Predicting Future Actions of Reinforcement Learning Agents

Stephen Chung[*]
University of Cambridge

Scott Niekum
University of Massachusetts Amherst

David Krueger
Mila

Abstract

As reinforcement learning agents become increasingly deployed in real-world scenarios, predicting future agent actions and events during deployment is important for facilitating better human-agent interaction and preventing catastrophic outcomes. This paper experimentally evaluates and compares the effectiveness of future action and event prediction for three types of RL agents: explicitly planning, implicitly planning, and non-planning. We employ two approaches: the inner state approach, which involves predicting based on the inner computations of the agents (e.g., plans or neuron activations), and a simulation-based approach, which involves unrolling the agent in a learned world model. Our results show that the plans of explicitly planning agents are significantly more informative for prediction than the neuron activations of the other types. Furthermore, using internal plans proves more robust to model quality compared to simulation-based approaches when predicting actions, while the results for event prediction are more mixed. These findings highlight the benefits of leveraging inner states and simulations to predict future agent actions and events, thereby improving interaction and safety in real-world deployments.

1 Introduction

As reinforcement learning (RL) becomes increasingly applied in the real world, ensuring the safety and reliability of RL agents is paramount. Recent advancements have shown that agents can exhibit complex behaviors, making it crucial to understand and anticipate their actions. This is especially important in scenarios where misaligned objectives [1] or unintended consequences could result in suboptimal or even harmful outcomes. For instance, consider an autonomous vehicle controlled by an RL agent that might unpredictably decide to run a red light to optimize travel time. Predicting this behavior in advance would enable timely intervention to prevent a potentially dangerous situation. This capability is also beneficial in scenarios that require effective collaboration and information exchange among multiple agents [2–4]. For example, if passengers and other drivers know whether a self-driving car will turn left or right, it becomes much easier and safer to navigate the roads. Thus, the ability to accurately predict an agent's future behavior can help reduce risks and ensure smooth interaction between agents and humans in real-world situations.

In this paper, we explore the task of predicting future actions and events when deploying a trained agent, such as whether an agent will turn left in five seconds. The distribution of future actions and events cannot be computed directly, even with access to the policy, because the future states are unknown. We consider two methods for predicting future actions and events: the inner state approach and the simulation-based approach. We apply these approaches to agents trained with various RL algorithms to assess their predictability[1].

[*]Correspondence to: mhc48@cam.ac.uk

[1]Full code is available at https://github.com/stephen-chung-mh/predict_action

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

In the *inner state approach*, we assume that we have full access to the *inner state* of the agent during deployment. Here, the inner state refers to all the intermediate computations required to determine the final action executed by the agent, such as the simulation of the world model for explicit planning agents or the hidden layers for agents parametrized by deep neural networks. We seek to answer the following questions: (i) How informative are these inner states for predicting future actions and events? (ii) How does the predictability of future actions and events vary across different types of RL agents with different inner states?

As an alternative to the inner state approach, we explore a *simulation-based approach* by unrolling the agent in a learned world model and observing its behavior. Assuming we have a sufficiently accurate world model that resembles the real environment, this simulation should provide valuable information for predicting future actions and events in the real environment. We seek to answer the following question: (iii) How do the performance and robustness of the simulation-based approach compare to the inner state approach in predicting future actions and events across different agent types?

We conduct extensive experiments to address the above research questions. To summarize, the main contributions of this paper include:

1. To the best of our knowledge, this is the first work to formally compare and evaluate the predictability of different types of RL agents in terms of action and event prediction.
2. We propose two approaches to address this problem: the inner state approach and the simulation-based approach.
3. We conduct extensive experiments to evaluate the effectiveness and robustness of these approaches across different types of RL agents, demonstrating that the plans of explicitly planning agents are more informative for prediction than other types of inner states.

2 Background and Notation

We consider a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, d_0)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function representing the dynamics of the environment, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\gamma \in [0, 1]$ is a discount factor, and $d_0 : \mathcal{S} \to [0, 1]$ is an initial state distribution. Denoting the state, action, and reward at time t by S_t, A_t, and R_t respectively, $P(s, a, s') = \Pr(S_{t+1} = s' | S_t = s, A_t = a)$, $R(s, a) = \mathbb{E}[R_t | S_t = s, A_t = a]$, and $d_0(s) = \Pr(S_0 = s)$, where P and d_0 are valid probability mass functions. An episode is a sequence of (S_t, A_t, R_t), starting from $t = 0$ and continuing until reaching the terminal state, a special state where the environment ends. Letting $G_t = \sum_{k=t}^{\infty} \gamma^{k-t} R_k$ denote the infinite-horizon discounted return accrued after acting at time t, an RL algorithm attempts to find, or approximate, a *policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, such that for any time $t \geq 0$, selecting actions according to $\pi(s, a) = \Pr(A_t = a | S_t = s)$ maximizes the expected return $\mathbb{E}[G_t | \pi]$.

In this paper, *planning* refers to the process of interacting with an environment simulator or a world model to inform the selection of subsequent actions. Here, a world model is a learned and approximated version of the environment. We classify an agent, which is defined by its policy, into one of the following three categories based on the RL algorithm by which it is trained:

Explicit Planning Agents. In explicit planning agents, an environment simulator or a world model is used explicitly for planning. We consider two explicit planning agents in this paper, MuZero [5] and Thinker [6], given their superior ability in planning domains. MuZero is a state-of-the-art model-based RL algorithm that combines a learned model with Monte Carlo Tree Search (MCTS) [7, 8] for planning. During planning, MuZero uses the learned model to simulate future trajectories and performs MCTS to select the best action based on the predicted rewards and values. Thinker is a recently proposed approach that enables RL agents to autonomously interact with and use a learned world model to perform planning. The key idea of Thinker is to augment the environment with a world model and introduce new actions designed for interacting with the world model. MuZero represents a handcrafted planning approach, while Thinker represents a learned planning approach.

Implicit Planning Agents. In implicit planning agents, there is no learned world model nor an explicit planning algorithm, yet these agents still exhibit planning-like behavior. A notable example is the Deep Repeated ConvLSTM (DRC) [9], which excels in planning domains. DRC agents are trained

by actor-critic algorithms [10] and employ a unique architecture based on convolutional-LSTM with internal recurrent steps. The authors observe that the trained agents display planning-like properties, such as improved performance with increased computational allowance, and so argue that the agent learns to perform model-free planning.

Non-planning Agents. In non-planning agents, there is neither a learned world model nor an explicit planning algorithm, and these agents do not exhibit planning-like behavior. Typically, these agents perform poorly in planning domains. Examples include most model-free RL algorithms, such as the actor-critic and Q-learning. In this paper, we focus exclusively on IMPALA [10], a variant of the actor-critic algorithm, chosen for its computational efficiency and popularity.

We believe that this distinction between RL agents, adopted from previous work [11], is useful for investigating their predictability. We hypothesize that the plan made by an explicit planning agent should be more informative of future actions or events than that of an implicit planning agent, as the plan in an explicit planning agent is typically human-interpretable, whereas the plan for an implicit planning agent is stored in hidden activations. Nevertheless, the computation in these two types of agents provides indications of their future actions and thus should carry more information than the hidden activations of a non-planning agent, which lacks future plans and may merely serve as a more compact representation of the state.

3 Problem Statement

Given a fixed policy π, we aim to estimate the distribution of a function of the future trajectory. For example, we may want to estimate the probability of an agent entering a particular state or performing a specific action within a certain horizon. Mathematically, let $H_t = (S_t, A_t, R_t)$ denote the transition at step t, and let $H_{t:T} = \{H_t, H_{t+1}, \dots, H_T\}$ denote the future trajectory from step t to the last step T. Let $\mathcal{H}$ denote the set of all possible future trajectories. We are interested in estimating the distribution of a random variable $f(H_{t:T})$ conditioned on the current state and action:

$$\mathbb{P}(f(H_{t:T}) \mid S_t, A_t), \tag{1}$$

where $f : \mathcal{H} \to \mathbb{R}^m$ is a function specifying the variables to be predicted.

This paper focuses on predicting two particular types of information. The first type is *future action prediction*, where we want to predict the action of the agent in L steps, i.e., $f(H_{t:T}) = (A_{t+1}, A_{t+2}, \dots, A_{t+L})$ and the problem becomes estimating:

$$\mathbb{P}(A_{t+1}, A_{t+2}, \dots, A_{t+L} \mid S_t, A_t). \tag{2}$$

An example of action prediction is whether an autonomous vehicle is going to turn left or right in the next minute. The second type is *future event prediction*, where we want to estimate the probability of a binary indicator $g : (\mathcal{S}, \mathcal{A}) \to \{0, 1\}$ being active within L steps, and the problem becomes estimating:

$$\mathbb{P}\left(\bigcup_{k=1}^{L} g(S_{t+k}, A_{t+k}) = 1 \,\middle|\, S_t, A_t\right), \tag{3}$$

which is equivalent to the case $f(H_{t:T}) = \max\{\, g(S_{t+k}, A_{t+k})\}_{k=1,\dots,L}$. In other words, (3) is the probability of the event defined by g occurring within L steps. An example of event prediction is predicting whether an autonomous vehicle will run a red light within a minute.

Event prediction shares resemblance to the generalized value function [12], where $f(H_{t:T}) = \sum_{k=0}^{\infty} \gamma^k g(S_{t+k}, A_{t+k})$ and we estimate its expectation $\mathbb{E}[f(H_{t:T}) \mid S_t, A_t]$. When g is a binary indicator, this expectation is equivalent to the discounted sum of the probabilities of the event defined by g. This is arguably harder to interpret than (3); for example, it can be larger than 1 and thus is not a valid probability.

To learn these distributions, we assume access to some transitions generated by the policy π as training data. The transitions may come from multiple episodes. In the case of future event prediction, we assume that $g(S_t, A_t)$ is also known for each transition. We further assume that the policy π and the inner computation for each action A_t within π is known.

In this work, we assume that π is already a trained policy and is fixed. This is the case where the agent is already deployed, and transitions during the deployment are collected. In cases where the training and deployment environments are similar, we can also use the transitions when training the agent as training data for predicting future actions and events, but this is left for future work.

4 Methods

Since we already have the state-action (S_t, A_t) and the target output $f(H_{t:T})$ in the training data, we can treat the problem as a supervised learning task.[2] In particular, We can train a neural network that takes the state-action pair as input to predict $f(H_{t:T})$. This network is trained using gradient descent on cross-entropy loss.

Besides the state-action, there can be additional information that may help the prediction. For example, the inner computation of the policy π may contain plans that are informative of the agent's future actions, especially in the case of an explicit planning agent. We refer to this information that is available before observing the next state S_{t+1} as auxiliary information and denote it as I_t. We will consider two types of auxiliary information: inner states and simulations.

4.1 Inner State Approach

In the inner state approach, we consider choosing the agent's inner state as the auxiliary information. Here, the inner state refers to all the intermediate computations required to compute the action A_t. As inner states are different across different types of agents and may not all be useful, we consider the following inner state to be included in the auxiliary information:

1. MuZero: Since MuZero uses MCTS to search in a world model and selects action with the largest visit count, we select the most visited rollout as the auxiliary information. Rollouts here refer to the simulation of the world model and are composed of a sequence of transitions $(\hat{S}_{t+l}, \hat{A}_{t+l}, \hat{R}_{t+l})_{1 \leq l \leq L}$. It should be noted that the agent may not necessarily select the action sequence of this rollout, as the MCTS is performed at every step, and the search result at the next step may yield different actions. We do not use all rollouts, as MCTS usually requires many rollouts.
2. Thinker: We select all rollouts and tree representations during planning as the auxiliary information. We do not choose a particular rollout because, unlike MCTS, in Thinker, it is generally unknown which action the agent will select at the end, and Thinker usually requires only a few rollouts.
3. DRC: We select the hidden states of the convolutional-LSTM at every internal tick as the inner state, as it was proposed that the hidden state contains plans to guide future actions [9].
4. IMPALA: We select the final layer of the convolutional network as the inner state, as it is neither too primitive which may only be processing the state, nor too refined which may only contain information for computing the current action and values.

Experiment results on the alternative choices of inner states can be found in Appendix D.

4.2 Simulation-based Approach

As an alternative to the inner state approach, we can train a world model concurrently with the agent. Once trained, we can simulate the agent in this world model using the trained policy π to generate rollouts. These rollouts can then be utilized as auxiliary information for the predictor. In this paper, we consider using the world model proposed in Thinker [6], which is an RNN that takes the current state and action sequence as inputs and predicts future states, rewards, and other relevant information. For both implicit and non-planning agents, the world model is trained in parallel with the agents but is not used during their selection of actions. Instead, the world model is solely employed to generate rollouts as auxiliary information for the predictors.

If the learned world model closely resembles the real environment, we expect these rollouts to yield valuable information for predicting future actions and events, as the agent's behavior in the world model should be similar to its behavior in the actual environment. In the ideal case where the world model perfectly matches the true environment dynamics, we could compute the exact future action and event distribution without needing any prediction. However, we do not consider this scenario in the paper, as this assumption is impractical for most settings.

[2]Temporal-difference methods are not directly applicable here, as both the action and event prediction tasks involve a limited horizon L and do not sum over variables.

It should be noted that in the simulation-based approach, the world model must always predict the same state space as the input to the agent, enabling the simulation of the agent within the world model. Since agents typically receive raw state inputs (with exceptions such as Dreamer [13], where agents receive abstract state inputs), the world model should also make predictions in the raw state space rather than in a latent state space.. Consequently, the simulation-based approach may not be suitable for situations where learning a world model in the raw state space is challenging, such as predicting camera input in real-world autonomous driving scenarios.

5 Experiments

We conduct three sets of experiments to evaluate the effectiveness and robustness of the discussed approaches. First, we apply the inner state approach to predict future actions and events. We compare it to the case where only state-action information is provided to the predictor so as to evaluate the benefits of the proposed inner state in the prediction. Second, we apply the simulation-based approach and compare it with the inner state approach to evaluate the benefits of these two different types of auxiliary information. Finally, we consider a model ablation setting, where we deliberately make the world model inaccurate to see how the different approaches perform under such conditions.

We consider the Sokoban environment, where the goal is to push all four boxes into the four red-bordered target spaces as illustrated in Fig 1. We choose this environment because (i) a wide range of levels in Sokoban make action and event prediction challenging, and we can evaluate the predictors on unseen levels to evaluate their generalization capability; (ii) there are multiple ways of solving a level; (iii) Sokoban is a planning-based domain, so it may be closer to situations where we want to discern plans of agents in more complex settings.

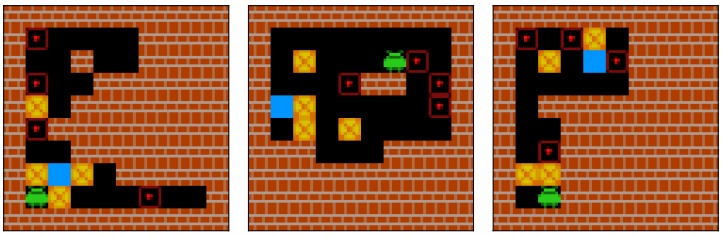

Figure 1: Example levels of Sokoban, where the goal is to push all four boxes into the four red-bordered target spaces. A box can only be pushed, not pulled, making the level irrecoverable if the boxes get stuck. We paint a random empty space blue (which still acts as an empty tile) and predict whether the agent will stand on the blue location within 5 steps.

We choose the prediction horizon L to be 5 in all experiments. For action prediction, we try to predict the next five actions $A_{t+1}, A_{t+2}, ..., A_{t+5}$. For event prediction, we randomly select an empty tile in the level and paint it blue. That blue tile acts as an empty tile to the agent and serves no special function. We define the event g that is to be predicted as the case where the agent stands on the blue location. In other words, we try to predict whether the agent will go to that blue location within L steps.

We train four different agents using MuZero, Thinker, DRC, and IMPALA. All agents are trained for 25 million transitions. To ensure that the result would not be affected by the particular choice of the world model, we uniformly employ the world model architecture proposed in Thinker, as the world model in Thinker predicts the raw state and is suitable for both MuZero and the simulation-based approach. We train a separate world model for each agent. For DRC and IMPALA, the world model is not needed for the policy and will only be used in the predictors in the simulation-based approach.

After training the agents, we generate 50k transitions, where part or all of it will be used as the training data for the predictors. We evaluate the performance of predictors with varying training data sizes: 1k, 2k, 5k, 10k, 20k, 50k. We also generate 10k transitions as a testing dataset. For simplicity, we use greedy policies, where we select the action with the largest probability instead of sampling. The predictor uses a convolutional network to process all image information, including the current state and states in rollouts (if they exist). The encoded current state, along with other auxiliary information such as encoded states, rewards, and actions in rollouts (if they exist), will be passed to a

three-layer Transformer encoder [14], and the final layer predicts the next L actions or the probability of the event within L steps. More details about the experiments can be found in Appendix A.

5.1 Inner State Approach

Figure 2 presents the final accuracy of action prediction and the F1 score of event prediction using the inner state approach. The error bars represent the standard deviation across three independently trained predictors. The accuracy here refers to the percentage of correctly predicting all the next five actions, with no credits awarded if any action is predicted incorrectly. The graph also shows the performance of the predictors when they only receive the current state S_t and action A_t as inputs, as indicated by 'baseline'. Several observations can be made.

First, when access to the plans is available, the prediction accuracy for both action and event is significantly higher for explicit planning agents. For example, with 50k training data, the action prediction accuracy of the MuZero agent increases from 40% to 87% when given access to the plans. Agents using handcrafted planning algorithms (MuZero) or learned planning algorithms (Thinker) show similar performance gains. This is perhaps not surprising, as these explicit planning agents tend to follow the plans either by construction or by learning, and the explicit nature of planning facilitates easy interpretation of the plans.

Second, the case for implicit planning agents (DRC) and non-planning agents (IMPALA) is more nuanced. For action prediction accuracy, both receive a moderate improvement from accessing the hidden state. There are two possible explanations: (i) plans of the agents are stored in the learned representations that are informative of future actions; (ii) the hidden states or hidden layers contain a latent representation that is easier to learn from, compared to the raw states. To discern between the two cases, interpreting the nature and the underlying circuit of the inner states is required. We leave this to future work as interpretability is outside the scope of this paper.

Third, in contrast to action prediction, the inner state does not improve event prediction for DRC and IMPALA, likely because the blue location in the environment does not affect the reward, and the agent may ignore it in its representation. This suggests an advantage of explicit planning agents, as in explicit planning agents, we explicitly train the world model and can train it to attend not just to the reward-relevant features but to all features (or features we deem useful) in the environment. This may be important for cases where the reward function is not well designed, leading to the agent ignoring certain features that are, in fact, important to us.

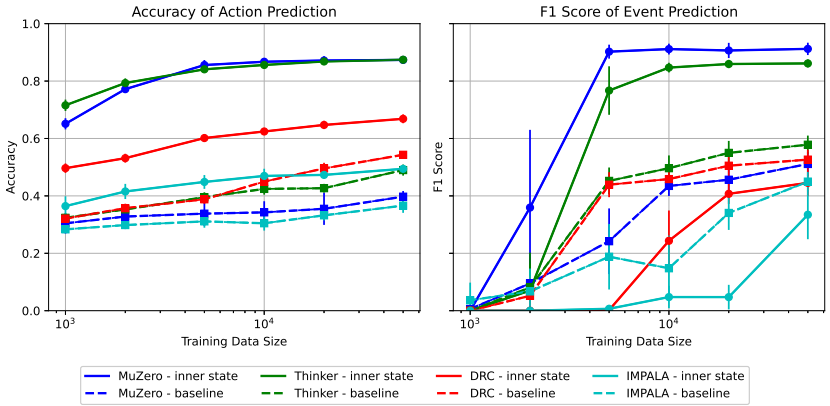

Figure 2: Final accuracy of action prediction and F1 score of event prediction with inner state approach on the testing dataset. The error bar represents two standard errors across 9 seeds.

5.2 Simulation-based Approach

We now consider applying the simulation-based approach to both implicit planning (DRC) and non-planning agents (IMPALA). We unroll the world model for $L = 5$ steps using the current policy and input this rollout as auxiliary information to the predictors. We can use a single rollout, as both the policy and the chosen world model are deterministic, so all rollouts will be the same.

We do not apply the simulation-based approach to explicit planning agents because (i) rollouts already exist as an inner state within the agent and can be input to the predictor, and (ii) it requires training a world model that can be unrolled for $2L$ steps instead of only L steps, as the agent needs to perform planning on every step in the simulated rollout. For a fair comparison, we assume we can only train a world model that is unrolled for L steps in all setups.

Figure 3 shows the final accuracy of action prediction and the F1 score of event prediction for the simulation-based approach of DRC and IMPALA. For easy comparison, we also include the result of the inner state approach of explicit planning agents in the figure.

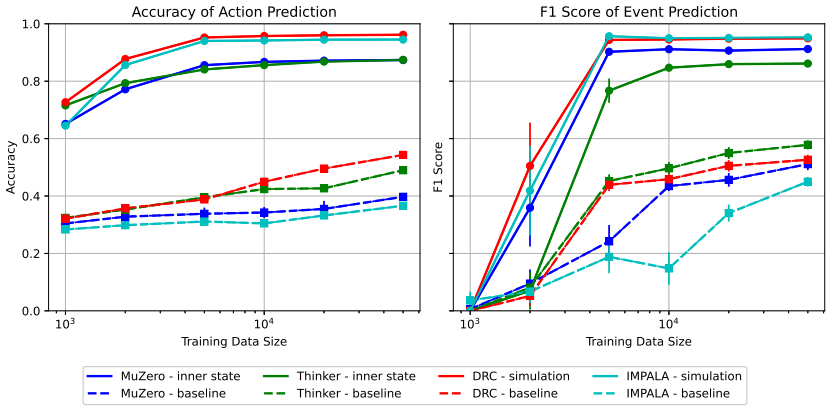

Figure 3: Final accuracy of action prediction and F1 score of event prediction with simulation-based approach (DRC and IMPALA) on the testing dataset. The absolute performance can be found in Appendix C. The error bar represents two standard errors across 9 seeds.

We observe that the predictors for DRC and IMPALA agents in the simulation-based approach perform very well, with performance surpassing that of the explicit planning agents with the inner state approach. This is because the world model we trained is very close to the true environment, so the behavior in the rollout is almost equivalent to that in the real environment. The high-quality world model also enables accurate prediction of when the agent will stand on the blue location, resulting in excellent event prediction performance.

5.3 World Model Ablation

Learning an accurate world model may not be feasible in some settings, such as auto-driving in the real world. An inaccurate world model will affect the plan quality of explicit planning agents, rendering the plan less informative in the inner state approach. An inaccurate world model will also affect the quality of rollouts in the simulation-based approach, leading to inconsistent behaviour between rollouts and real environments. As such, it is important to understand how the inner state approach and simulation-based approach differ when the learned world model is not accurate.

To investigate this, we designed three different settings where learning an accurate world model is challenging. In the first setting, we use a world model with a much smaller size, making it more prone to errors. In the second setting, we randomly replace the agent's action with a no-operation 25% of the time, introducing stochastic dynamics into the environment. However, since the world model we use is deterministic, it cannot account for such stochastic transitions and will yield errors. In the third setting, we consider a partially-observable Markov decision process (POMDP) case, where we randomly display the character at a position within one step of the true character location. As the world model we use only observes the current state, this will lead to uncertainty over both the true character location and the displayed character location. We repeat the above experiments in these three different settings.

Figure 4 shows the change in the final accuracy of action prediction and the F1 score of event prediction for the model ablation settings compared to the default setting. We observe that in terms of action prediction, the accuracy generally drops less in the inner state approach of explicit planning agents than in the simulation-based approach of the two other agents. This is likely because planning

does not necessitate an accurate world model, as plans can still be made without the ability to perfectly predict the future. For example, in MCTS, only values and rewards need to be predicted well, but not the state. In contrast, if we simulate an agent in a poor world model, the agent may be confused as the states may be out of distribution and never encountered during training. This leads to inconsistent behavior compared to the agent's behavior in the real environment.

In contrast, the results for event prediction are more nuanced, with the inner state approach sometimes performing better and the simulation-based approach performing better at other times. We conjecture that because the world model is not accurate, the event under consideration is often not predicted correctly. As such, more informative plans that can predict future actions do not help in event prediction, leading to mixed results.

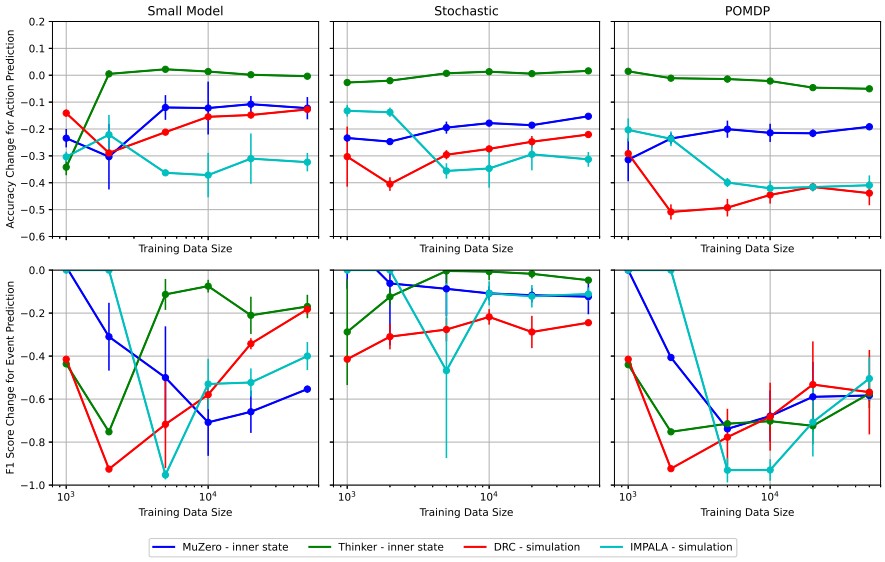

Figure 4: Change in the final accuracy of action prediction and F1 score of event prediction for the world model ablation settings. The error bar represents two standard errors across 3 seeds.

6 Related Works

Safe RL: Our work is related to safe RL, where we try to train an agent to maximize rewards while satisfying safety constraints during the learning and/or deployment processes [15]. A wide variety of methods have been proposed in safe RL, such as shielding, where one manually prevents actions that violate certain constraints from being executed [16], and Constrained Policy Optimization (CPO), which performs policy updates while enforcing constraints throughout training [17]. Many works in safe RL are based on correcting the action when deemed unsafe. Dalal et al. [18] fit a linear model to predict the violation of constraint functions and use it to correct the policy; Cheng et al. [19] project the learned policy to safe policy based on the barrier function; Thananjeyan et al. [20] guide the agent back to learned recovery zones when it is predicted that the state-action pair will lead to unsafe regions; Thomas et al. [21] use world models to predict unsafe trajectories and change the rewards to penalize safety violations.

In contrast to the works in safe RL, we are solely interested in predicting future actions and events of trained agents. The actions or events do not necessarily need to be unsafe. In the case of unsafe action or event prediction, our work allows for preemptive interruption of the deployed agent, which can be used as a last resort in addition to the above safety RL works.

Opponent Modelling in Multi-agent Setting: In a multi-agent setting, modeling the opponent's behavior may be beneficial in both competitive and cooperative scenarios. He et al. [22] use the opponent's inner state to better predict Q-values in a multi-agent setting. Foerster et al. [23] update an agent's policy while accounting for its effects on other agents, and Raileanu et al. [24] predict the

other agent's actions based on the same network that outputs the agent's own action. In contrast to these works, our research involves predicting agent actions multiple steps ahead and does not involve a multi-agent setting or learning a policy.

Predictability for Human-Agent Interaction: Recent research has highlighted the importance of predictability in enhancing human-agent interaction and collaboration. The agents in these studies are not necessarily RL agents but are often hardcoded to follow certain rules. Daronnat et al. [3] demonstrated that higher predictability in agent behavior facilitates better human-agent interaction and collaboration, particularly in real-time scenarios. Dragan et al. [2] found that legible motion, which makes an agent's intent clear, leads to more fluent human-robot collaboration. Kandul et al. [25] found that humans are better at predicting human performance than agent performance, raising concerns about human control over agents in high-stakes environments. Finally, Ahrndt et al. [4] discussed the significance of mutual predictability in human-agent teamwork. These works support the motivation that predictability of agents is an important concern for human-agent interaction.

7 Conclusion

In this paper, we investigated the predictability of future actions and events for different types of RL agents. We proposed and evaluated two approaches for prediction: the inner state approach and the simulation-based approach. The simulation-based approach performs well with an accurate world model but is less robust when the world model quality is compromised. Conversely, the performance of the inner state approach depends on the type of inner states and the agents. Internal plans of explicit planning agents are particularly useful compared to other types of inner states. These findings highlight the importance of leveraging auxiliary information to predict future actions and events. Enhanced predictability could lead to more reliable and safer deployment of RL agents in critical real-world applications such as autonomous driving, robotics, and healthcare, where understanding and anticipating agent behavior is important for safety and effective human-agent interaction.

Future research directions include extending our analysis to more diverse environments and RL algorithms, exploring safety mechanisms to modify agent behavior based on action prediction, and developing RL algorithms that are both predictable and high-performing.

Limitation

The paper only evaluates the proposed approaches in a limited set of environments. Including additional environments would provide a better understanding of agent predictability, but this requires finding or designing new benchmark environments with diverse states. Additionally, the paper focuses on only four different RL algorithms. Evaluating a broader range of RL algorithms could allow for better comparisons of their predictability.

Broader Impact Statement

This work involves predicting the actions and events of trained agents during deployment. It is important to consider the risk of false alarms, where the predictor forecasts that an agent is going to perform an unsafe action, but in fact, the agent would not be doing it. This may lead to improper responses (such as shutting down the agent) that are not warranted.

References

[1] Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *International Conference on Machine Learning*, pages 12004–12019. PMLR, 2022.

[2] Anca D Dragan, Shira Bauman, Jodi Forlizzi, and Siddhartha S Srinivasa. Effects of robot motion on human-robot collaboration. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, pages 51–58, 2015.

[3] Sylvain Daronnat, Leif Azzopardi, Martin Halvey, and Mateusz Dubiel. Impact of agent reliability and predictability on trust in real time human-agent collaboration. In *Proceedings of the 8th International Conference on Human-Agent Interaction*, pages 131–139, 2020.

[4] Sebastian Ahrndt, Johannes Fähndrich, and Sahin Albayrak. Human-agent teamwork: what is predictability, why is it important? In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 284–286, 2016.

[5] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[6] Stephen Chung, Ivan Anokhin, and David Krueger. Thinker: Learning to plan and act. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[8] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[9] Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sébastien Racanière, Théophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, et al. An investigation of model-free planning. In *International conference on machine learning*, pages 2464–2473. PMLR, 2019.

[10] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[11] Thomas M Moerland, Joost Broekens, Aske Plaat, Catholijn M Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[12] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768, 2011.

[13] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[16] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[17] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[18] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

[19] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3387–3395, 2019.

[20] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6 (3):4915–4922, 2021.

[21] Garrett Thomas, Yuping Luo, and Tengyu Ma. Safe reinforcement learning by imagining the near future. *Advances in Neural Information Processing Systems*, 34:13859–13869, 2021.

[22] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813. PMLR, 2016.

[23] Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

[24] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pages 4257–4266. PMLR, 2018.

[25] Serhiy Kandul, Vincent Micheli, Juliane Beck, Thomas Burri, François Fleuret, Markus Kneer, and Markus Christen. Human control redressed: Comparing ai and human predictability in a real-effort task. *Computers in Human Behavior Reports*, 10:100290, 2023.

A Agent Details

In this section, we describe the details of how we trained the four types of agents discussed in the paper. Most of them follow the procedure outlined in the original paper:

1. MuZero: We use the same agent configuration as in the original paper, except for the world model, where we adopt the architecture and training method proposed in Thinker. This is to ensure that the results are not affected by the choice of the world model. We conducted 100 simulations for each search[3].
2. Thinker: We use the same default agent as described in the original paper.
3. DRC: We use the DRC(3,3) described in the original paper.
4. IMPALA: We use the large architecture but omit the LSTM component described in the original paper.

All the hyperparameters are consistent with those in the original paper, and the agents are all trained using 25 million transitions. For each RL algorithm in the default model case, we train three separate agents with different seeds. For the model ablation case, we train only one agent due to computational cost.

World Model: The world model utilizes the *dual network* architecture proposed in Thinker, as it enables the prediction of raw states, values, and policies, allowing its use in both simulation-based approaches and planning in MuZero and Thinker. We also found that using the dual network results in better performance in the environment than the original network proposed in MuZero, likely due to the addition of learning signals from predicting the raw state. In the small model ablation case, we reduced all channel sizes in the RNN block from the default 128 to 32.

We follow the training procedure discussed in Thinker to train the world model, except that we added an additional loss based on L2-distance between the predicted raw state and the true raw state. This ensures that the world model focuses on all features of the raw states, not just those relevant to rewards. Consequently, non-reward-affecting features, such as the blue location, can still be encoded and predicted by the world model. We found that the addition of this loss does not negatively impact the agent's performance in the environment. It should be noted that we do not assume we have knowledge about the event g when training the world model or the agent. We train a separate world model following the same training procedure for each agent.

Figure 5 shows examples of model outputs for both the default setting and the model ablation setting. In the default case, the model predicts the states accurately, albeit with slight blurring. In the small model case, the agent erroneously pushes the box across the wall, which should not be allowed, and the blue location is missing, likely due to the model's limited capacity preventing it from fitting into the representation. In the stochastic case, the agent gradually fades due to the uncertainty of its position. Lastly, in the POMDP case, the agent is completely missing, attributable to the difficulty of ascertaining the agent's true position. These three model ablation cases thus showcase the different failure modes of the model.

Learning Curve: The learning curves of the agents in both the default setting and the model ablation setting can be found in Figure 6. We observe that in the default setting, the Thinker agent performs the best, with results closely replicating those of the original paper. The MuZero agent here outperforms the MuZero agent in the Thinker's paper due to the use of a dual network as the world model. In the small model setting, the performance of the DRC and baseline agents is similar to that in the default case, as the world model is not used in the policy. In the POMDP case, all agents perform poorly, likely because they cannot be certain of their own location, making the problem too challenging.

B Predictor Details

Predictor architecture : The raw states and predicted states (if they exist) are processed by a separate convolutional encoder with the same architecture. The encoder shares the same architecture as follows:

[3]We count each node traversal as one simulation, as opposed to one new node expansion

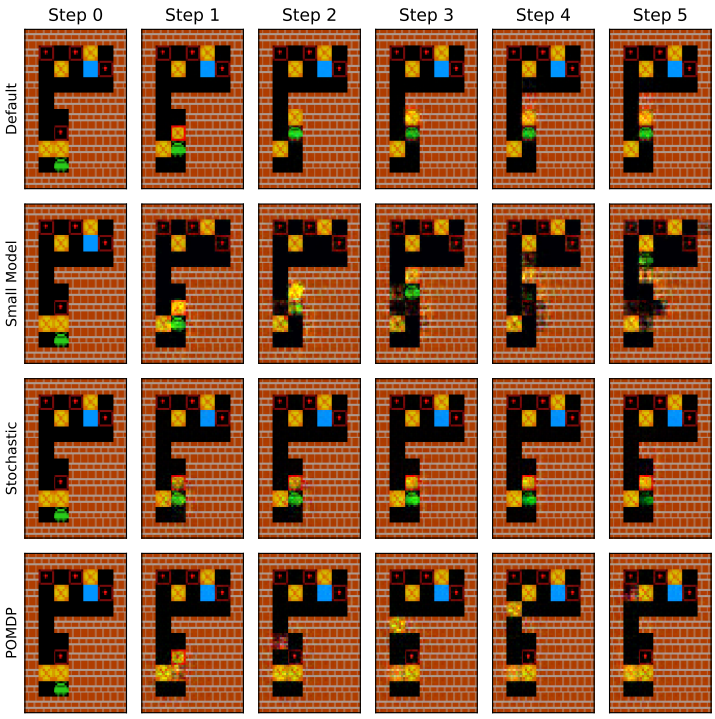

Figure 5: The predicted states output by the trained world model, where the starting state is shown in the leftmost column and the input action is five consecutive UP actions.

- Convolution with 64 output channels and stride 2, followed by a ReLu activation.
- 1 residual blocks, each with 64 output channels.
- Convolution with 128 output channels and stride 2, followed by a ReLu activation.
- 1 residual blocks, each with 128 output channels.
- Average pooling operation with stride 2.
- 1 residual blocks, each with 128 output channels.
- Average pooling operation with stride 2.

All convolutions use a kernel size of 3. The resulting output shape is (128, 6, 6). The output is then flattened and passed to a linear layer with an output size of 128. The encoded state is then concatenated with the action selected on that state to form an embedding. For the inner state approach of Thinker, we also concatenate the tree representation [6] to the embedding.

For the simulation-based approach applied to all agents and the inner state approach used by explicit planning agents, the embedding of the current state combined with the rollouts forms a sequence of embeddings of size 1 + total rollout length. For the inner state approach on the DRC agent, we encode the hidden state of each internal tick using the same convolutional encoder mentioned above, but without average pooling. Since there are four internal ticks ($t = 0, 1, 2, 3$) in DRC(3,3), they form a sequence of embeddings of size four. This sequence is concatenated with the current state embedding to form a sequence of five embeddings. Similarly, for the inner state approach on the IMPALA agent, we encode the hidden layer using the same convolutional encoder but also without average pooling. This is concatenated with the current state embedding to form a sequence of two embeddings. Note that the embedding of the current state is always positioned at the first slot in the sequence.

In all cases, the sequence of embeddings passes through a three-layer Transformer encoder with a dimension of 512. The output from the Transformer encoder at the first token is then passed to a linear layer, which predicts the required probabilities using either softmax or sigmoid output units.

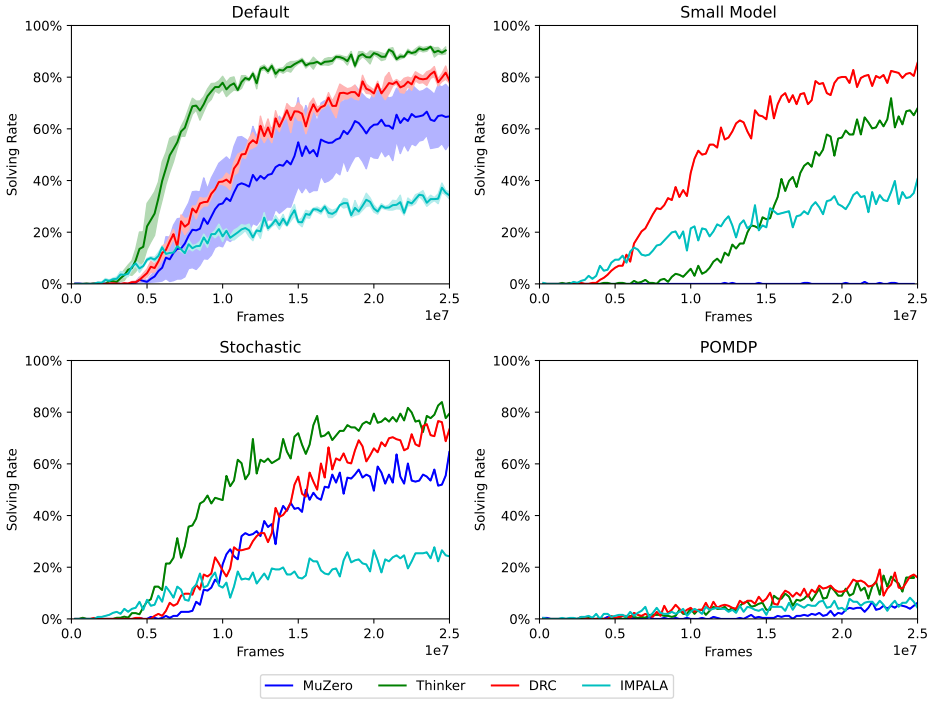

Figure 6: Running average solving rate over the last 200 episodes in Sokoban in both the default setting and the model ablation settings. For the default case, the shaded area represents two standard errors across 3 seeds.

Training: We generate 50,000 training samples, 10,000 evaluation samples, and 10,000 testing samples using the trained agents. We perform stochastic gradient descent on the cross-entropy loss to train both the action predictors and incident predictors. We utilize a batch size of 128 and an Adam optimizer with a learning rate of 0.0001. Training is halted when the validation loss fails to improve for 10 consecutive steps. For each agent with a unique seed, we train three independent predictors. As we have three separately trained agents for the default model case, this leads to a total of 9 runs.

C Experiment Details

Figure 7 shows the final accuracy of action prediction and the F1 score of incident prediction for the model ablation settings. The change in performance shown in Figure 4 is computed as the difference between the performance in the default model case and the performance shown here in Figure 7.

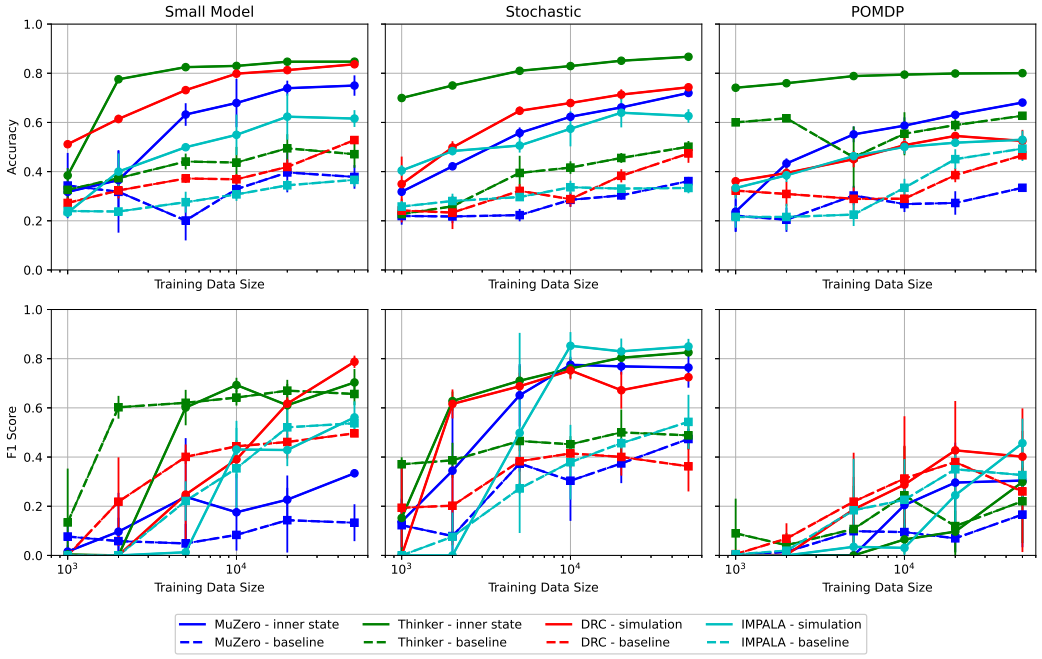

Figure 7: The final accuracy of action prediction and F1 score of incident prediction for the world model ablation settings. The error bar represents two standard errors across 3 seeds.

Computational Resources: Each agent is trained using a single A100 GPU, with training time varying by algorithm. MuZero, Thinker, DRC, and IMPALA take approximately 7, 3, 2, and 1 days, respectively, to complete training. The world model is trained concurrently with the agent on the same GPU. For training the predictors, we also use a single A100 GPU, and it takes about 2 days to complete training across all auxiliary information settings for a single agent and a single seed.

Code: The code used for these experiments is available at `https://github.com/stephen-chung-mh/predict_action` and is based on the public code released in Thinker [6].

D Ablation on Inner State Approach

We consider alternative choices for inner states and repeat the experiment shown in Figure 2. The following inner states are considered:

1. MuZero: We considered using the top 3 rollouts ranked by visit counts against only the top rollouts (the default case).
2. DRC: We considered using the hidden state at all ticks (the default case) against only the hidden state at the last tick.
3. IMPALA: We considered using the output of all three residual blocks against only the last residual block (the default case).

The results can be found in Fig 8. We observe that the results are similar with different chosen inner states, except that (i) using top 3 rollouts in MuZero leads to slightly lower event prediction accuracy, possibly because the top rollout is sufficient to make the prediction, and (ii) using all residual blocks in IMPALA gives slightly better performance in event prediction, likely because lower residual blocks still encode the blue location that is helpful for predicting the chosen event.

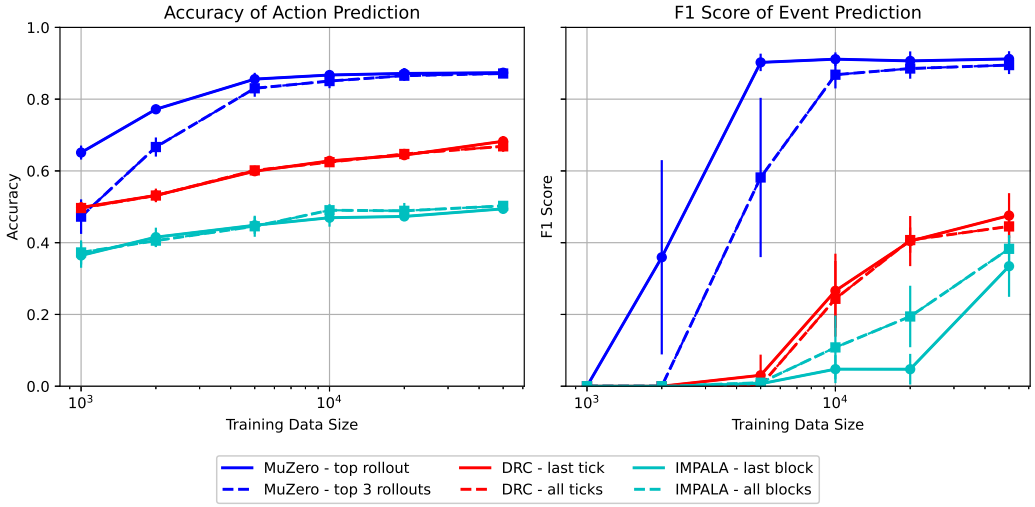

Figure 8: Ablation experiments on the chosen inner state. The error bars represent two standard errors across 9 seeds.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims are supported by experiments in Section 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 7.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Experiment details are provided in both the main paper and the appendix, and the code is released.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is provided in the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experiment details can be found in the appendix and the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in the appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: 1-sigma error bars are included in the figure. The train/test split is explained in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The details can be found in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section 7.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing assets are properly cited in the appendix.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The codes are well documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.