A Framework for Formal Verification to Correct Actions in Reinforcement Learning

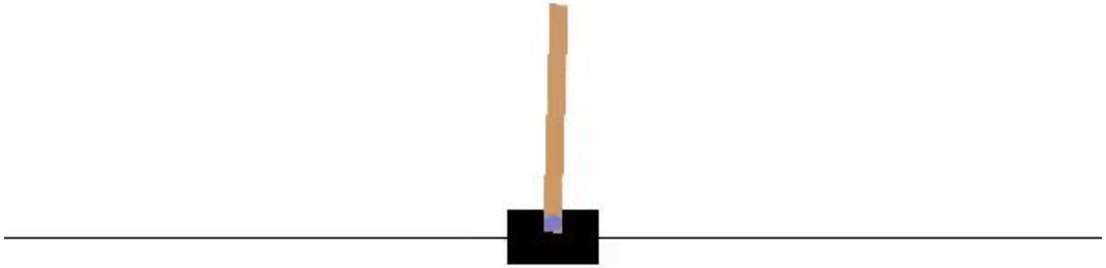
Ethan Hobbs and Vikas Nataraja

Outline

- Reinforcement Learning Background
- Motivation
- Our Approach
- Algorithm
- Implementation
- Results
- Conclusion and Future Work

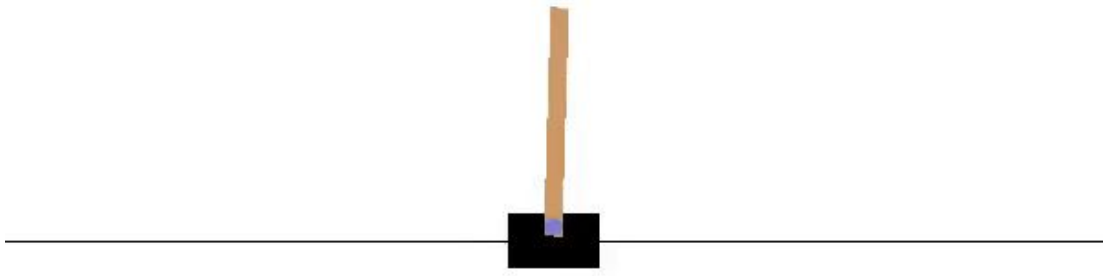
Reinforcement Learning Background

- Cartpole
- Maximize a reward in a given situation.
- +1: every timestep it stays upright
- -1: every time it falls.



Reinforcement Learning Background

- RL agent has to take the “correct” actions at the “correct” states.
- policy $\boldsymbol{\pi}$: how the agent knows what action ‘a’ to take at a state ‘s’.
- States - Safe vs Unsafe



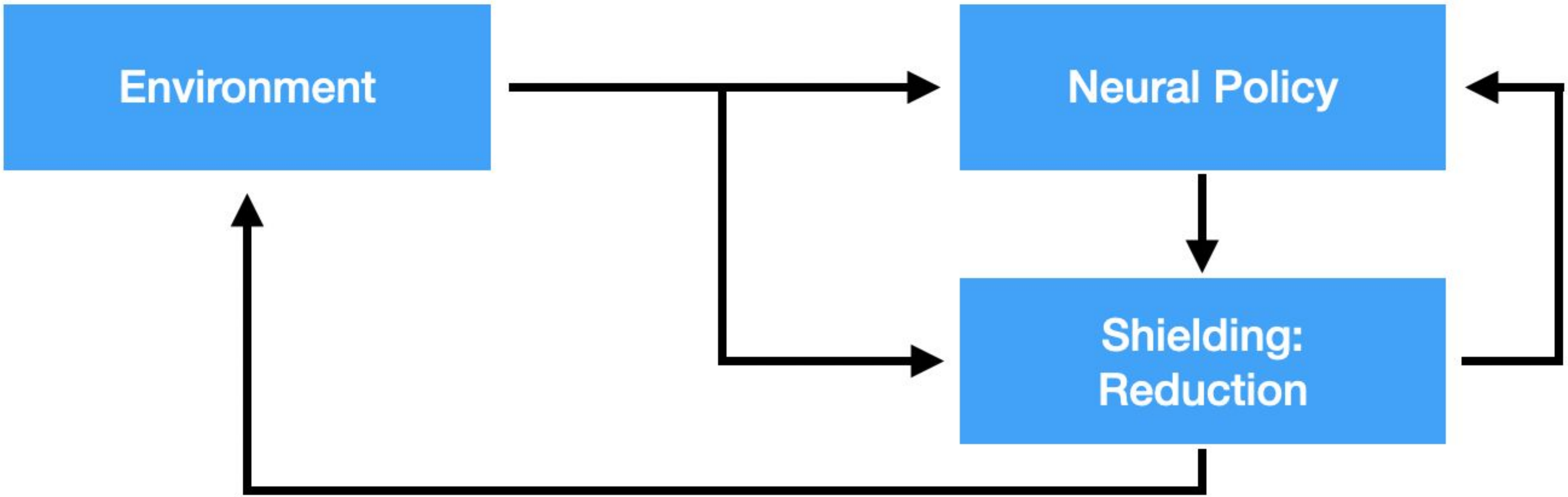
State Verification is Hard

Reachability Analysis **or**

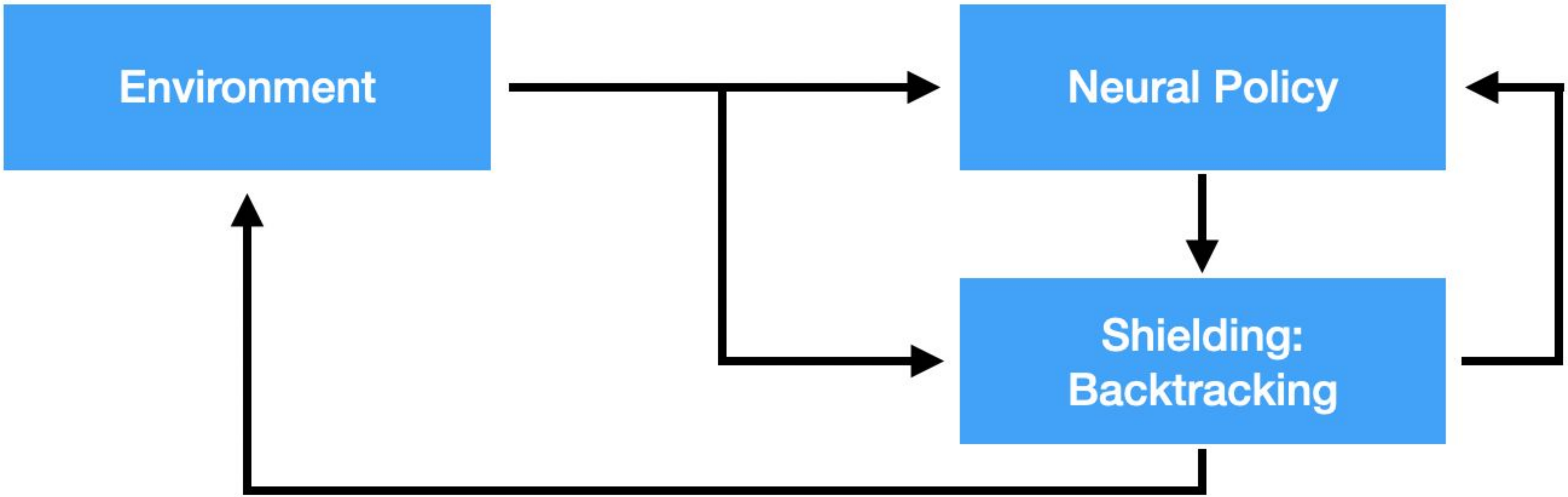
Markov Decision Process

Shielding Methods

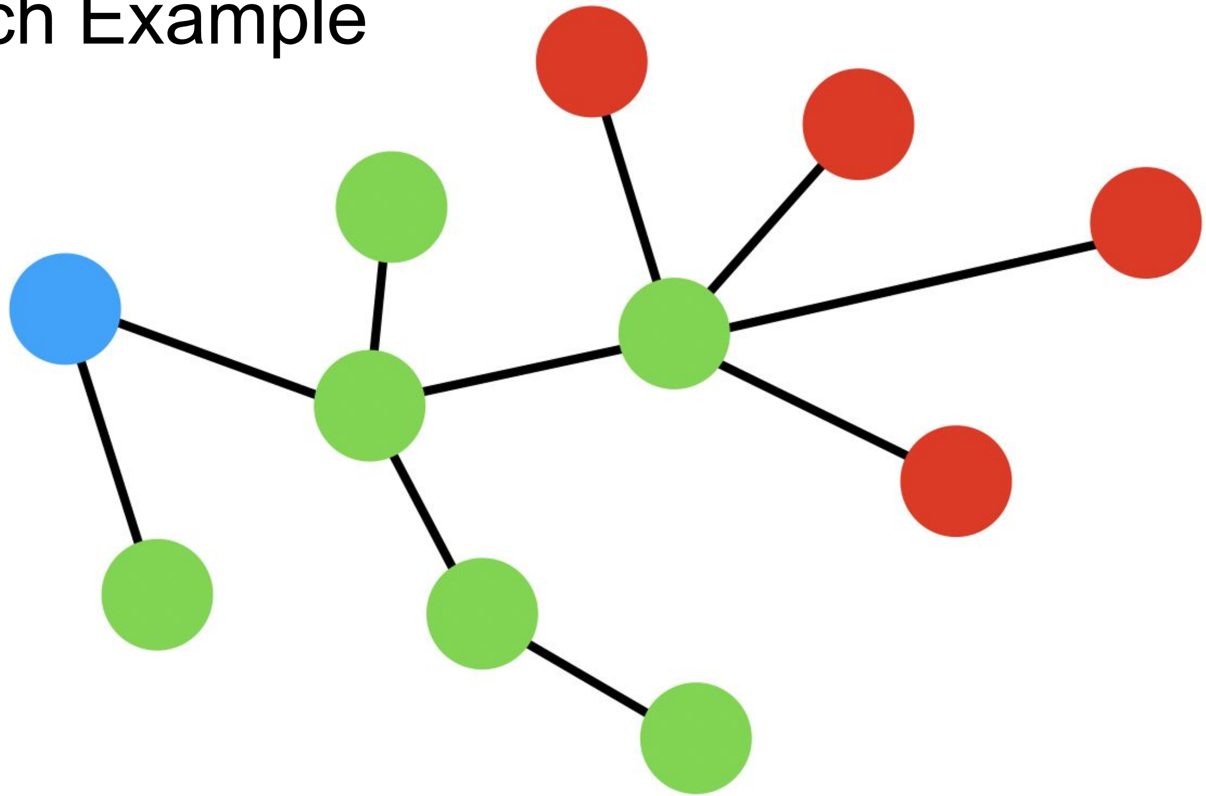
Our Approach



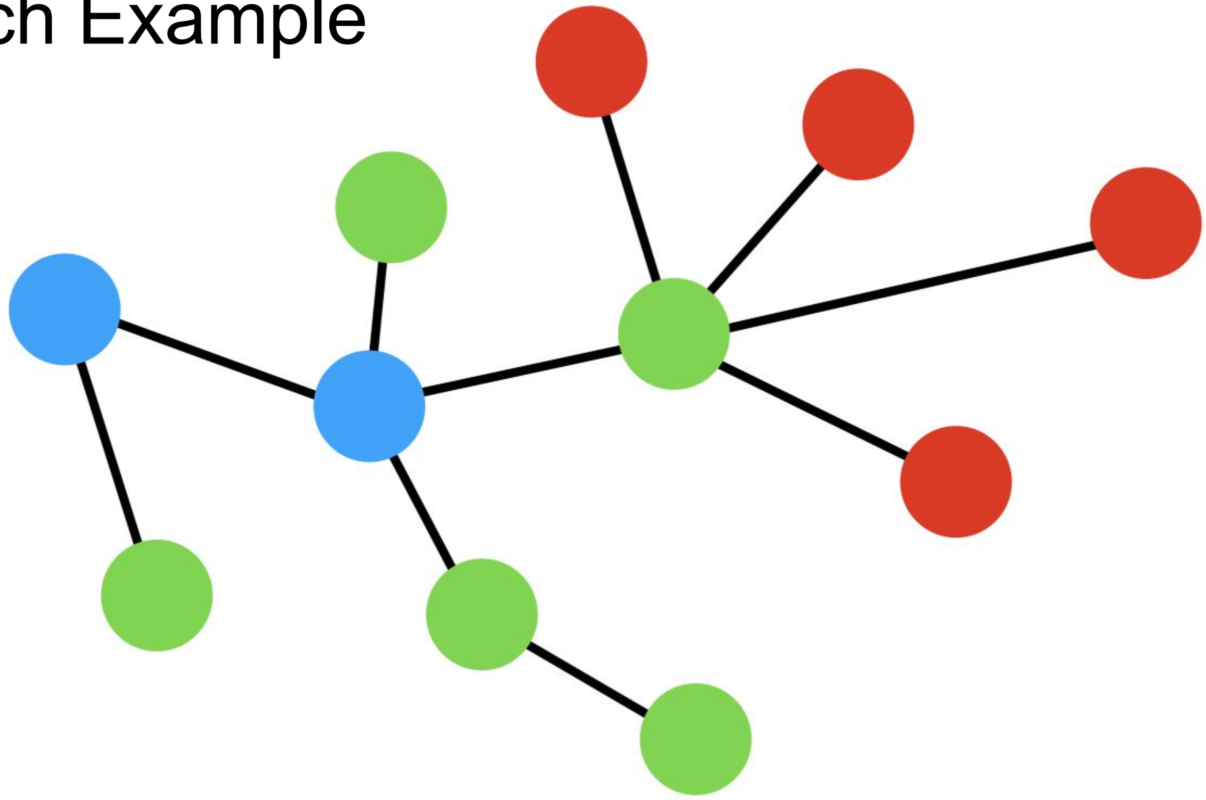
Our Approach



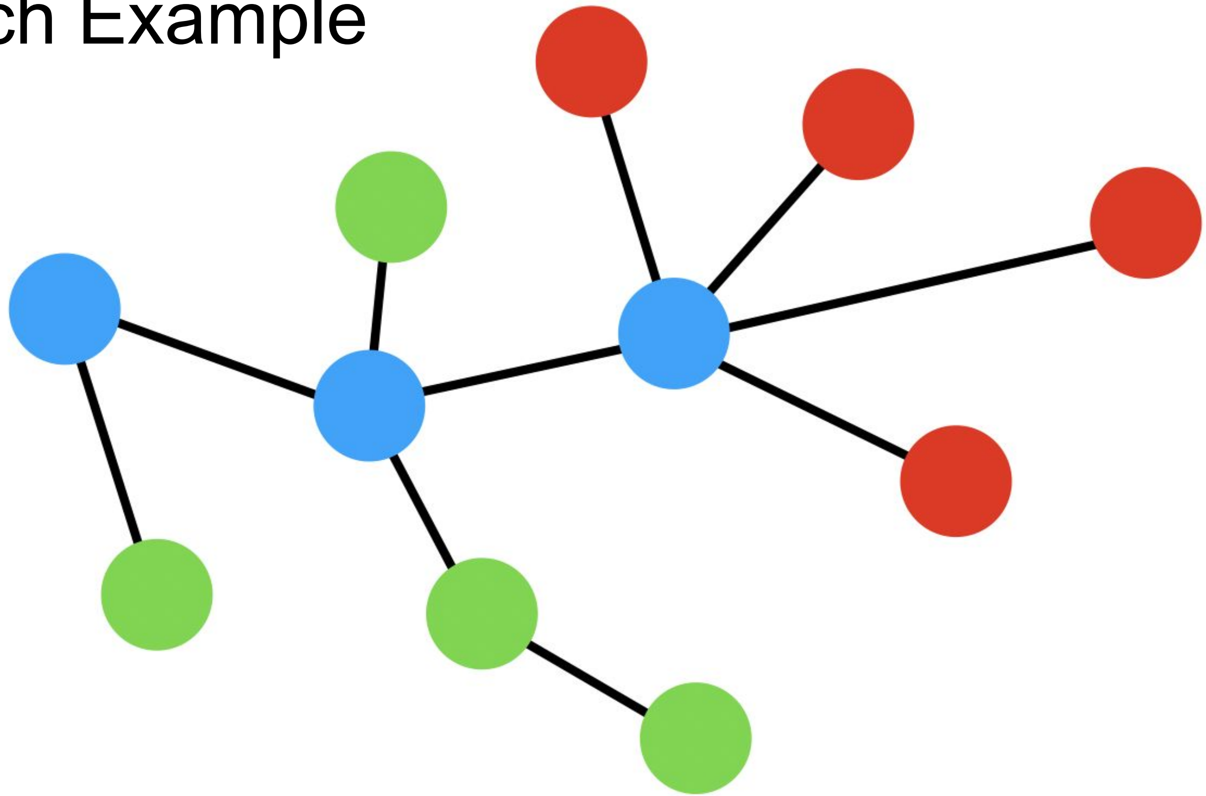
Approach Example



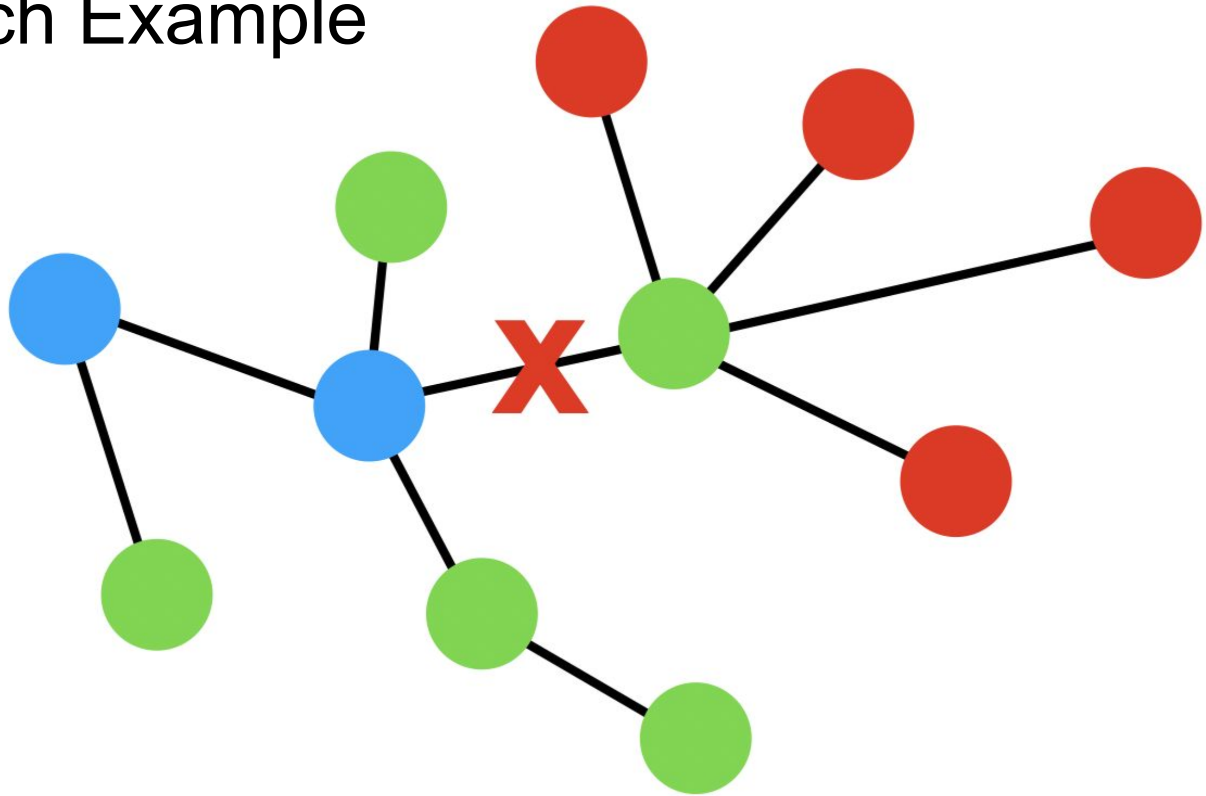
Approach Example



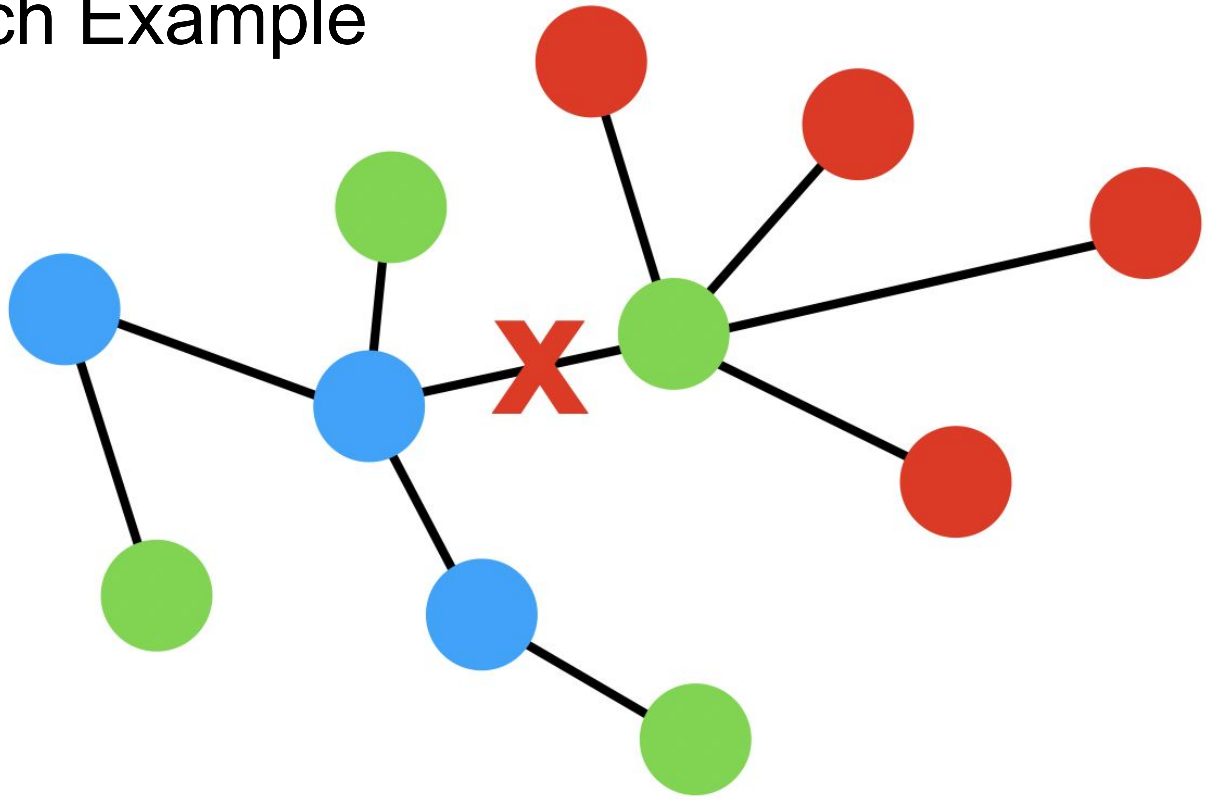
Approach Example



Approach Example

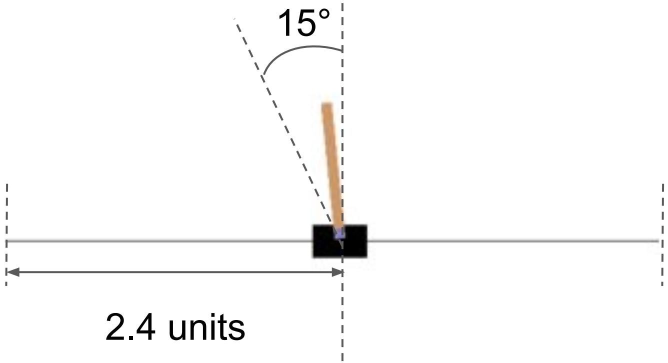


Approach Example

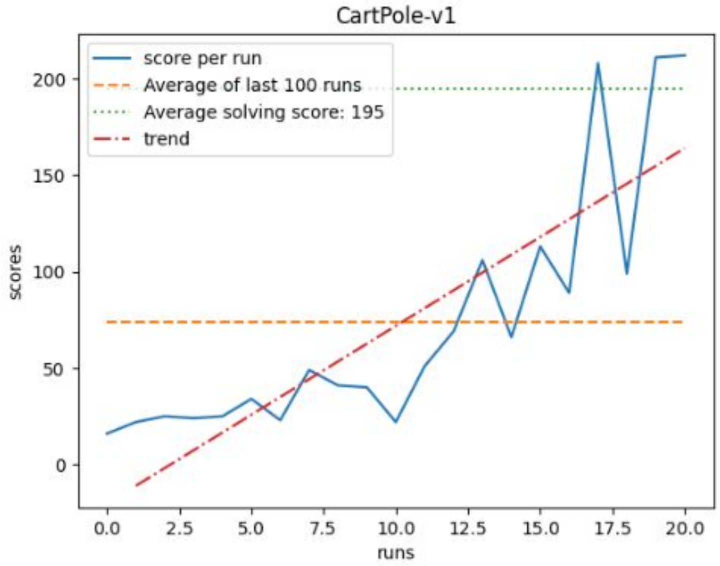


Implementation

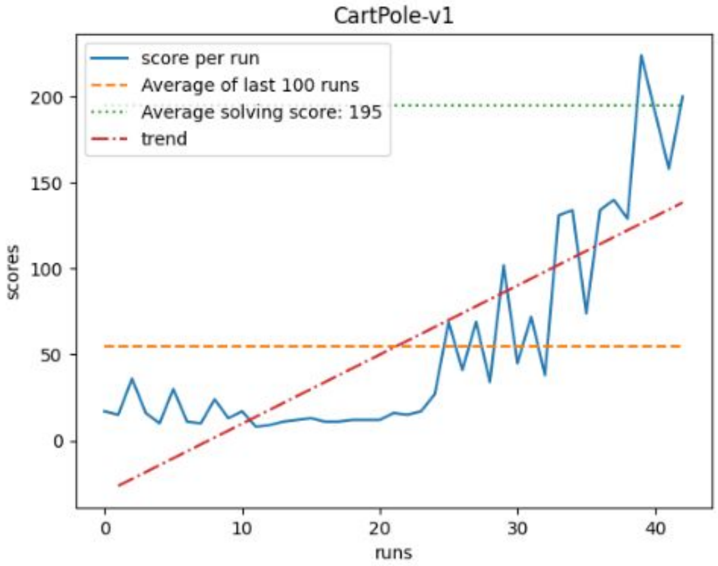
- Used OpenAI gym CartPole scenario
- A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track.
- The system is controlled by applying a discrete force measure to the cart. The pendulum starts upright, and the goal is to prevent it from falling over.
- Reward of +1 for every time step that the cart doesn't fall over.
- Episode ends when the pole moves more than 15° from the vertical or when the cart moves 2.4 units from the center.



Results - All valid states

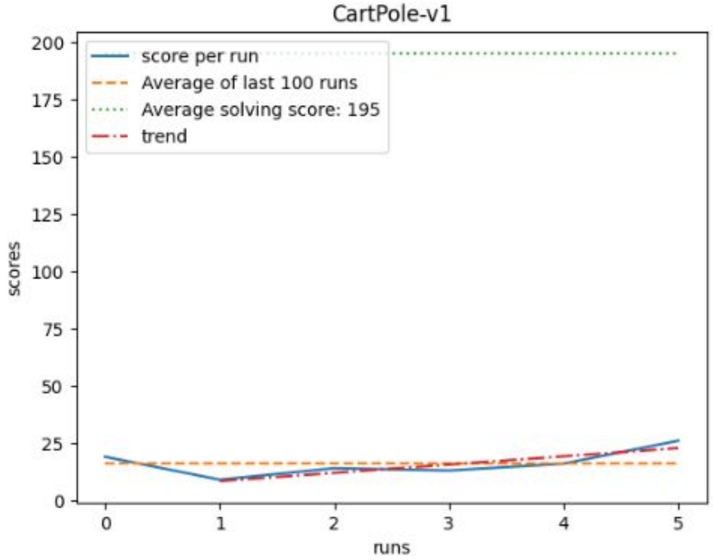


(a) Baseline model evaluated under all valid states

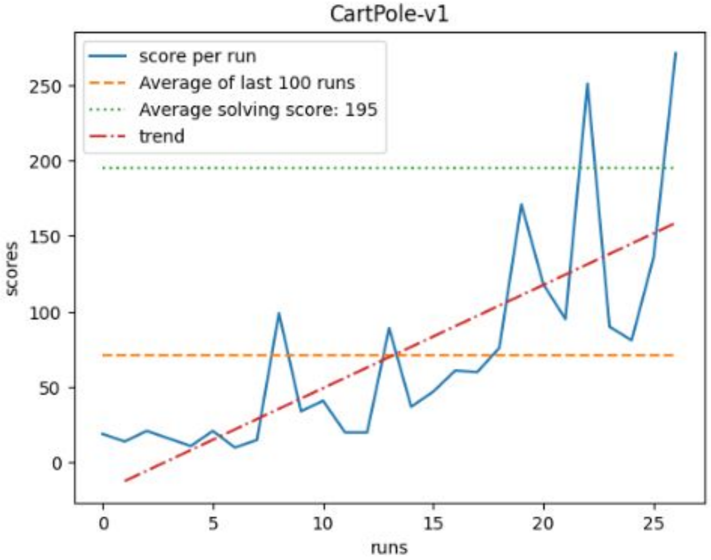


(b) Proposed model evaluated under all valid states

Results - One invalid state

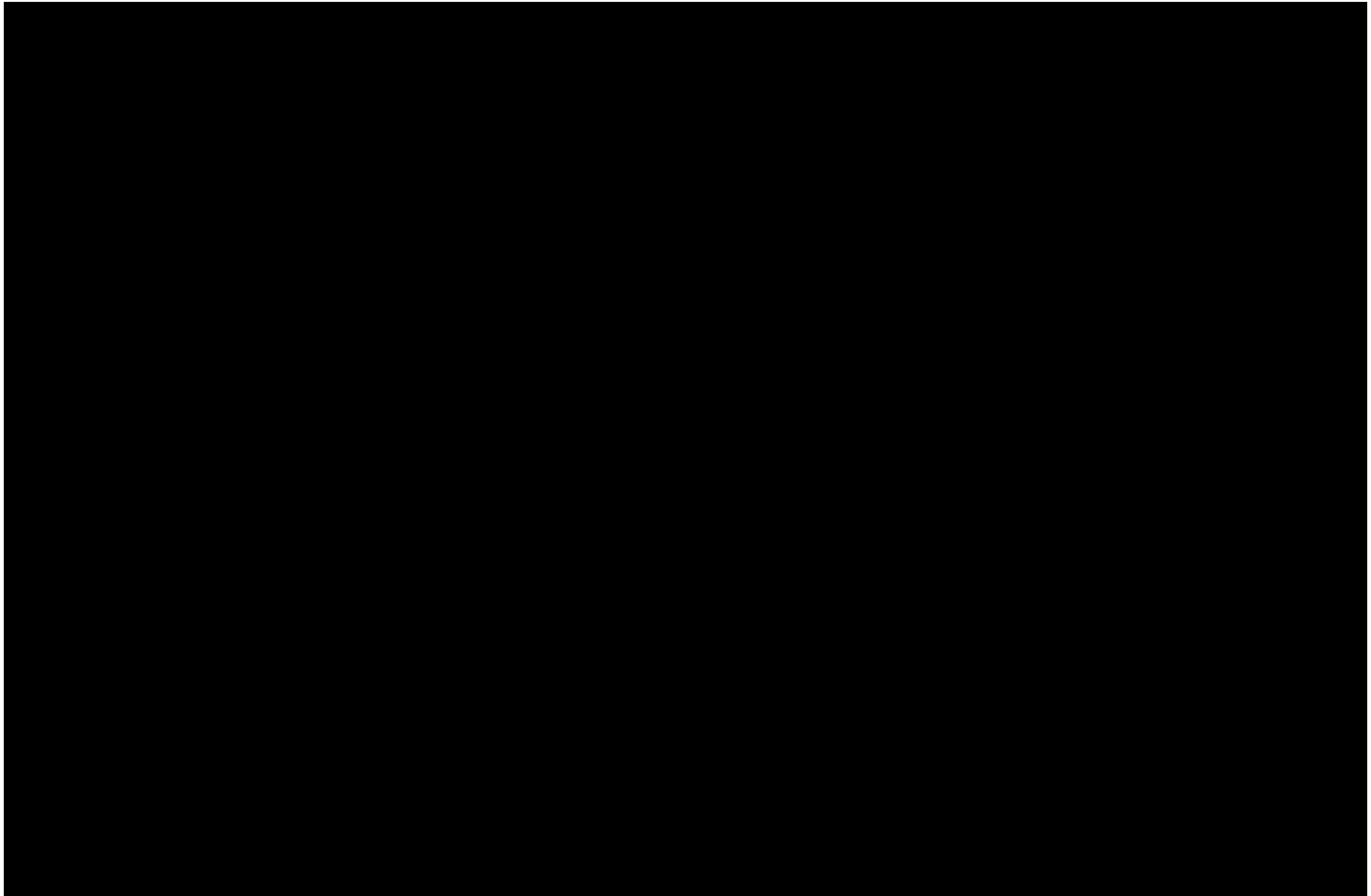


(a) Baseline model evaluated under one invalid state



(b) Proposed model evaluated under one invalid state

Video Demonstration



Conclusion and Future Work

- We propose an alternative approach to verifying validity of states and actions in reinforcement learning.
- Instead of reverting to one of initial states or reducing the state space, we propose using “backtracking” where we use the agent’s history of states and actions to switch to a safe state.
- Our approach although results in a temporary sub-optimal policy guarantees safe state transitions.
- To expand on our approach, we intend to test on other scenarios and dynamics where the environment’s observation space is more complex.
- We also would like to work on making the model perform faster because currently the model gets slower as the look-ahead factor increases

Thank you!

Reinforcement Learning Background

- To maximize the reward, the RL agent has to take the “correct” actions at the “correct” states.
  - This is called a *policy* $\boldsymbol{\pi}$ which is how the agent knows what action ‘a’ to take at a state ‘s’.
  - To do this, a Q-function is used:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_t | S_t = s, A_t = a]$$

- After an action is taken, the outcome is observed and the Q value is updated:

$$\underbrace{\text{New}Q^{\pi}(s,a)}_{\text{New Q-Value}} = Q^{\pi}(s,a) + \underset{\text{Learning Rate}}{\alpha}\, [\underbrace{R(s,a)}_{\text{Reward}} + \underset{\text{Discount factor}}{\gamma}\, \overbrace{\max Q'(s',a')}^{\text{Maximum predicted reward, given new state and all possible actions}} - Q(s,a)]$$

Algorithm

Algorithm 1: Verification Algorithm

```
Synthesize the deterministic program
Project forward n-steps
Set threshold for backtracking timestep thresh
Check the validity of possible future states dependent on γ
if valid states ≥ 1 then
    Q^π(s, a) = E_π[R_t | S_t = s, A_t = a]
    NewQ^π(s, a) = Q^π(s, a) + α[R(s, a) + γ. max Q'(s', a') − Q(s, a)]
else
    search replay buffer history and choose random state with k > thresh
    s = s_k, a = a_k
    Q^π(s, a) = E_π[R_t | S_t = s, A_t = a]
    NewQ^π(s, a) = Q^π(s, a) + α[R(s, a) + γ. max Q'(s', a') − Q(s, a)]
end
```