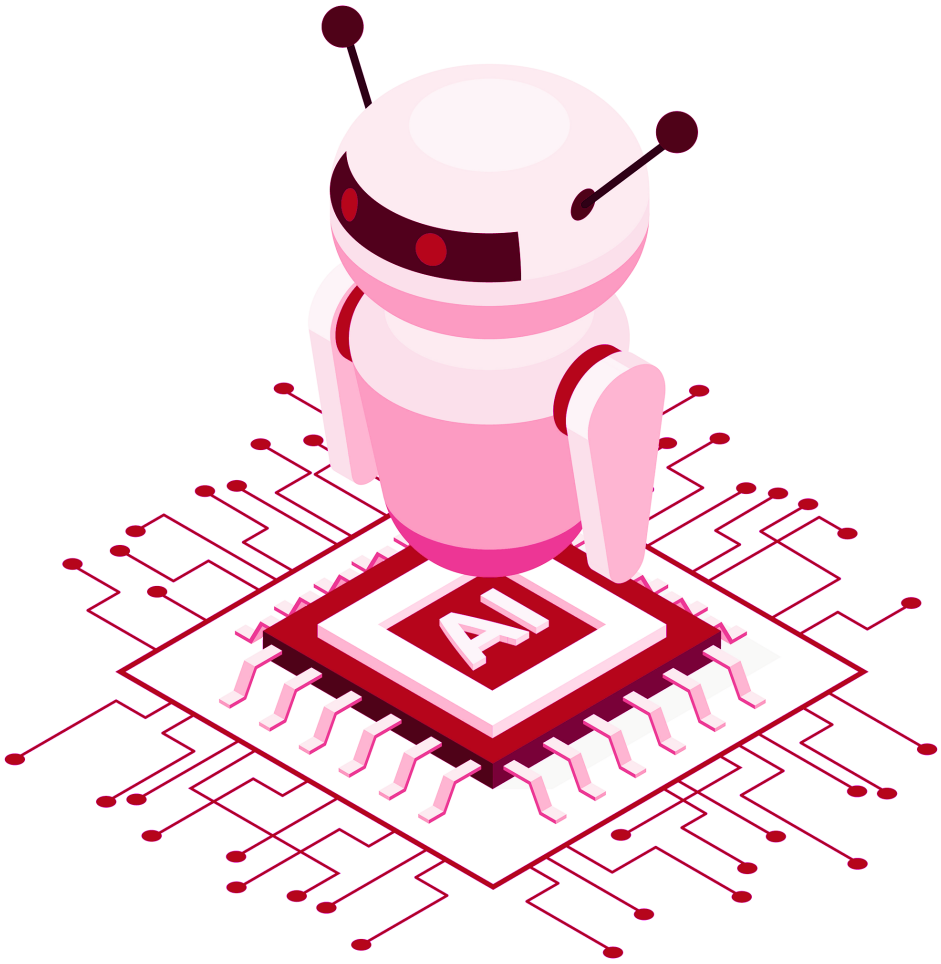

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Homework 10:

Exploration in Deep Reinforcement Learning

By:

[Full Name]

[Student Number]

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?

     Random Network Distillation (RND) uses two networks: a fixed random network and a trainable predictor network. The predictor learns to match the output of the fixed one for each state the agent visits. When the agent sees a new or rare state, the prediction error is high, which gives a higher reward. As the agent visits that state more often, the predictor gets better, and the reward goes down. This way, the agent is encouraged to explore new areas where its predictions are still bad, instead of staying in familiar places.
  2. Why is it beneficial to use both intrinsic and extrinsic returns in the PPO loss function?

     Using both intrinsic and extrinsic returns lets the agent explore and learn the env. Extrinsic returns come from the environment's real rewards, guiding the agent toward goal. Intrinsic returns like the RND prediction error reward novelty and new trajectories, pushing the agent into unfamiliar states. Relying only on extrinsic rewards can trap the agent in sub-optimal routines, while only intrinsic rewards lead to aimless exploration but combining them give the better exploration/exploitation balance.

  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 you increase predictor proportion, the predictor network is trained on more of the target's feature dimensions each update. That makes it learn faster and drives its prediction error down

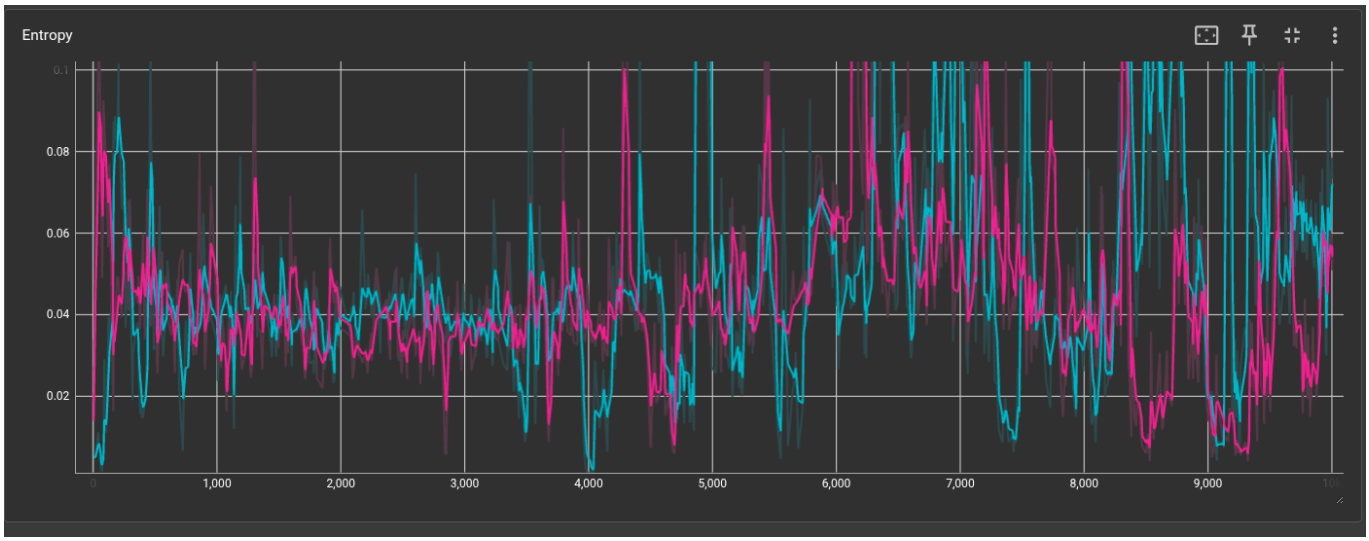

Figure 1: Entropy, Pink is int_adv_coeff=0 and blue is 1

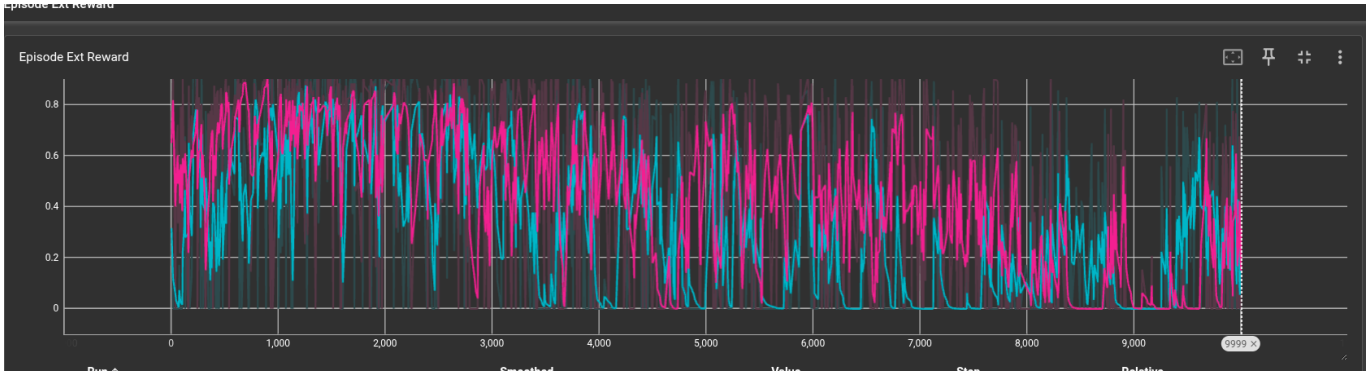

Figure 2: Extrinsic Reward, Pink is int_adv_coeff=0 and blue is 1

sooner, so the agent's curiosity bonus shrinks more quickly. As a result, the agent settles into exploiting known strategies earlier and explores less.

4. Try training with `int_adv_coeff=0` (removing intrinsic motivation). How does the agent's behavior and reward change?

   As expected, when we remove the intrinsic bonus (pink), the agent's entropy quickly collapses and it converges to a narrow, sub-optimal policy—its episode rewards spike early but then plateau at a lower level and remain unstable—whereas with the intrinsic reward enabled (blue), exploration stays high for longer, leading to more variable early returns but ultimately higher and more consistent extrinsic performance once the agent has sufficiently explored the environment.

5. Inspect the TensorBoard logs. During successful runs, how do intrinsic rewards evolve over time? Are they higher in early training?

   In successful runs, intrinsic rewards start off high, since the predictor network's error is large on new states, and then gradually go down as the predictor learns and the agent revisits familiar areas. You'll see a drop in the exploration bonus during the first few thousand steps, after which intrinsic rewards level off near zero, indicating that most states have been explored and

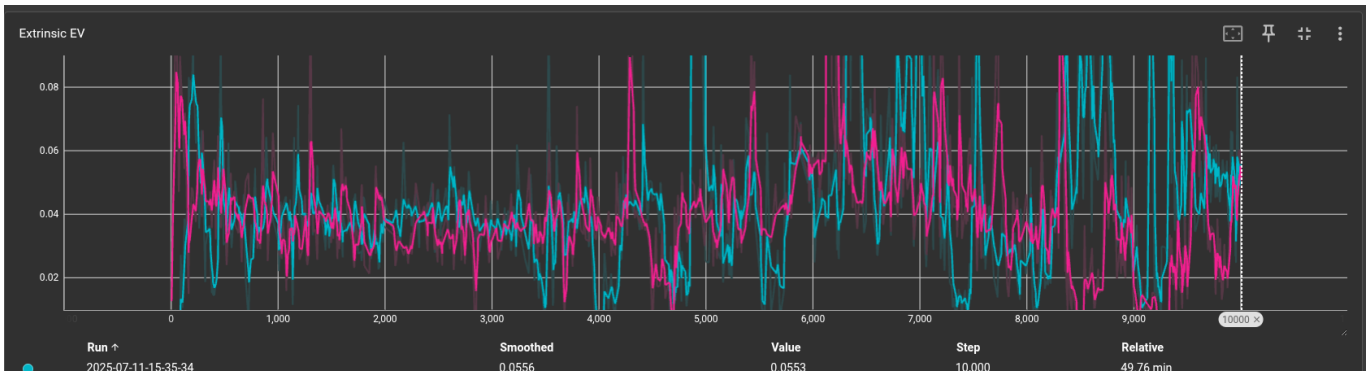

Figure 3: Extrinsic expected variance, Pink is int_adv_coeff=0 and blue is 1

the agent is shifting focus to maximizing extrinsic return.