Although the gains are more modest than in the other domains we tested, InfoBot still performs significantly better.

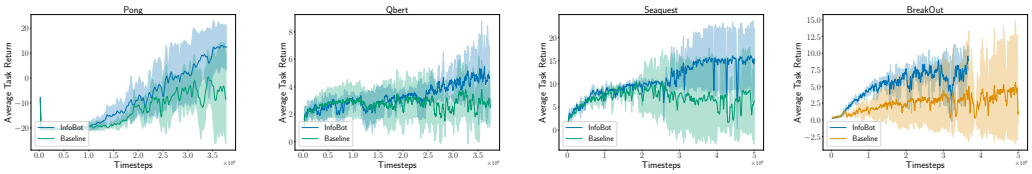


Figure 6: **Transferable exploration strategies on Pong, Qbert, Seaquest, and Breakout.** The baseline is a vanilla A2C agent. Results averaged over three random seeds.

5 CONCLUSION

In this paper, we proposed to train agents to develop "useful habits" as well as the knowledge of when to break those habits, using an information bottleneck between the agent's goal and policy. We demonstrated empirically that this training procedure leads to better direct policy transfer across tasks. We also demonstrated that the states in which the agent learns to deviate from its habits, which we call "decision states", can be used as the basis for a learned exploration bonus that leads to more effective training than other task-agnostic exploration methods.

REFERENCES

A. Achille and S. Soatto. Information Dropout: Learning Optimal Representations Through Noisy Computation. *ArXiv e-prints*, Nov. 2016.

A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. *International Conference on Learning Representations (ICLR)*, abs/1612.00410, 2017. URL http://arxiv.org/abs/1612.00410.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying Count-Based Exploration and Intrinsic Motivation. *ArXiv e-prints*, June 2016.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *ArXiv e-prints*, June 2016.

M. Chalk, O. Marre, and G. Tkacik. Relevant sparse codes with variational information bottleneck. *ArXiv e-prints*, May 2016.

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

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

D. Foster and P. Dayan. Structure in the space of value functions. *Machine Learning*, 49(2):325–346, Nov 2002. ISSN 1573-0565. doi: 10.1023/A:1017944732463. URL https://doi.org/10.1023/A:1017944732463.

A. Goyal, P. Brakel, W. Fedus, T. Lillicrap, S. Levine, H. Larochelle, and Y. Bengio. Recall traces: Backtracking models for efficient reinforcement learning. *arXiv preprint arXiv:1804.00379*, 2018.

A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. Meta-Reinforcement Learning of Structured Exploration Strategies. *ArXiv e-prints*, Feb. 2018.

D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–248, 2017. URL https://doi.org/10.1016/j.neuron.2017.06.011.

R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

S. J. Kazemitabar and H. Beigy. Using strongly connected components as a basis for autonomous skill acquisition in reinforcement learning. In *International Symposium on Neural Networks*, pages 794–803. Springer, 2009.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations (ICLR)*, Dec. 2014.

A. Kolchinsky, B. D. Tracey, and D. H. Wolpert. Nonlinear Information Bottleneck. *ArXiv e-prints*, May 2017.

W. Kool and M. Botvinick. Mental labour. *Nature Human Behaviour*, 2018.

I. Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr, 2018.

M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. *arXiv preprint arXiv:1703.00956*, 2017.

A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. 2001.

I. Menache, S. Mannor, and N. Shimkin. Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European Conference on Machine Learning*, pages 295–306. Springer, 2002.

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

V. Mnih, A. Puigdomènech Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *ArXiv e-prints*, 2016.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4026–4034. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6501-deep-exploration-via-bootstrapped-dqn.pdf.

G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos. Count-Based Exploration with Neural Density Models. *ArXiv e-prints*, Mar. 2017.

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/pathak17a.html.

M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *ArXiv e-prints*, Feb. 2018.

T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/schaul15.html.

J. Schmidhuber. Curious model-building control systems. In *In Proc. International Joint Conference on Neural Networks, Singapore*, pages 1458–1463. IEEE, 1991.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *ArXiv e-prints*, July 2017.

Ö. Şimşek, A. P. Wolfe, and A. G. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd international conference on Machine learning*, pages 816–823. ACM, 2005.

M. Stolle and D. Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.

A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for markov decision processes. *J. Comput. Syst. Sci.*, 74(8):1309–1331, Dec. 2008. ISSN 0022-0000. doi: 10.1016/j.jcss.2007.08.009. URL http://dx.doi.org/10.1016/j.jcss.2007.08.009.

D. Strouse, M. Kleiman-Weiner, J. Tenenbaum, M. Botvinick, and D. Schwab. Learning to share and hide intentions using information regularization. In *Advances in Neural Information Processing Systems (NIPS) 31*. 2018.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999a. MIT Press. URL http://dl.acm.org/citation.cfm?id=3009657.3009806.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, Aug. 1999b. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1. URL http://dx.doi.org/10.1016/S0004-3702(99)00052-1.

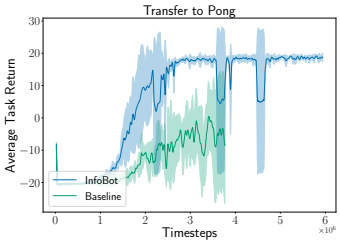
H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. *ArXiv e-prints*, Nov. 2016.

N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. *Proceedings of The 37th Allerton Conference on Communication, Control, and Computing*, pages 368–377, 1999.

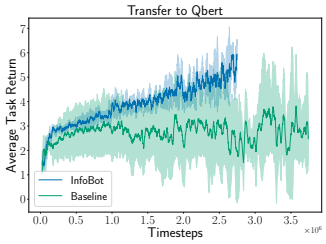
S. G. van Dijk and D. Polani. Grounding subgoals in information transitions. *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2011. URL https://ieeexplore.ieee.org/document/5967384/.

Y. Whye Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust Multitask Reinforcement Learning. *ArXiv e-prints*, 2017.

R. J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.



(a) Transfer task to Pong



(b) Transfer task to Qbert

Figure 7: **Transfer across ALE Games** (Pong, Qbert and Freeway) using egocentric encoder to provide exploration bonus, trained from Seaquest. Comparison of InfoBot (A2C + KL Regularizer) with a Baseline A2C. Experiment results averaged over four random seeds. See Section A.

A TRANSFERRED EXPLORATION STRATEGY IN ATARI

We demonstrate that the encoder can be used to transfera exploration strategy across Atari games to help agents learn a new game quickly. To initialize the encoder, we train an agent on the game Seaquest, where the agent is trained to identify the decision states. We then re-use this encoder on another Atari environment to provide an exploration bonus. Our experiments on the Atari games are evaluated on the games of Pong and Qbert. On Pong, an agent with learns to get a task return of 20 in 3M time steps. We show that the identification of bottleneck states can be used to transfer exploration strategies across Atari games (Fig. 7).

B ALGORITHM IMPLEMENTATION DETAILS

We evaluate the InfoBot framework using Adavantage Actor-Critic (A2C) to learn a policy $\pi_\theta(a|s, g)$ conditioned on the goal. To evaluate the performance of InfoBot, we use a range of maze multi-room tasks from the gym-minigrid framework (Chevalier-Boisvert and Willems, 2018) and the A2C implementation from (Chevalier-Boisvert and Willems, 2018). For the maze tasks, we used agent's relative distance to the absolute goal position as "goal".

For the maze environments, we use A2C with 48 parallel workers. Our actor network and critic networks consist of two and three fully connected layers respectively, each of which have 128 hidden units. The encoder network is also parameterized as a neural network, which consists of 1 fully connected layer. We use RMSProp with an initial learning rate of 0.0007 to train the models, for both InfoBot and the baseline for a fair comparison. Due to the partially observable nature of the environment, we further use a LSTM to encode the state and summarize the past observations.

For the Atari experiments, we use the open-source A2C implementation from Kostrikov (2018) and condition the goal state into the policy network. For the actor network, we use 3 convolution layers with ReLU activations. For training the networks, we use RMSProp with pre-defined learning rates for our algorithm and the baseline. The goal state in our experiments is used as the high value state following (Goyal et al., 2018).

For the Mujoco Experiments, we use Proximal Policy Optimization (PPO) (Schulman et al., 2017) with the open-source implementation available in Kostrikov (2018), using the defined architectures and hyperparameters as given in the repository for both our algorithm and the baseline. Again, for the goal-conditioned policies, the goal state for Mujoco experiments is defined as the high value state following (Goyal et al., 2018).

For reproducibility purposes of our experiments, we will further release the code on github that will be available on [2]

Hyperparameter Search for Grid World Experiments: For the proposed method, we only varied the weight of KL loss. We tried 3 different values 0.1, 0.9, 0.01, 0.09, 0.005 for each of the env. and

[2] https://github.com/anonymous

plotted the results which gave the most improvement. We used CNN policy for the case of FindObjSY env, and MLP policy for the case of MultiRoomNXSY. We used the agent's relative distance to goal as a goal for our goal conditioned policy.

Hyperparameter Search for Control experiments We have not done any hyper-parameter search for the baseline. For the proposed method, we only varied the weight of KL loss. We tried 3 different values 0.5, 0.01, 0.09 for each of the env. and plotted the results which gave the most improvement over the PPO baseline.

C MINIGRID ENVIRONMENTS FOR OPENAI GYM

The FindObj and MultiRoom environments used for this research are part of MiniGrid, which is an open source gridworld package[3]. This package includes a family reinforcement learning environments compatible with the OpenAI Gym framework. Many of these environments are parameterizable so that the difficulty of tasks can be adjusted (eg: the size of rooms is often adjustable).

C.1 THE WORLD

In MiniGrid, the world is a grid of size NxN. Each tile in the grid contains exactly zero or one object. The possible object types are wall, door, key, ball, box and goal. Each object has an associated discrete color, which can be one of red, green, blue, purple, yellow and grey. By default, walls are always grey and goal squares are always green.

C.2 REWARD FUNCTION

Rewards are sparse for all MiniGrid environments. In the MultiRoom environment, episodes are terminated with a positive reward when the agent reaches the green goal square. Otherwise, episodes are terminated with zero reward when a time step limit is reached. In the FindObj environment, the agent receives a positive reward if it reaches the object to be found, otherwise zero reward if the time step limit is reached.

The formula for calculating positive sparse rewards is $1 - 0.9 * (step_count/max_steps)$. That is, rewards are always between zero and one, and the quicker the agent can successfully complete an episode, the closer to 1 the reward will be. The max_steps parameter is different for each environment, and varies depending on the size of each environment, with larger environments having a higher time step limit.

C.3 ACTION SPACE

There are seven actions in MiniGrid: turn left, turn right, move forward, pick up an object, drop an object, toggle and done. For the purpose of this paper, the pick up, drop and done actions are irrelevant. The agent can use the turn left and turn right action to rotate and face one of 4 possible directions (north, south, east, west). The move forward action makes the agent move from its current tile onto the tile in the direction it is currently facing, provided there is nothing on that tile, or that the tile contains an open door. The agent can open doors if they are right in front of it by using the toggle action.

C.4 OBSERVATION SPACE

Observations in MiniGrid are partial and egocentric. By default, the agent sees a square of 7x7 tiles in the direction it is facing. These include the tile the agent is standing on. The agent cannot see through walls or closed doors. The observations are provided as a tensor of shape 7x7x3. However, note that these are not RGB images. Each tile is encoded using 3 integer values: one describing the type of object contained in the cell, one describing its color, and a flag indicating whether doors are open or closed. This compact encoding was chosen for space efficiency and to enable faster training. The fully observable RGB image view of the environments shown in this paper is provided for human viewing.

[3]https://github.com/maximecb/gym-minigrid

C.5 LEVEL GENERATION

The level generation in this task works as follows: (1) Generate the layout of the map (X number of rooms with different sizes (at most size Y) and green goal) (2) Add the agent to the map at a random location in the first room. (3) Add the goal at a random location in the last room. **MultiRoomN**X**S**Y - In this task, the agent gets an egocentric view of its surroundings, consisting of 3×3 pixels. A neural network parameterized as MLP is used to process the visual observation.

FindObjSY - In this task, the agent's egocentric observation consists of 7×7 pixels. We use Convolutional Neural Networks to encode the visual observations.