Contrastive introspection to identify critical steps in reinforcement learning

0.1 Explanation A: Inductive biases in ConSpec that enable rapid credit assignment

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: $p(\text{success}|\{O_1, \ldots, O_T\})$, which can be a highly nontrivial non-linear function.

In contrast, ConSpec helps learn policies that hone in on steps that are highly probable given success, i.e. with high values for $p(O_t|\text{success})$. Therefore, ConSpec solves the reverse problem of predicting the critical steps given success, which is a much easier problem.

To see a concrete example, conditioned on the successful acceptance of a paper, one can be sure that all the critical steps had each been achieved (an idea conceived, experiments run, paper written and reviewer concerns addressed). As such, the replay buffers of successful episodes will include each one of these steps. 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 combination of critical steps that predict reward must consider all the potential combinations of critical steps, which means that they face a search of order up to $\mathcal{O}(2^k)$.

0.2 New Experiments

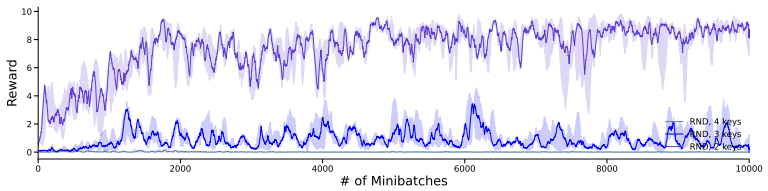

Figure 1: Performance of random network distillation (RND) on the multi-key-to-door task with 2, 3, or 4 keys.

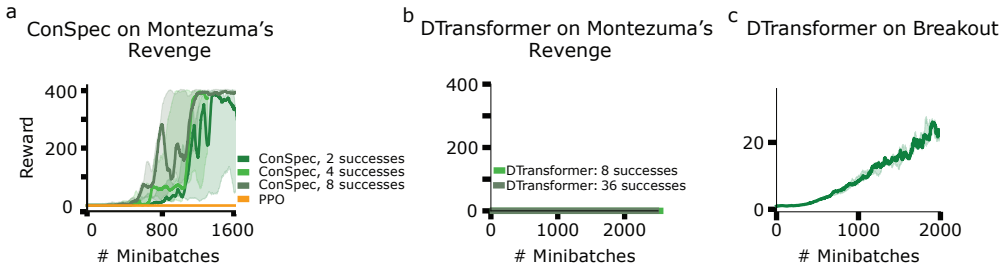

Figure 2: ConSpec improves performance on (a) delayed Bowling and (b) delayed Pong.

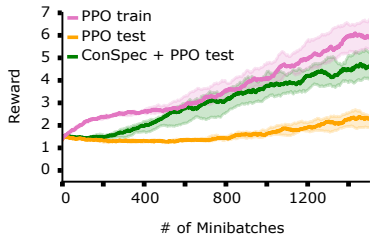

Figure 3: PPO trained on Procgen Dodgeball has a large generalization gap. In this game, ConSpec receives much higher reward during test environments than PPO during test environments and almost as much as as PPO during training environments.

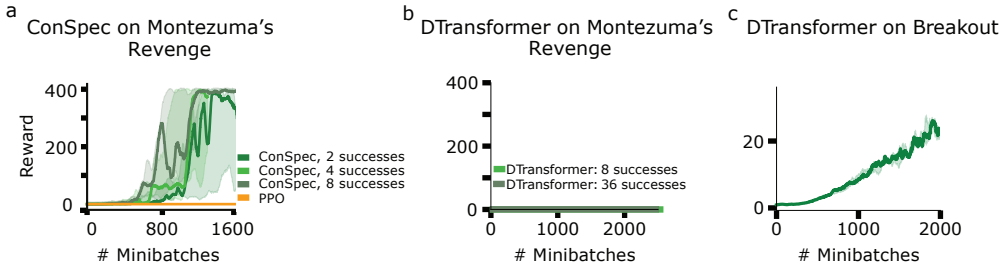

Figure 4: Performance of RUDDER on the multi-key-to-door task with 2, 3, or 4 keys.

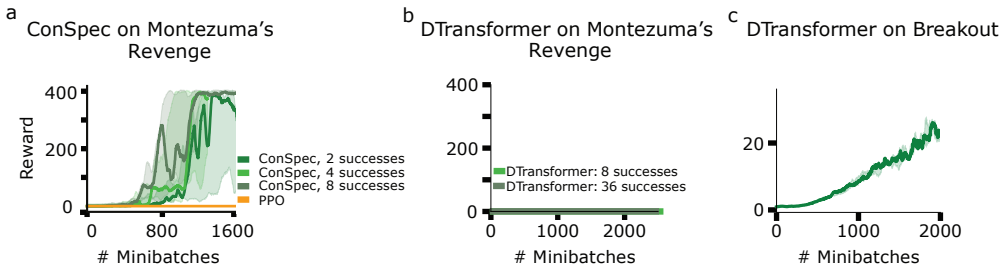

Figure 5: (a) ConSpec learns on Montezuma's Revenge with 2, 4, and 8 success demonstrations, highlighting how ConSpec uses few successes very efficiently. By contrast, Decision transformers (b) do not learn Montezuma's Revenge despite 8 or even 36 success demonstrations, but (c) do learn Breakout (as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

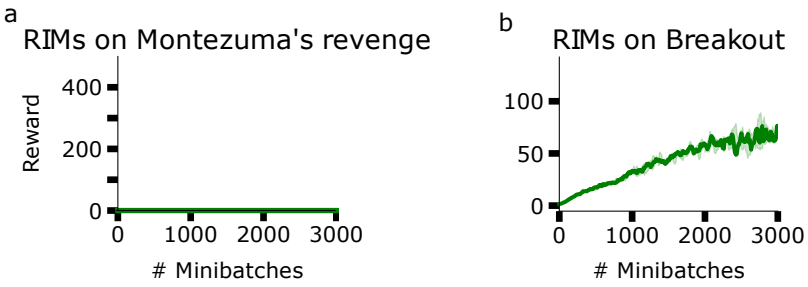

Figure 6: Performance of recurrent independent mechanisms (RIMs) on (a) Montezuma's Revenge, and (b) Breakout