Contrastive introspection to identify critical steps in reinforcement learning

0.1 Explanation A: Inductive biases in ConSpec for learning when success is contingent upon multiple critical steps

We aim to provide an intuition as to why the inductive biases present in ConSpec make it well suited to the problem of learning when success is contingent upon multiple critical steps. A key motivation for ConSpec is that learning a full model of the world and its transitions and values is very difficult, especially for a novice agent. Concretely, most other classic and contemporary approaches can be characterized at a high level as being about learning to predict rewards given sequences of observations in memory, i.e. learning a model of the probability of success given a trajectory: $p(\text{success}|\{O_i\})$. Modelling this function can potentially be highly nontrivial, especially if success is not dependent on a single critical step, but rather, depends jointly on a collection of critical steps $\{O_{t_1}..O_{t_k}\}$. To illustrate this, successful acceptance of a paper is only achieved if all the critical steps have jointly been achieved (an idea conceived, experiments run, paper written and reviewer concerns addressed) and not when only a subset has.

In contrast, ConSpec solves the reverse problem of predicting the critical steps given success. Put another way, it hones in on the set of critical steps, i.e. the steps that have high probability given success $p(\{O_{t_1}..O_{t_k}\}|\text{success}=1)$. This is a much easier problem, because conditioned on success, the critical steps are independent of one another $p(\{O_{t_1}..O_{t_k}\}|\text{success}=1) = \prod p(O_{t_i}|\text{success}=1)$. To see a concrete example, conditioned on the successful acceptance of a paper, one can be sure that all the critical steps had each individually been achieved (an idea conceived, experiments run, paper written and reviewer concerns addressed).

Given that all of the prototypes are learned in parallel, this means that the search for these critical steps do not depend on one another. More broadly, this implies that even with k critical steps the search is $\mathcal{O}(constant)$ when using ConSpec. On the other hand, any algorithm that was attempting to learn the function of how combinations of steps predict reward must consider all the potential combinations of critical steps, which means that they face a search of order up to $\mathcal{O}(2^k)$.

Crucially, critical steps arises ubiquitously in real episodic experiences. Hence their conditional independence property is tailored for exploitation in RL, which ConSpec does.

.

0.2 New Experiments

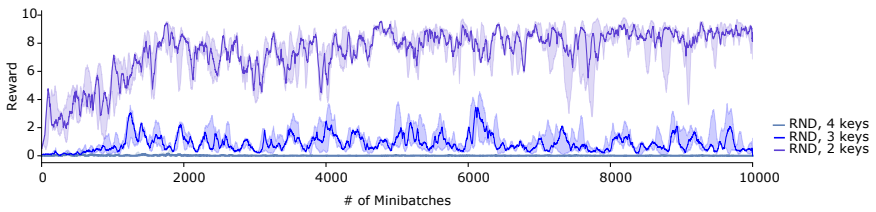

Figure 1: Performance of random network distillation (RND) on the multi-key-to-door task with 2, 3, or 4 keys.

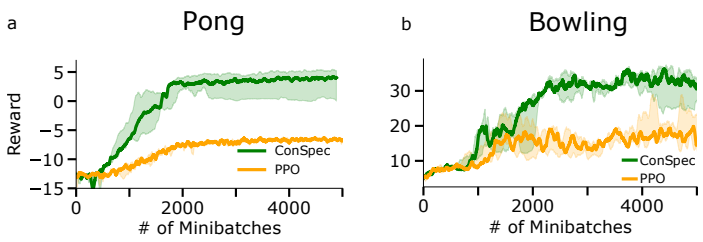

Figure 2: ConSpec improves performance on (a) delayed Pong and (b) delayed Bowling. Episodes were terminated after 600 timesteps and cumulative rewards were given only at the end.

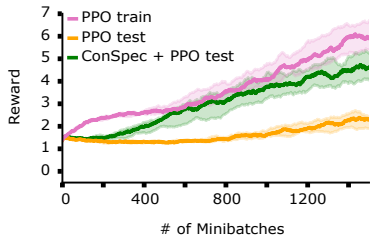

Figure 3: PPO trained on Procgen Dodgeball has a large generalization gap. In this game, ConSpec restored test reward to levels seen during PPO training, and significantly above reward levels during PPO test.

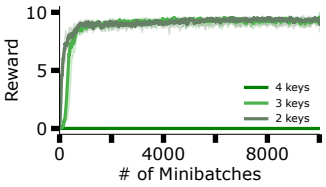

Figure 4: Performance of RUDDER on the multi-key-to-door task with 2, 3, or 4 keys.

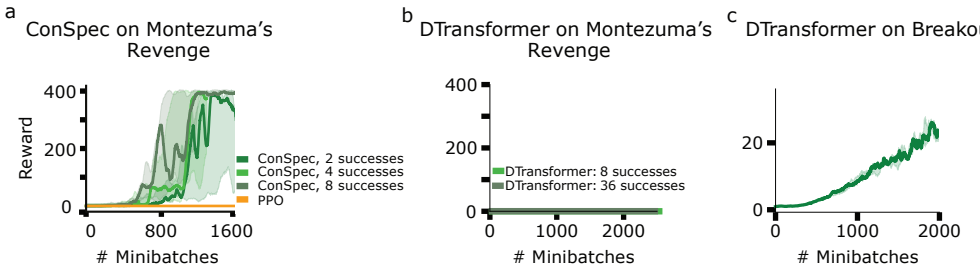

Figure 5: (a) ConSpec learns on Montezuma's Revenge with 2, 4, and 8 success demonstrations, highlighting how ConSpec uses few successes very efficiently. By contrast, Decision transformers (b) do not learn Montezuma's Revenge despite 8 or even 36 success demonstrations, but (c) do learn Breakout (as a positive control).

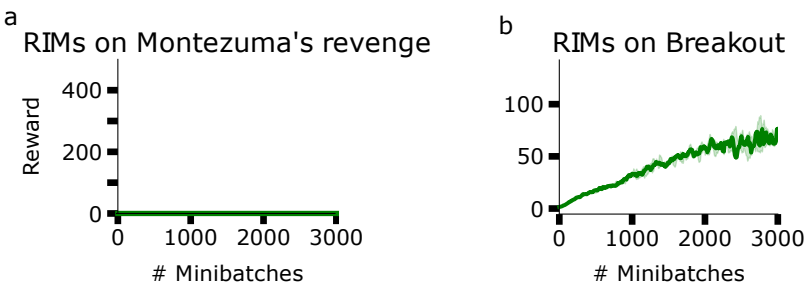

Figure 6: Performance of recurrent independent mechanisms (RIMs) on (a) Montezuma's Revenge, and (b) Breakout.