A Consciousness-Inspired Planning Agent for Model-Based RL

PAPER→  ←CODE

By Mingde Zhao, Zhen Liu, Sitao Luan, Shuyuan Zhang, Doina Precup and Yoshua Bengio

What is consciousness? Awareness!

- *What are the 2 axes of consciousness [What is consciousness, and could machines have it, Science 2017]?*
  - C1 (aware of the environment): selective attention towards environment elements. We seek to design a C1-capable planning agent in this work.
  - C2: Self-awareness: C2-capable planning is left for future works.

- *Where it comes from? An evolved prior? Or a learned behavior? Or both?*
  - Controversial. Personally, I think that *the biological structures that enable consciousness are likely from evolution, but the conscious behaviors are likely a result of learning after birth.*

Why is consciousness + planning powerful?

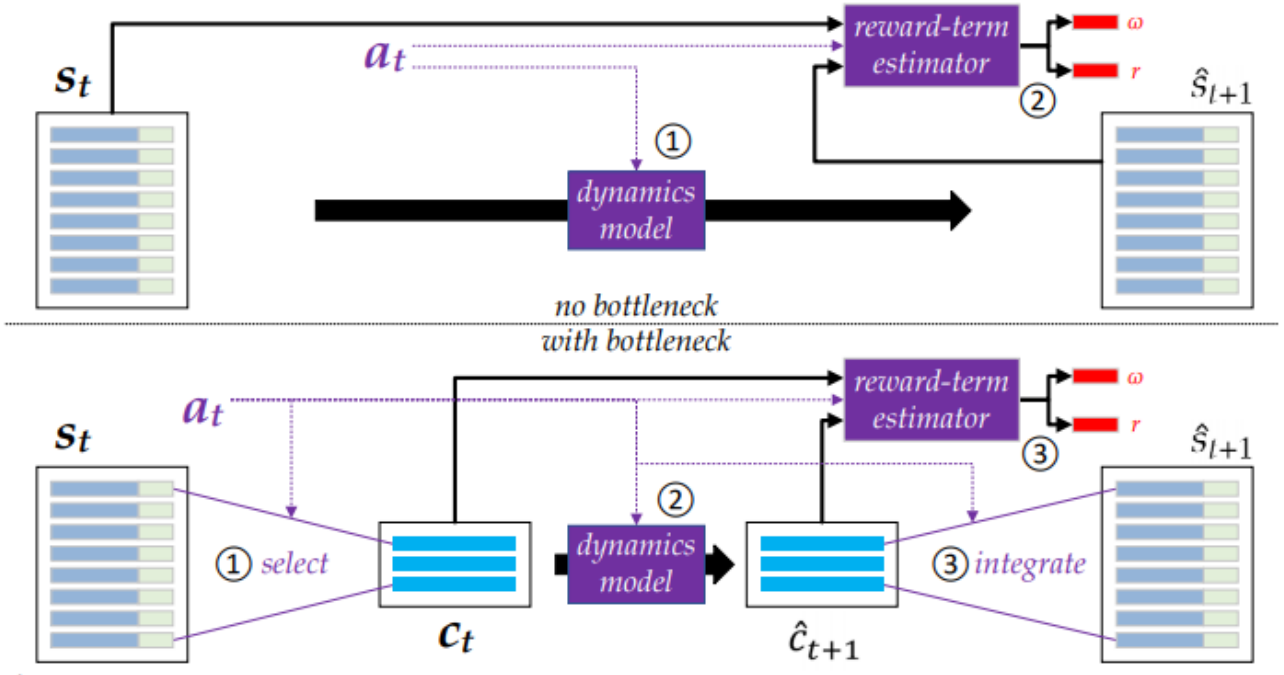
- *What kind of planning is C1-conscious?*
  - That takes into consideration only the parts that are relevant. (the advantages can be imagined ☺)
- *Out-Of-Distribution (OOD) generalization v.s. Memorization*
  - With C1, we can generalize learned skills to OOD scenarios: humans can predict the freefall of an apple no matter we are under the stars of Sahara or on the beach of Vancouver. Note that what we mean by OOD scenarios in this paper is limited to the case where the dynamics used to achieve the task are consistent across different tasks.

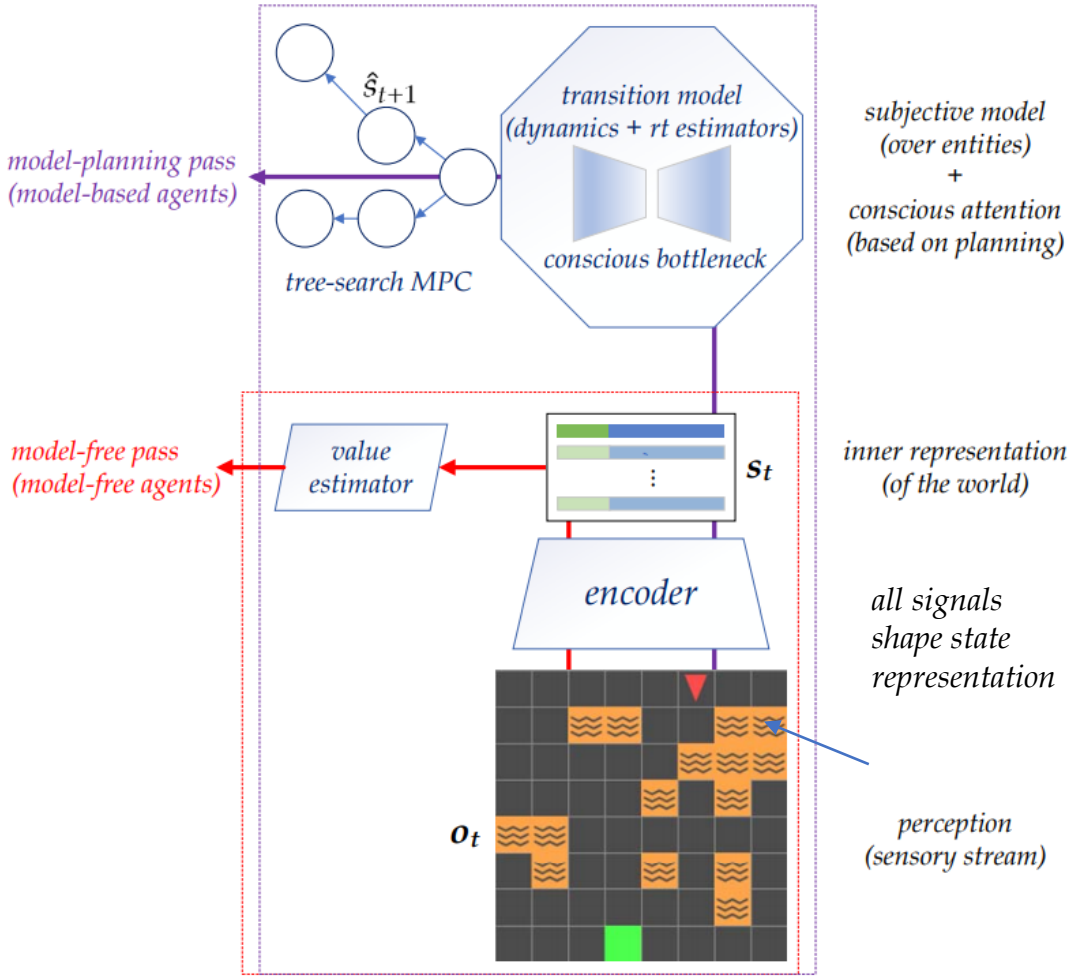
This work

- Shows proof of concept that C1 could help model-based RL significantly (in OOD) via crafted experiments.
- Extracts important intuitions from the experimental observations.
- Proposes effective architectures for set-based dynamics modeling.
- Provides insights on learning representations for RL.

How to enable consciousness + planning?

- *More interpretable representation:*
  - Set-representation based inner-world model
- *Attention-based Bottleneck*
  - Attend to a subset of objects in the state set
  - Conditioned computations
- *Decision-Time Planning*
  - Model predictive control





← *Check the paper for detailed architectures*

How to train?

- Temporal Difference (TD) $\mathcal{L}_{\text{TD}}$: regresses the current state value estimate to the update target, *e.g.* provided by DQN or Double DQN (DDQN) [32, 46]. In experiments, a distributional output is used for both value and reward estimation, making this loss a KL-divergence.
- Dynamics Consistency $\mathcal{L}_{\text{dyn}}$: A squared L_2 distance established between the aligned $\hat{s}_{t+1}$ and s_{t+1}, where $\hat{s}_{t+1}$ is the imagined next (latent) state given o_t, a_t and s_{t+1} is the true next (latent) state encoded from o_{t+1}.
- Reward Estimation $\mathcal{L}_r$: the KL-divergence between the imagined reward $\hat{r}_{t+1}$ predicted by the model and the true reward r_{t+1} of the observed transition.
- Termination Estimation $\mathcal{L}_\omega$: the binary cross-entropy loss from the imagined termination $\hat{\omega}_{t+1}$ to the ground truth ω_{t+1}, obtained from environment feedback.

The resulting total loss for end-to-end training of this set-based MBRL agent is thus[1]:

$$\mathcal{L} = \mathcal{L}_{\text{TD}} + \mathcal{L}_{\text{dyn}} + \mathcal{L}_r + \mathcal{L}_\omega$$

Jointly shaping the states avoids the representation collapsing to trivial solutions and makes them useful for all signal predictions of interest. In our experimental implementation, no recurrent mechanism is used however the same training procedure is naturally extendable.

Experiments?

Observations: arrays of objects (including empty cells)

Crafted with Gym-Minigrid + BabyAI

Non-static (randomized) In-distribution Training

+

A gradient of Non-static (randomized) OOD Evaluation

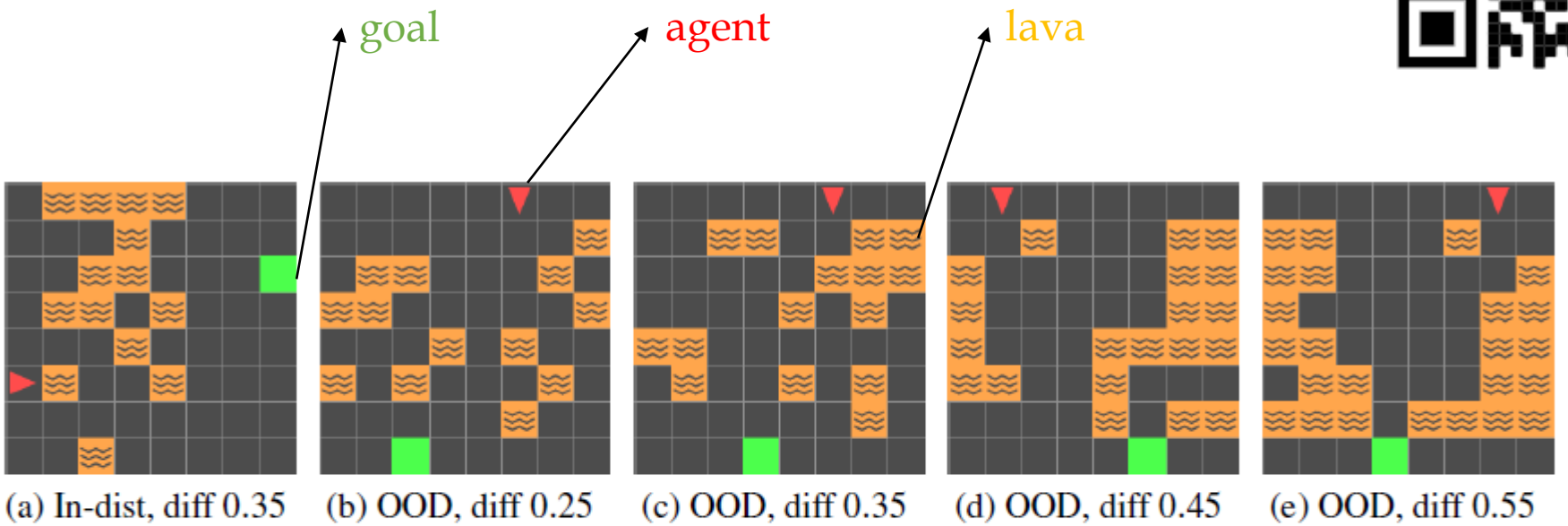
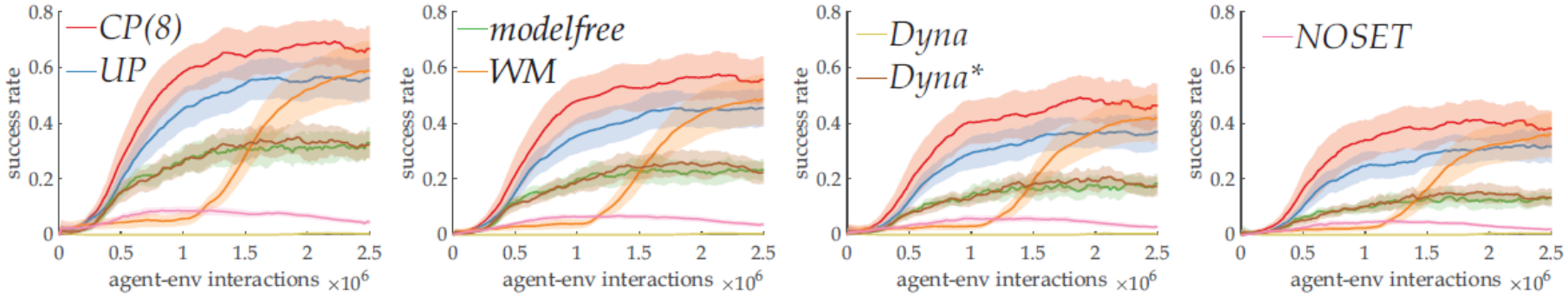


Figure 5: **Non-Static RL Setting, with in-distribution and OOD tasks**: (a) example of training environments (b - e) examples of OOD environments (rotated 90 degrees, changing the distribution of grid elements). For OOD testing, we evaluate different levels of difficulty (b - e). The agent (red triangle) points in the forward movement direction. The goal is marked in green. For each episode (training or OOD), we randomly generate a new world from a sampling distribution. Note that the training environments and the OOD testing environments have no intersecting observations.

Hit the lava: episode terminates, layout changes
Hit the goal: reward +1, episode terminates, layout changes

Results: navigation w/ turn-OR-forward dynamics



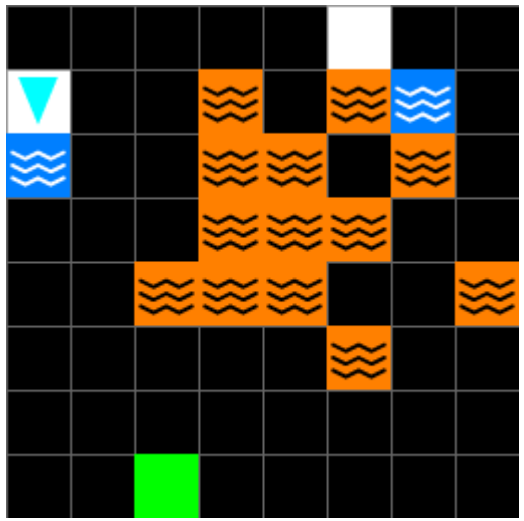
(a) OOD, difficulty 0.25 (b) OOD, difficulty 0.35 (c) OOD, difficulty 0.45 (d) OOD, difficulty 0.55

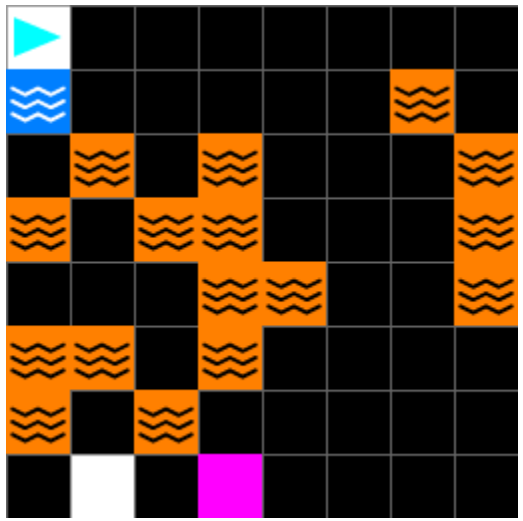
Figure 7: **OOD performance under a gradient of difficulty.** The figures show a consistent pattern: the MPC-based end-to-end agent equipped with a bottleneck (CP) performs the best. All error bars are obtained from 20 independent runs.

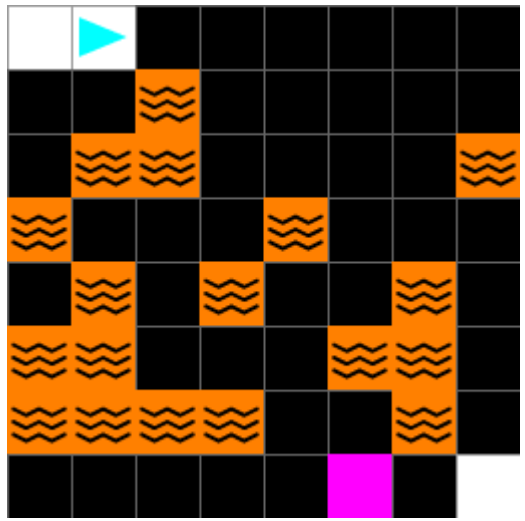
CP(n): conscious planning agent with bottleneck size n.
UP: unconscious planning agent.
modelfree: baseline agent with set-based state encoder,
WM: a world-model baseline which spends 1M steps for exploration
Dyna: a dyna agent with perfect model accuracy.*
NOSET: planning agent with vectorized state encoder

For all sets of experiments, we compare against multiple baselines with architectures as similar as possible. The number of seed runs for each curve (+bar) is 20.

Visualization of Attention during Planning

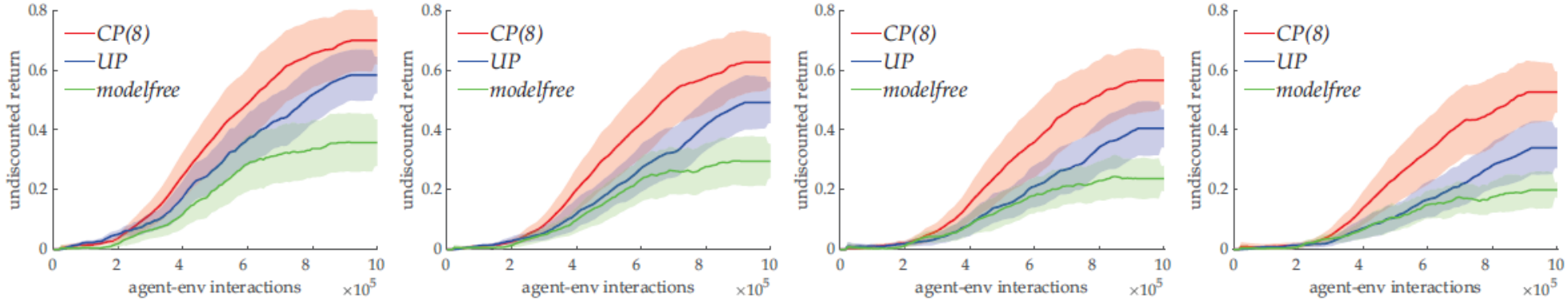






Using semi-hard attention therefore no other cells are attended to
These figures have no statistical purposes but only for intuitive visualization.

Results: navigation w/ turn-AND-forward dynamics



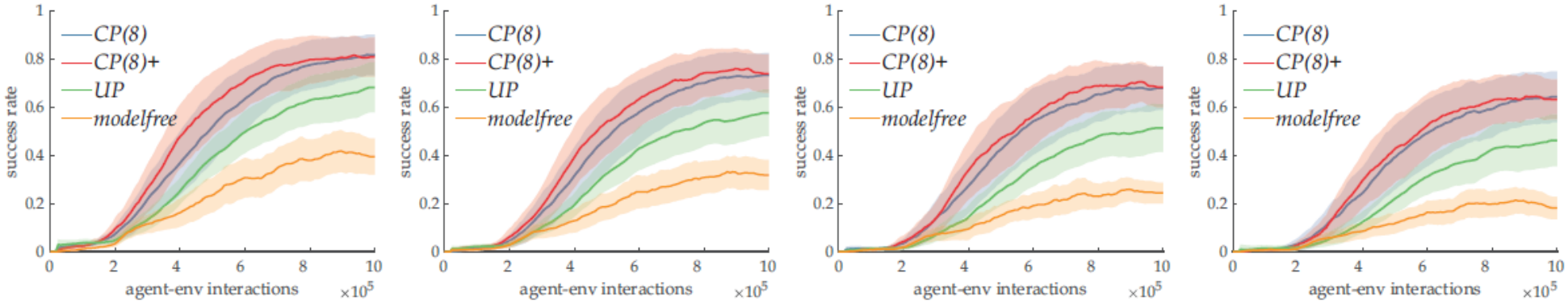
(a) OOD 0.25 (b) OOD 0.35 (c) OOD 0.45 (d) OOD 0.55

Figure 20: **OOD performance under a gradient of difficulty in Turn-and-Forward tasks.** All error bars are obtained from 20 independent runs.

CP(n): conscious planning agent with bottleneck size n.
UP: unconscious planning agent.
modelfree: baseline with set-based state encoder

***Turn-OR-Forward**: the agent turns left, right or goes forward based on the current facing direction.*
***Turn-AND-Forward**: the agent chooses to turn to one of the four directions based on the current facing direction (relative not absolute) and then forward.*

Results: navigation w/ turn-AND-forward dynamics & color cluttering



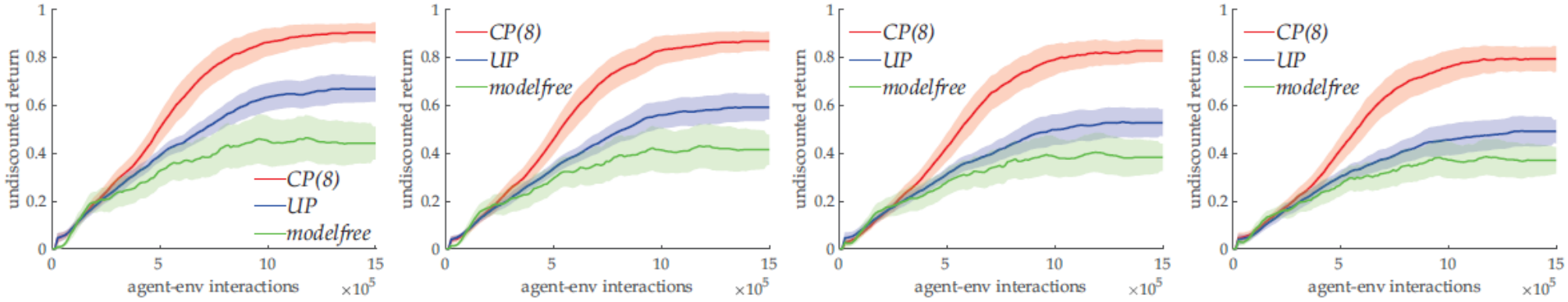
(a) OOD 0.25 (b) OOD 0.35 (c) OOD 0.45 (d) OOD 0.55

Figure 21: **OOD performance under a gradient of difficulty in Turn-and-Forward tasks with color distractions.** All error bars are obtained from 20 independent runs.

CP(n): conscious planning agent with bottleneck size n.
CP(n)+: conscious planning agent with bottleneck size n. The model is trained with noise injected.
UP: unconscious planning agent.
modelfree: baseline with set-based state encoder

Color cluttering: the colors of objects are changing constantly across the episode.

Results: navigate & unlock w/ turn-and-forward dynamics



(a) OOD 0.25 (b) OOD 0.35 (c) OOD 0.45 (d) OOD 0.55

Figure 22: **OOD performance under a gradient of difficulty in Key-Chest Unlock Task.** The y-axes values are undiscounted cumulative episodic return. All error bars are obtained from 20 independent runs.

CP(n): conscious planning agent with bottleneck size n.
UP: unconscious planning agent.
modelfree: baseline with set-based state encoder

In these tasks, the agent needs to find a key on the map first, or it would not be able to unlock the goal. The task is built upon the simple navigation tasks, i.e. the agent still cries before the lava field.

Ablation

- Planning steps: 1 to 6
- World Sizes: 6x6 to 10x10
- Static *v.s.* non-static setting: the struggle of the NO-SET baseline
- Decision Quality: via DP
- Bottleneck Size: some small size is good
- Regularization: inverse modeling and layernorms
- Model Accuracy: cumulative, in-dist and OOD
- Types of Attention: soft *v.s.* semihard
- Unsupervised representation learning: potentials

Interesting Q&As

Q: Can you define what you mean by OOD in this work?

A: We focus on skills *transferrable* to totally different environments with *consistent dynamics*. This means the environmental dynamics that is sufficient for solving the in-distribution training tasks and the OOD evaluation tasks are consistently preserved, while the rest can be very different. Intuitively, we want to train our agent to be able to plan routes in the home city and expect this ability to be generalized to totally different municipality. The places can be very different, but the route planning skill depends on the knowledge that is quite universal.

Q: Why do you employ a non-static setting where environments change every episode?

A: Intuitively, the agent has no need to understand the dynamics of the task if the environment is fixed. Learning to memorize where to go and where not to is far easier than reasoning about what may happen. With the challenges, it became necessity for the agent to indeed learn and understand the environment dynamics, which is crucial for OOD evaluation.

Q: How do you avoid collapse in the state representation?

A: Apart from the training signal for dynamics, whose exclusive existence might cause collapse, all other training signals go through the common bottleneck of the encoder thus the representation is also shaped by the TD signals, the reward-termination prediction signals. We also want to mention that the regularizations participate in this as well.

Q: Why not use richer architectures for experiments?

A: Since there are already many components in the proposed agent, to isolate possible factors that would influence the agent behaviors, we use the minimal architecture sizes that marginally enables good RL performance on our test settings.

Q: Why not use MCTS as the search method for decision-time planning?

A: Unlike AlphaGo, which employs MCTS, our architecture is based on the simplest baseline DQN. DQN uses a value-estimator based greedy policy instead of a parameterized one as the actor-critic architecture used in AlphaGo. The DQN policy is deterministic *w.r.t.* the estimated values therefore the planning cannot take sufficient advantage of the sample based MCTS.