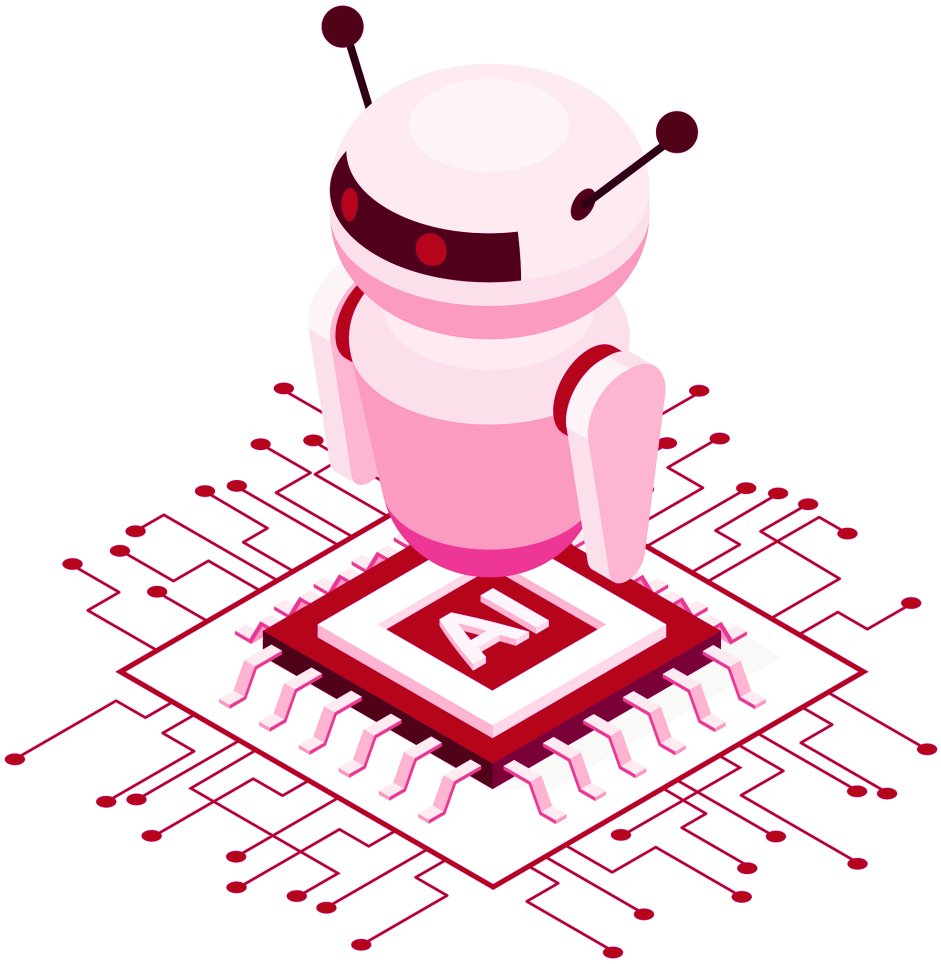

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 10:

Exploration in Deep Reinforcement Learning

By:

Asemaneh Nafef

400105285

RIML

Deep RL Course

Spring 2025

Contents

Grading

The grading will be based on the following criteria, with a total of 290 points:

Task	Points
Task 1: Bootstrap DQN Variants	100
Task 2: Random Network Distillation (RND)	100
Clarity and Quality of Code	5
Clarity and Quality of Report	5
Bonus 1	80

1 Task 1: Bootstrap DQN Variants

- The complete guidelines for implementing the Bootstrap DQN algorithm, including the RPF and BIV variants, are provided in the Jupyter notebook. You will find detailed instructions on how to set up the environment, implement the algorithms, and evaluate their performance.
- Make sure to read Guidelines section in the notebook carefully.

2 Task 2: Random Network Distillation (RND)

- You will implement the missing core components of Random Network Distillation (RND) combined with a Proximal Policy Optimization (PPO) agent inside the MiniGrid environment.
- **TODO:** You must complete the following parts:

File	TODO Description
`Core/model.py`	Implement the architecture of `TargetModel` and `PredictorModel`.
`Core/model.py`	Implement `_init_weights()` method for proper initialization.
`Core/ppo_rnd_agent.py`	Implement `calculate_int_rewards()` to compute intrinsic rewards.
`Core/ppo_rnd_agent.py`	Implement `calculate_rnd_loss()` to compute predictor training loss.

Table 1: Summary of required TODO implementations

- Questions:
  1. What is the intuition behind Random Network Distillation (RND)? Why does a prediction error signal encourage better exploration?
     The core idea of *Random Network Distillation (RND)* is to use curiosity as a learning signal to encourage the agent to explore novel states. In RND, we have two neural networks: a **target network** and a **predictor network**. The target network is initialized with random weights and remains fixed throughout training, while the predictor network is trained to mimic the target network's output for a given input state.
     The intuition: When the agent visits a new, unfamiliar state, the predictor network will initially struggle to accurately predict the target network's output because it has not seen this state before. This results in a high prediction error. On the other hand, for states the agent has visited frequently, the predictor network becomes good at predicting the target output, which leads to a low prediction error.
     Why does this encourage exploration? The prediction error acts as an *intrinsic reward* that tells the agent, "This is something new and potentially interesting." The agent is intrinsically motivated to seek out states that produce high prediction errors because it is essentially rewarded for encountering novelty. Over time, as the agent explores, the predictor network learns these states and the prediction errors decrease, pushing the agent to keep looking for new, unexplored regions of the environment.
     In this way, the *curiosity-driven* intrinsic reward provided by the prediction error helps the agent avoid getting stuck in familiar or trivial behaviors, leading to more thorough exploration of the environment.
  2. Why is it beneficial to use both intrinsic and extrinsic returns in the PPO loss function?

Combining **intrinsic** and **extrinsic** returns in the PPO loss function creates a more balanced learning signal that leverages the strengths of both types of motivation.
Extrinsic rewards come directly from the environment and represent the primary task the agent is expected to solve, such as reaching a goal or maximizing score. However, in many environments, these rewards can be sparse, delayed, or deceptive, which may cause the agent to struggle in discovering successful strategies.
On the other hand, **intrinsic rewards**, such as those from Random Network Distillation (RND), encourage the agent to explore unfamiliar or novel states by rewarding it based on prediction errors or other curiosity-based signals. This helps the agent stay motivated to explore even when extrinsic rewards are rare or hard to find.
By combining both:

- The agent learns to explore efficiently (thanks to intrinsic rewards) instead of getting stuck in known regions.
- The agent remains focused on solving the main task (guided by extrinsic rewards).

In summary Using both rewards creates a more robust agent that can balance exploration and exploitation. Intrinsic rewards help the agent discover valuable states, while extrinsic rewards ensure it ultimately optimizes for the task's true objectives.

3. What happens when you increase the `predictor_proportion` (i.e., the proportion of masked features used in the RND loss)? Does it help or hurt learning?
   When we increase the `predictor_proportion`, we are essentially using a larger portion of the feature vector in the Random Network Distillation (RND) loss calculation. This means that the predictor network is asked to predict more of the target network's features at each step.
   Intuitively, this can have mixed effects:
   - **Potential Benefit:** If the predictor focuses on more features, the learning signal might become richer and the intrinsic reward could more accurately reflect novelty across different aspects of the state space.
   - **Potential Downside:** However, using too many features may make the prediction task overly complex or noisy, especially early in training when the predictor is not yet capable. This can lead to weaker or less stable intrinsic rewards, which may confuse the agent or slow down exploration.

   In summary: Increasing the `predictor_proportion` is a trade-off. A moderate increase might help the agent form a more complete understanding of novelty, but if the proportion is too high, it can hurt learning by making the prediction task harder and diluting the curiosity signal. Careful tuning is needed to find a balance.
4. Try training with `int_adv_coeff=0` (removing intrinsic motivation). How does the agent's behavior and reward change?
   When we set `int_adv_coeff=0`, we are essentially removing the intrinsic reward from the agent's learning process. This means the agent is no longer encouraged to explore novel states—it focuses only on maximizing extrinsic rewards provided directly by the environment.
   What typically happens:
   - The agent tends to exploit known rewarding strategies much earlier instead of exploring new areas of the environment.
   - If the environment has sparse rewards, the agent might struggle to find them at all, getting stuck in local behaviors with low overall performance.
   - The total reward collected by the agent often decreases compared to runs with intrinsic motivation, especially in environments where exploration is critical to success.

   as you can see in the fig. 1 when by removing `int_adv_coeff=0` the agent can not find the

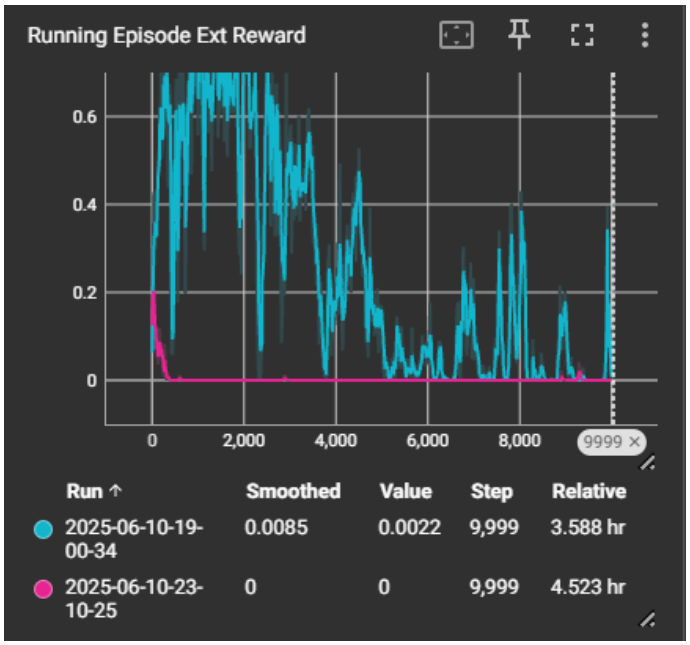

Figure 1: pink line refers to `int_adv_coeff=0` $= 0$

optimal policy.
and as you can see in the fig. 2 the agent can reach heigh ex reward (above 0.6) when the `int_adv_coeff=0` is not zero but by putting it to zero t can never give a reward above 0.4 In summary: Without intrinsic rewards, the agent becomes more short-sighted. It is less curious and explores less, which can significantly hurt learning in environments where discovering new states is key to solving the task.

5. Inspect the TensorBoard logs. During successful runs, how do intrinsic rewards evolve over time? Are they higher in early training?
   When we look at the TensorBoard logs, we usually observe that intrinsic rewards are significantly higher during the early stages of training. This makes sense because, at the beginning, almost all states are novel to the agent, so the prediction error (which drives the intrinsic reward) is large.
   As training progresses:
   - The agent repeatedly visits the same states and the predictor network becomes better at predicting the target network's outputs for these states.
   - The prediction error naturally decreases, leading to lower intrinsic rewards over time.
   - This reduction in intrinsic reward is actually desirable—it means the agent has sufficiently explored those states and should now focus more on exploiting the knowledge it has gained.

   intrinsic rewards are typically higher in early training because everything is unfamiliar to the agent. Over time, as the agent learns and familiar states become easier to predict, intrinsic rewards decrease, signaling that the agent has "satisfied its curiosity" in those parts of the environment.
   When we set `int_adv_coeff=0`, the intrinsic rewards are no longer used to guide the agent's behavior. However, the intrinsic rewards themselves are still being calculated based on the prediction error between the target and predictor networks.
   At the start of training, intrinsic rewards are usually high because most states are new and the predictor network struggles to predict the target network's outputs. Over time, though, something interesting happens:
   - Since the agent is no longer intrinsically motivated to explore, it tends to focus only on

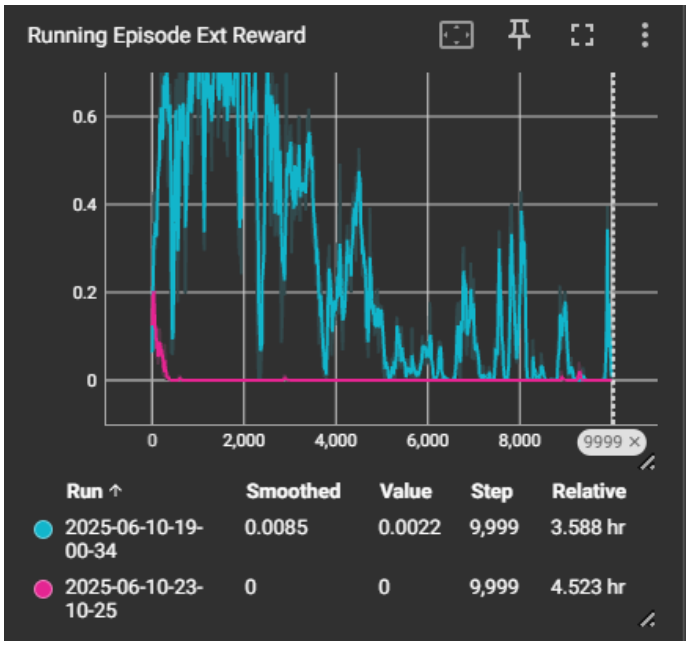

Figure 2: pink line refers to `int_adv_coeff`=0 $= 0$

maximizing extrinsic rewards and may not visit new or novel states.

- As a result, the predictor network may not have the chance to improve on less-visited areas of the state space.
- This can lead to the intrinsic rewards decreasing more slowly, or even staying relatively high, because the agent doesn't sufficiently explore to "solve" the prediction errors in novel states.

Without intrinsic motivation, the agent typically explores less, and intrinsic rewards may remain higher for longer because there are still many states that the agent has not learned to predict well.