Thinker: Learning to Plan and Act

Stephen Chung[1] Ivan Anokhin[2] David Krueger[1]

[1] Computational and Biological Learning, University of Cambridge; University of Cambridge

[2] Mila

Background

- **Planning**: Using a world model to simulate future trajectories to inform action selection.
- For example, we imagine different strategies before making a move in chess; similarly, MCTS simulates multiple rollouts before selecting a real action.
- Combined with RL, planning leads to powerful algorithms such as AlphaGo and MuZero.
- But planning algorithms are generally **handcrafted**; e.g., BFS, MCTS.

Drawbacks of handcrafted planning algorithms:

1. Inflexible – same degree of planning irrespective of the state
2. High computational cost – require hundreds of simulations
3. Non-optimal performance – e.g., rely on naïve exploration strategies
4. Not biologically plausible – animals exhibit flexible planning behavior

How can we train an RL agent to learn a planning algorithm?

Thinker: An MDP-augmentation Algorithm

We construct a new MDP from a given MDP to allow planning within the new MDP:

Step 1: Add imaginary steps

We insert a fixed number of ***imaginary steps*** (e.g. 19) before each ***real step*** (see Fig 1 for example).

Imaginary Step

- The agent selects an imaginary action, and the world model predicts the next state and rewards (Fig 1: Step 1-19).
- The agent can opt to *reset* the world model to the current state (Fig 1: Step 5, 10, 15).

Real Step

- The agent selects a real action in the real MDP (Fig 1: Step 20).

The reward of the new MDP equals that of the real MDP, thus fully aligning the agent's objectives across both MDPs.

Step 2: Simplify the augmented MDP

We provide additional information about the predicted state in the augmented MDP's state, including:

1. Real policy $\pi(\hat{s})$, trained with imitation learning on real actions
2. Value $v(\hat{s})$, trained with TD-learning on real trajectories
3. Predicted reward $\hat{r}$, which is learned in the world model
4. Hints, such as visit counts and mean rollout return

This information forms a compact representation that typically has fewer than 100 dimensions and is also included in the state of the augmented MDP, which is much easier to learn from compared to the high-dimensional predicted states.

The resulting MDP, the **Thinker-Augmented MDP**, can be used in conjunction with any RL algorithm.*

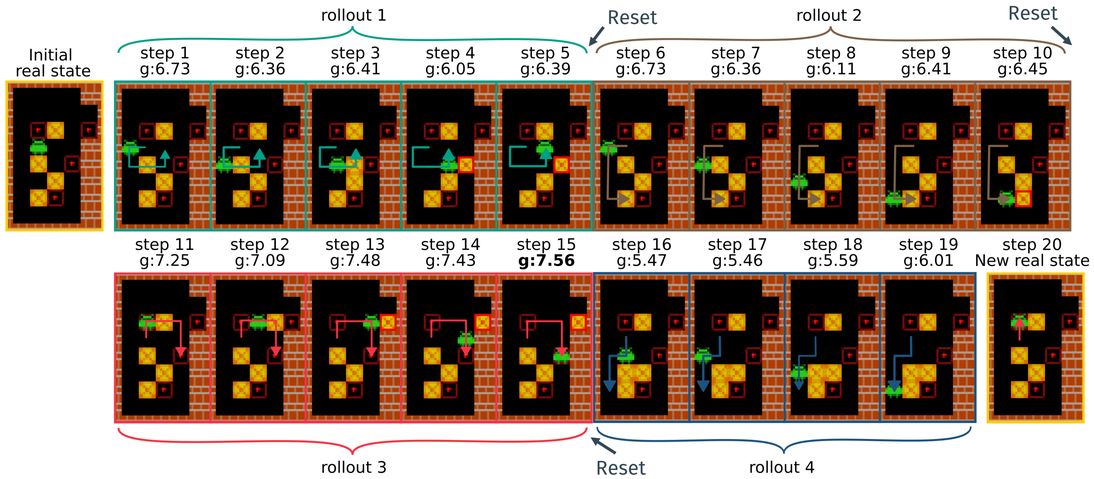


Fig 1. Illustration of a trained agent on the Thinker-augmented MDP of Sokoban, where the goal is to push all four boxes to red-bordered squares. Except for the real states, all frames are generated by a learned model. The agent proposed four reasonable plans that push different boxes towards targets and executed rollout 3 in the next five real steps (not fully shown here). Estimated rollout return g computed within the augmented MDP is displayed. Crucially, **plan generation, plan execution, and the world model are all learned.** Video of the whole game and other Atari games can be found in the paper.

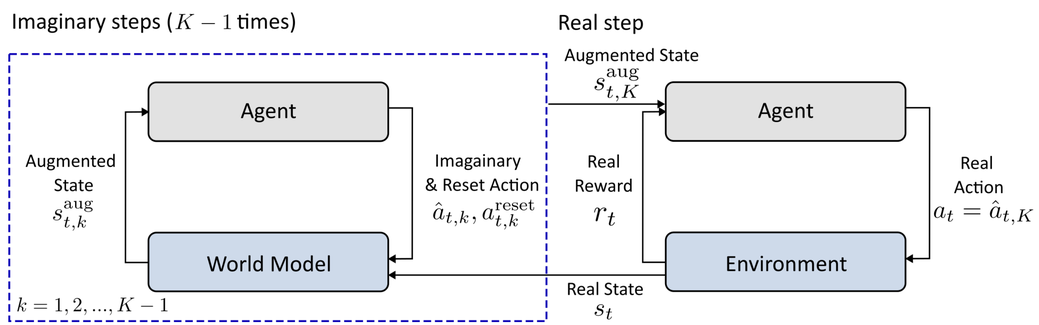


Fig 2. Illustration of the Thinker-augmented MDP. We add K-1 imaginary steps before each real step, and the agent can choose to unroll the world model with the imaginary action or reset the world model to the real state so as to start another rollout.

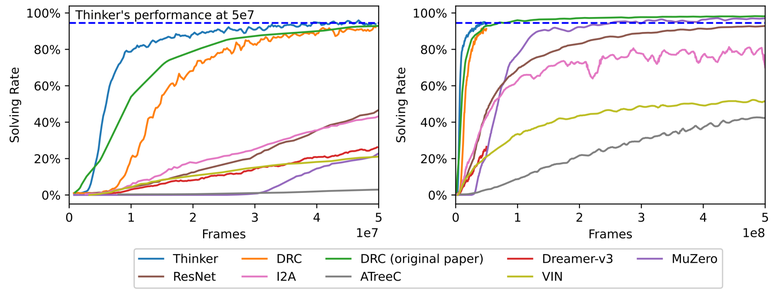
Key Idea: Augmenting the Environment with a World Model and Simplifying the Augmented Environment

* The world model and the estimated real policy and value are trained within the augmented MDP, independent of the RL algorithm applied on the MDP.

Experiment Results

We applied a standard actor-critic algorithm on Thinker-augmented MDP; we used 19 imaginary steps in all experiments.

Learning Curve on Sokoban



New SOTA results on Sokoban, achieving 95% solving rate within 50 million frames; outperforms other model-based RL baselines

Learning Curve on Atari-57

We compare the performance of actor-critic on the Thinker-augmented MDP and the raw MDP of 57 Atari games; the benefit of the Thinker-augmented MDP is significant across diverse sets of environments.

The first work showing that an RL agent can learn to plan with a learned world model in complex environments

Use Thinker in your RL algorithm!

Our publicly available code allows for using the Thinker-augmented MDP with the same interface as OpenAI Gym:

```
import Thinker
name = "BreakoutNoFrameskip-v4"
env = thinker.make(name)
state, reward, done, info = \
    env.step(action=[2], reset=[0])
```

PAPER + CODE