Machine Learning Paradigms, Artificial Intelligence Degree, UAB

Machine Learning Paradigms Project

[Group 2] Joan Lafuente & Adrián García

School of Engineering

Index

1 Frogger

2 Ms. Pacman

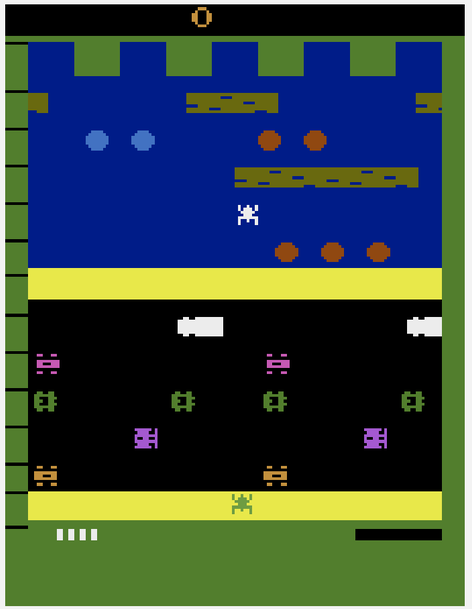
3 Pong

Frogger

Frogger Enviroment

You are a frog trying to make their way home. Cross a highway and a perilous river without being crushed or eaten.

Value	Meaning	Value	Meaning	Value	Meaning
`0`	`NOOP`	`1`	`UP`	`2`	`RIGHT`
`3`	`LEFT`	`4`	`DOWN`		

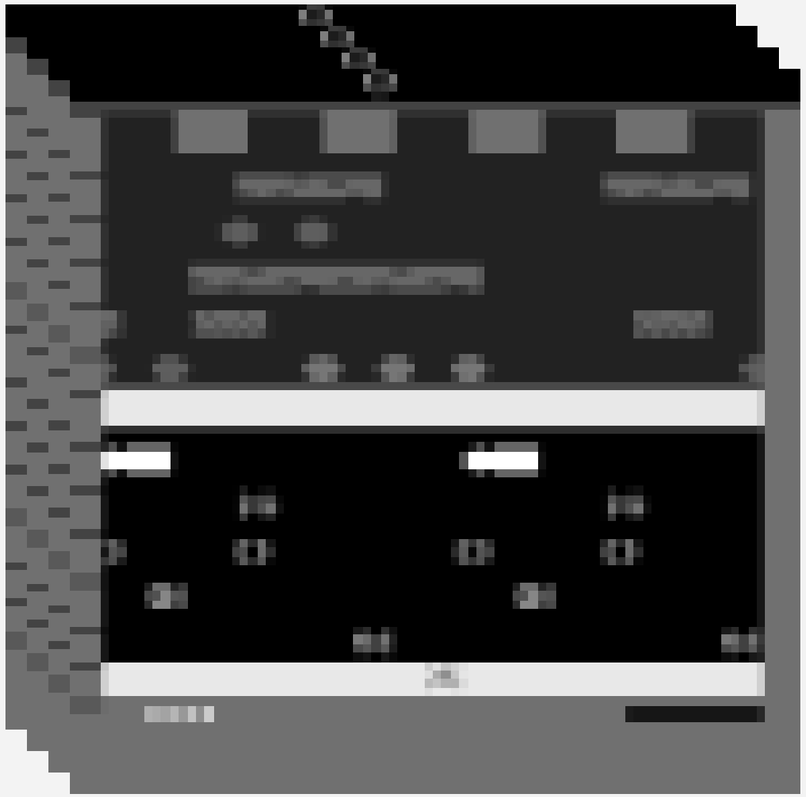


Frogger Enviroment: Preprocessing

Preprocessing steps:

- *Frame-skipping = 16*
- *Image resize to 84x84*
- *Observation stacking = 4*
- *Image Scaling in Range [0, 1]*

** Greyscale given by the game*



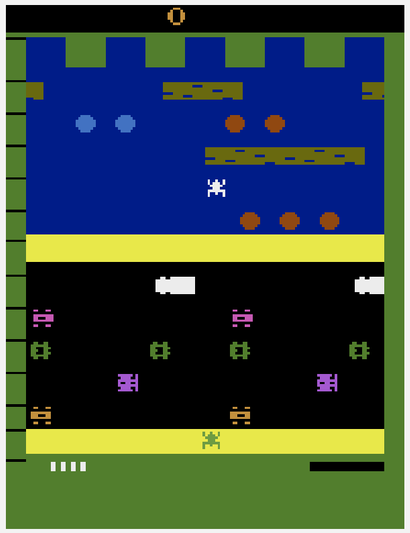
DQN + Few Extensions

Extensions:

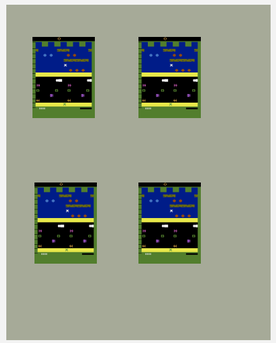
- ***Dueling DQN**: Uses a specialized Dueling Q Head to separate Q to an A (advantage) stream and a V stream.*
- ***Double DQN**: Utilises Double Q-learning to reduce overestimation by decomposing the max operation in the target into action selection and action evaluation.*
- ***Prioritized Experience Replay**: Replay in reinforcement learning where we more frequently replay transitions with high expected learning progress, as measured by the magnitude of their temporal-difference (TD) error.*

DQN + Few Extensions

Extensions explanation



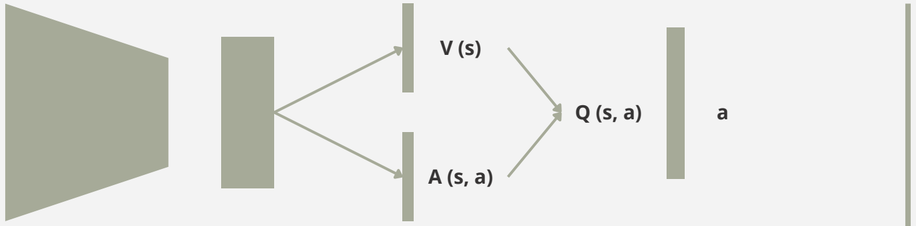
Prioritized Experience Replay



Based on TD-Error

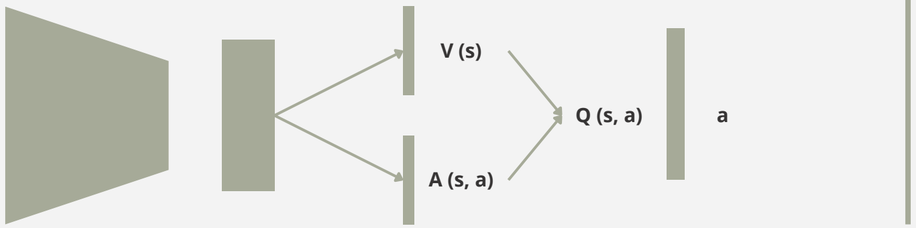
$$\delta = |Q_{\text{current}} - Q_{\text{target}}|.$$

Target Network ❄️



$$L = \mathbb{E}\left[w \cdot (Q_{\text{current}} - Q_{\text{target}})^2\right]$$

Online Network 🔥



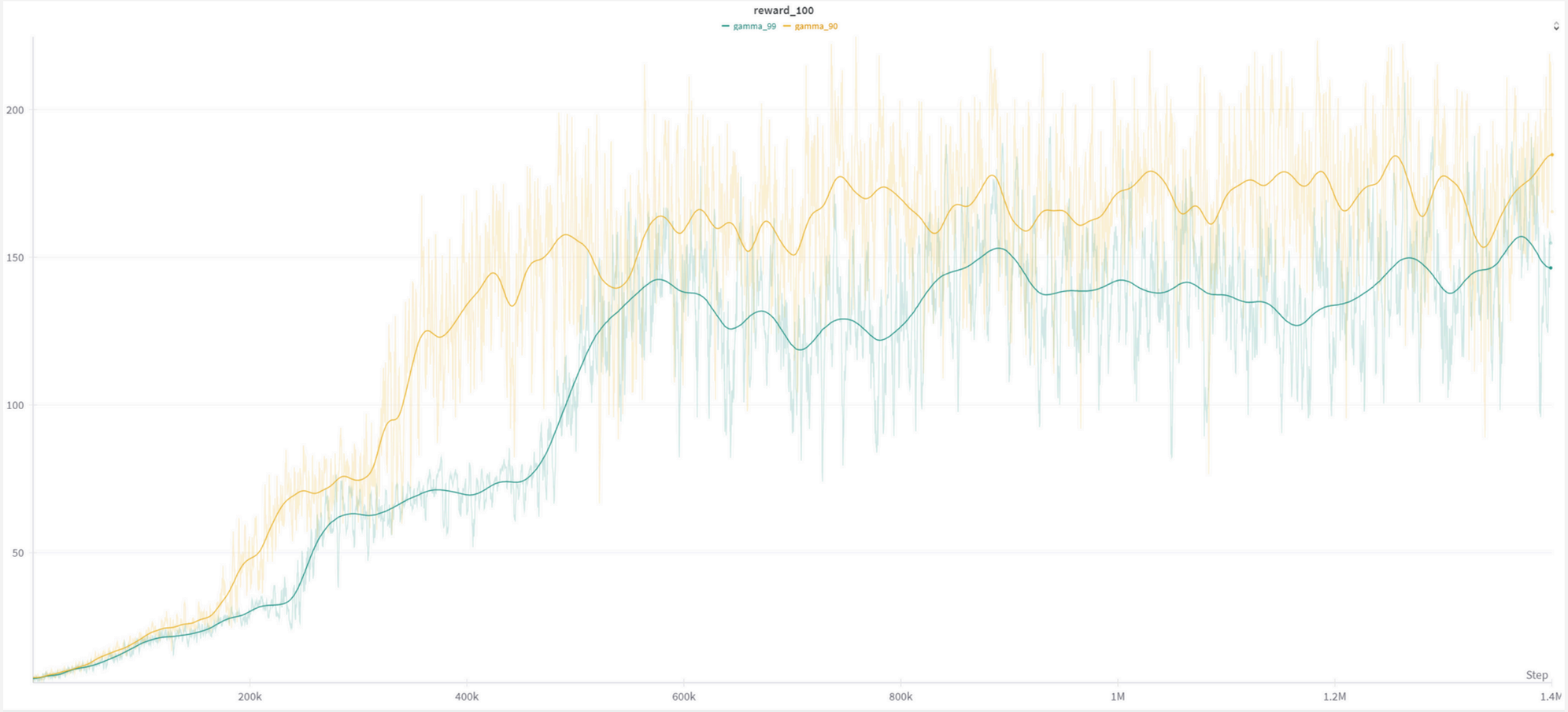
Periodic Target Synchronization

$$Q' \leftarrow Q$$

Deal with *Exploration vs Exploration*: **Epsilon- Greedy**

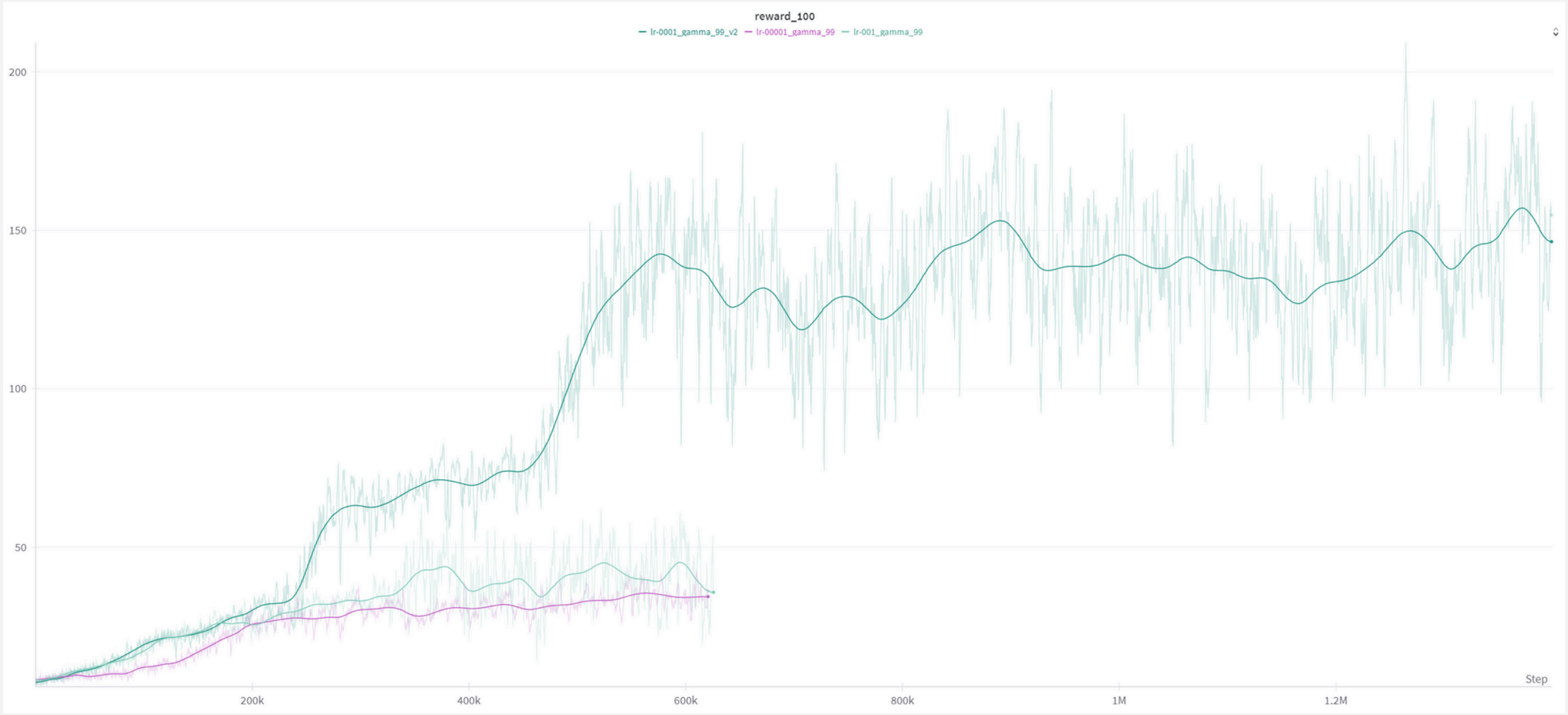
DQN + Few Extensions

Hyperparameter Search



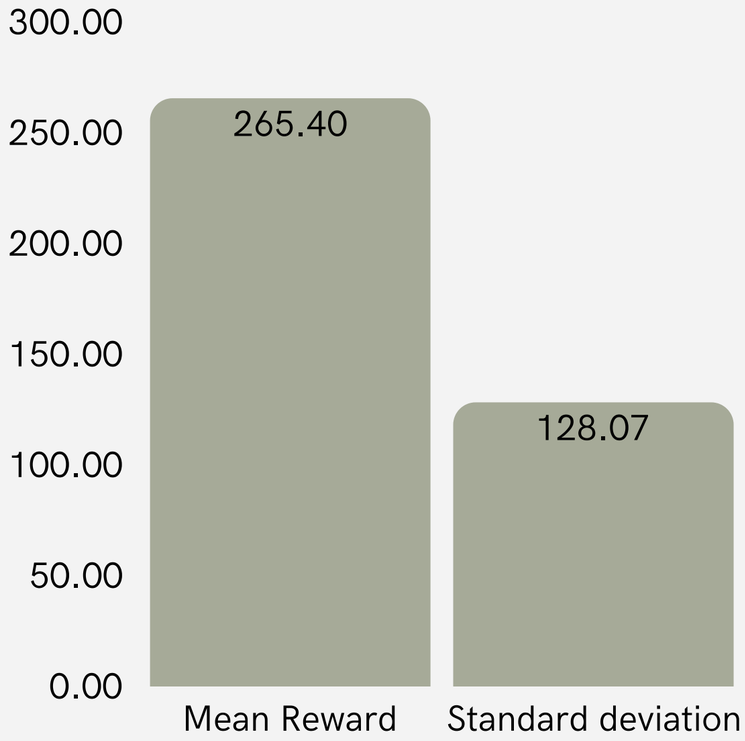
DQN + Few Extensions

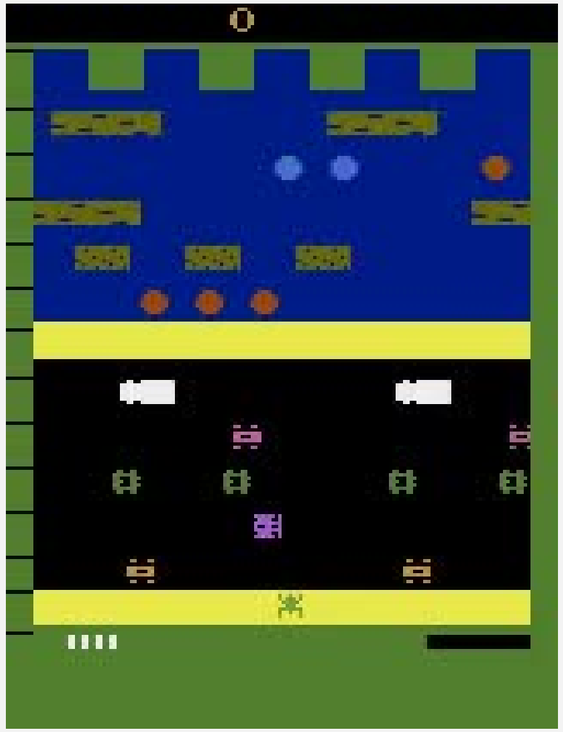
Hyperparameter Search



DQN + Few Extensions

Results





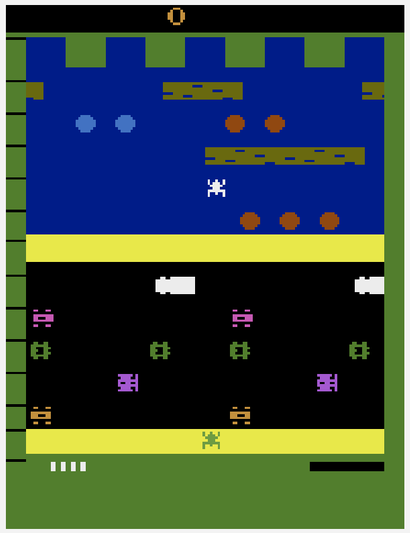
Rainbow DQN

Extensions:

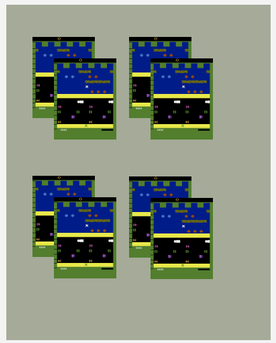
- ***Dueling DQN**: Uses a specialized Dueling Q Head to separate Q to an A (advantage) stream and a V stream.*
- ***Double DQN:** Utilises Double Q-learning to reduce overestimation by decomposing the max operation in the target into action selection and action evaluation.*
- ***N-Step Prioritized Experience Replay:** Prioritizes and samples experiences based on their importance, using multi-step returns for more efficient learning.*
- ***Noisy layers:** Linear layer with parametric noise added to the weights.*

Rainbow DQN

Extensions explanation



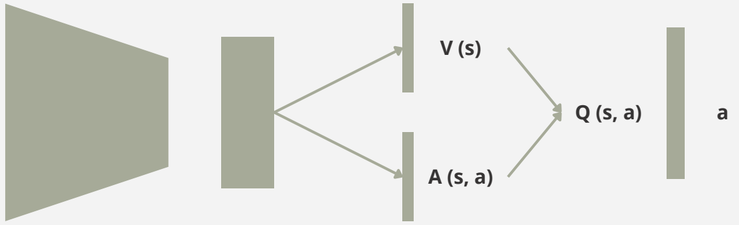
Prioritized Experience Replay



Based on TD-Error

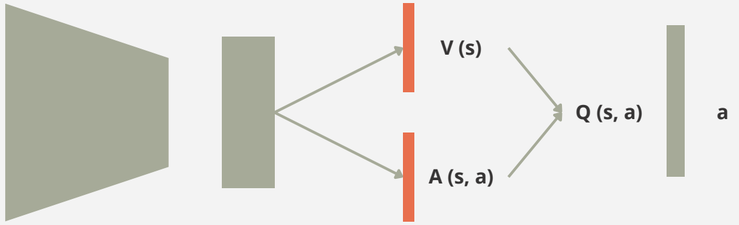
$$\delta = |Q_{\text{current}} - Q_{\text{target}}|.$$

Target Network ❄



$$L = \mathbb{E}\left[w \cdot (Q_{\text{current}} - Q_{\text{target}})^2\right] + \lambda \cdot \sum |\sigma_{\text{weights}}|$$

Online Network 🔥



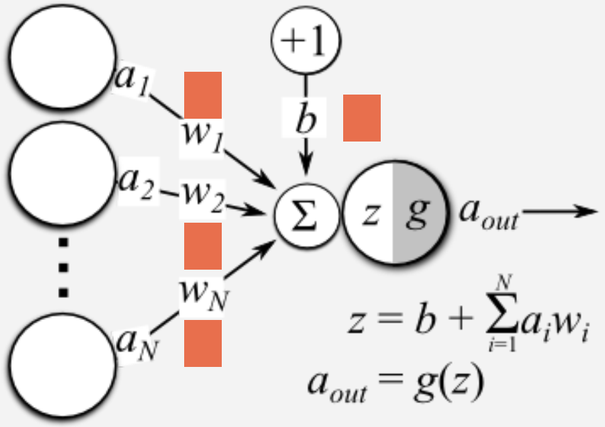
Periodic Target Synchronization

$$Q' \leftarrow Q$$

Deal with *Exploration vs Exploration*: * **Gaussian Noise Injection in Noisy Layers if Mean Reward Plateau Detection**

Rainbow DQN

Gaussian Noise Injection in Noisy Layers if Mean Reward Plateau Detection



- **Like a regular linear layer (matrix multiplication + bias) *but with added noise*.**
- **Explore by making predictions less predictable.**

$$W = \mu_W + \sigma_W \cdot \epsilon_W$$

$$b = \mu_b + \sigma_b \cdot \epsilon_b$$

- μ_W, μ_b: The main weights and biases (like in a regular layer).
- σ_W, σ_b: How much noise to add (scaling factor).
- ϵ_W, ϵ_b: The actual random noise (Gaussian samples).

$$\epsilon_w = \text{sign}(\epsilon) \cdot \sqrt{|\epsilon|}.$$

- **Mean weights & bias = *Initialized uniformly***
- **Std weights & bias = *Initialized with high variance for exploration***

Learnable parameters:

- ***Weight and bias scaling factors***

Rainbow DQN

Gaussian Noise Injection in Noisy Layers if Mean Reward Plateau Detection

Problem:

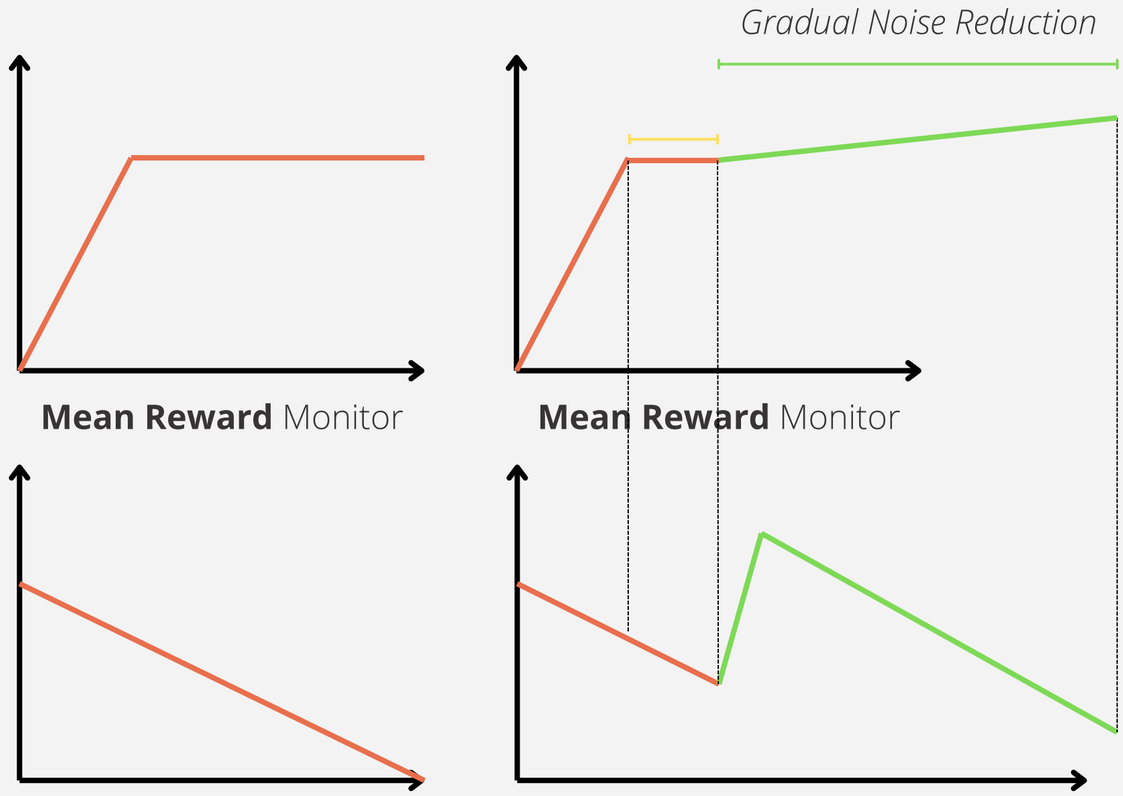
The model's shrinking weight and bias scaling factor (learnable parameters) can reduce exploration and prioritize exploration in later training stages, causing the model to become overconfident and missing *more optimal policies*.

Solution:

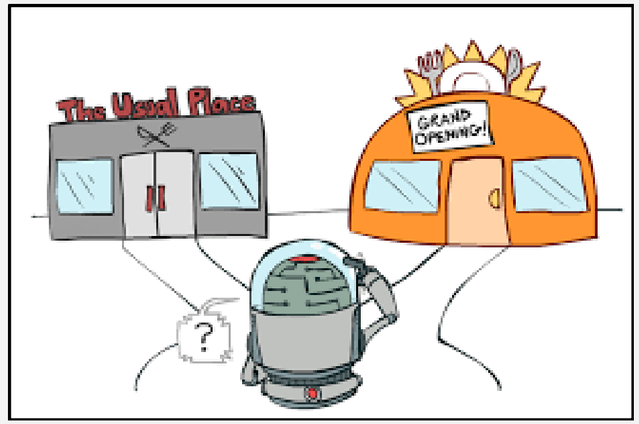
- **(*Regularization*): Add a *small penalty to the loss function* to prevent the noise scaling factors from shrinking too quickly. The penalty is proportional to the sum of the absolute values of all scaling parameters.**
- **Re-introduce or amplify exploration when the agent's performance plateaus, indicating that it's potentially stuck in a local optimum: *Dynamic Noise Injection Based on Plateau Detection***

Rainbow DQN

Gaussian Noise Injection in Noisy Layers if Mean Reward Plateau Detection



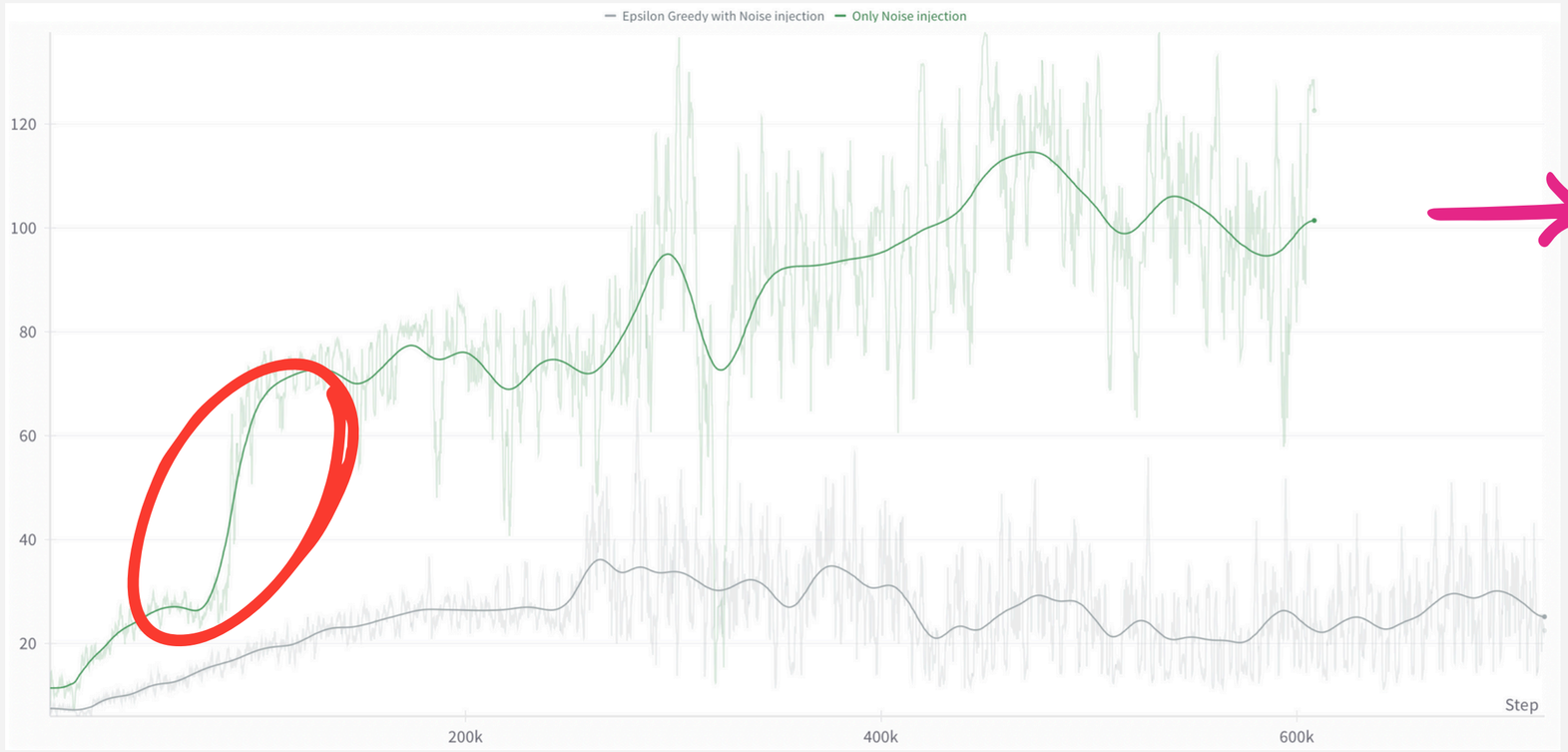
$$L = \mathbb{E}\left[w \cdot (Q_{\text{current}} - Q_{\text{target}})^2\right] + \lambda \cdot \sum |\sigma_{\text{weights}}|$$



By avoiding overconfidence: ***Dynamic Adaptive Exploration***

Rainbow DQN

Experiments



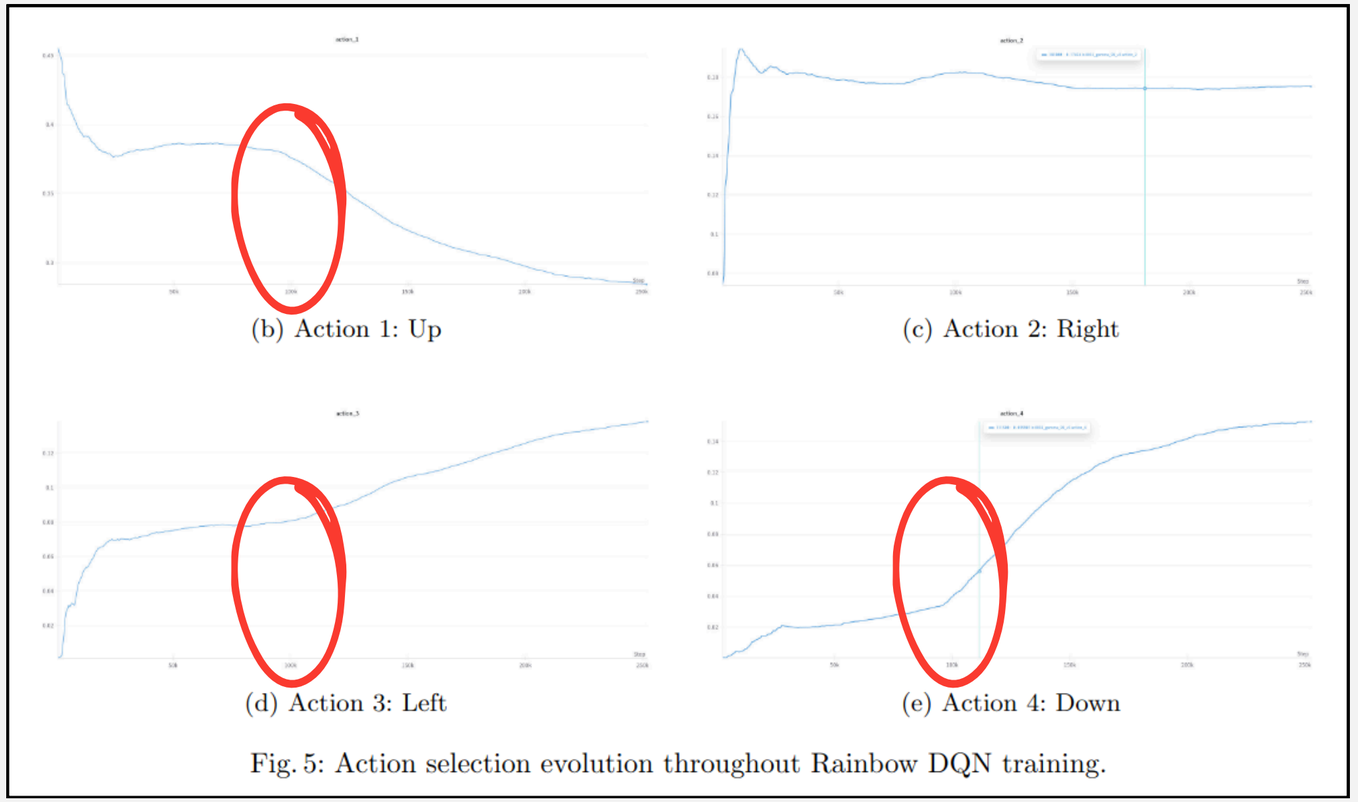
Rainbow DQN
with *Gaussian Noise Injection* approach

Rainbow DQN
with *Epsilon-greedy* approach

Rainbow DQN

Experiments

Change in Strategy with
Gaussian Noise Injection
approach



(b) Action 1: Up

(c) Action 2: Right

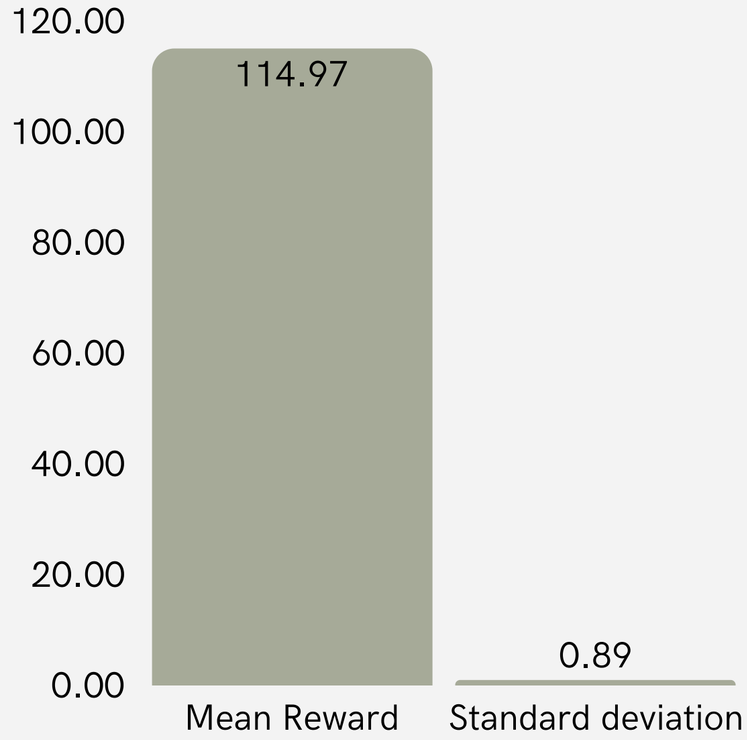
(d) Action 3: Left

(e) Action 4: Down

Fig. 5: Action selection evolution throughout Rainbow DQN training.

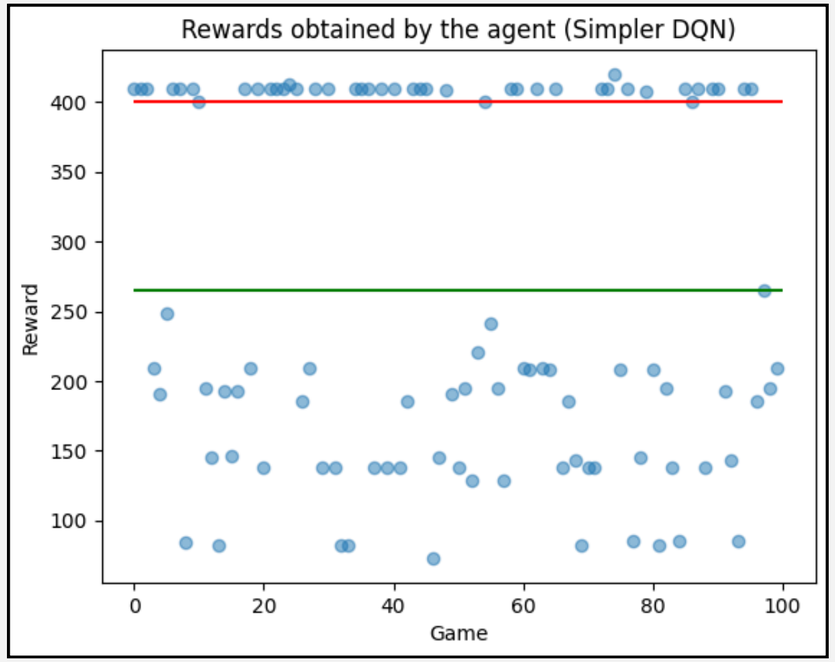
Rainbow DQN

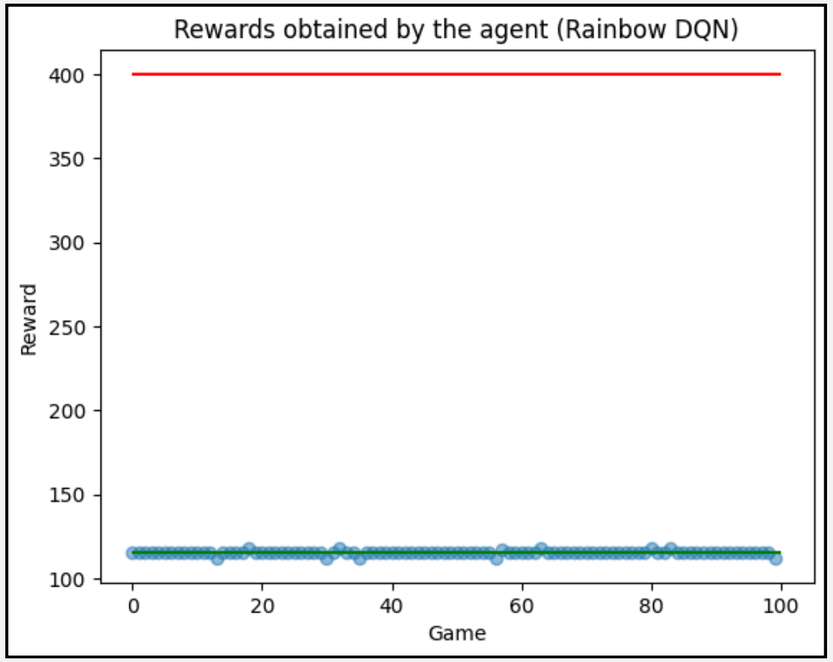
Results



Results

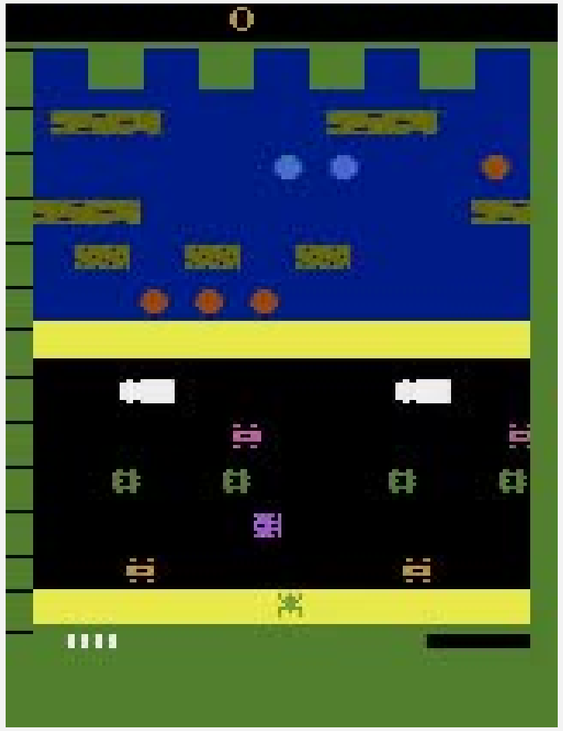
Frogger



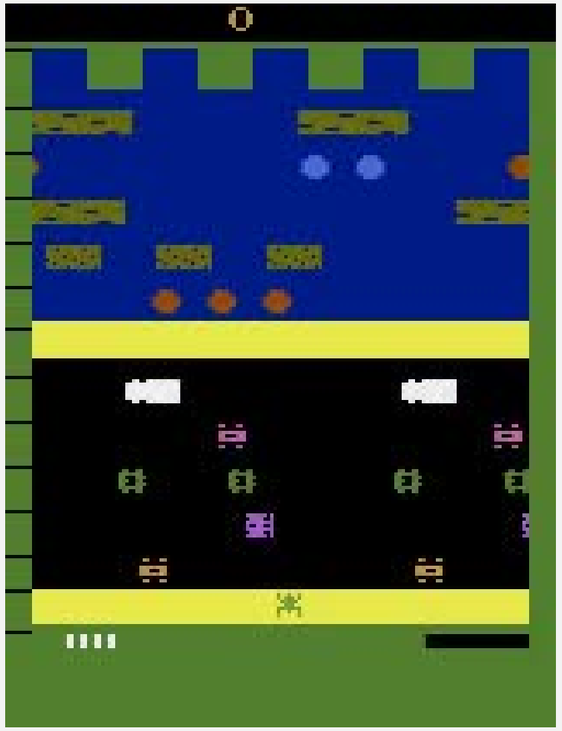


Results

Frogger



DQN + Extensions



Rainbow DQN

Conclusions

Frogger

Rainbow + Extensions

Best performance on Frogger environment

Rainbow DQN

Although the Rainbow exploration strategy seems to work in the early training stages further considerations can be taken into account:

- **Initialize the scaling factor to a higher initial value in Noisy Layers.**
- **A higher scaling factor in Gaussian Noise Injection in Noisy Layers would help escape local minima in later training dynamics. The *Trained model seems to have optimized exploration Frogger level 1.***

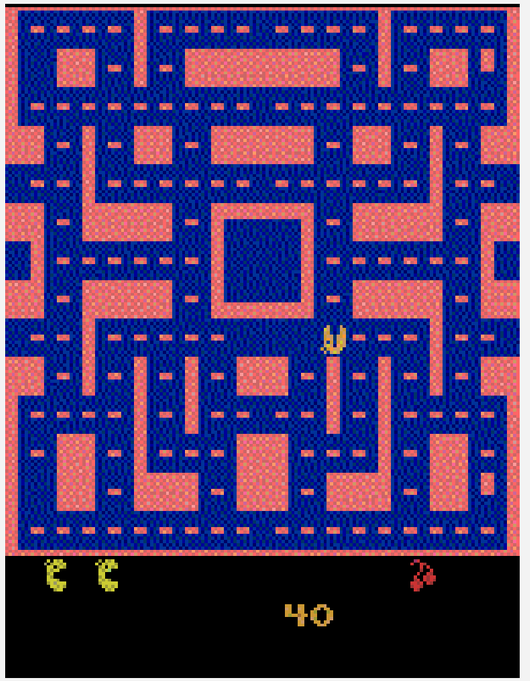
Other tried implementations: ***Teacher Student Distillation logic with Q-values (on target online & network), Reinforce***

Ms. Pacman

Ms.Pacman Enviroment

Your goal is to collect all of the pellets on the screen while avoiding the ghosts.

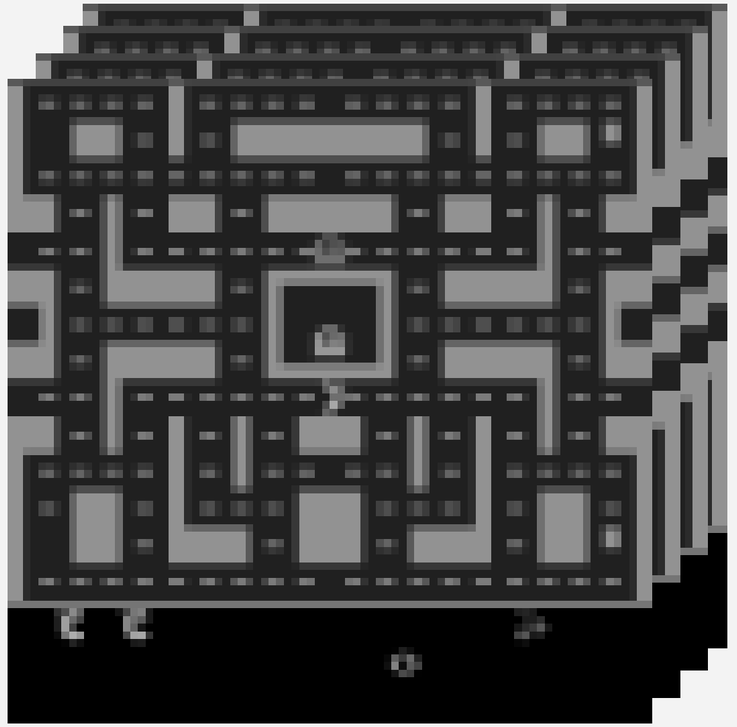
Value	Meaning	Value	Meaning	Value	Meaning
0	NOOP	1	UP	2	RIGHT
3	LEFT	4	DOWN	5	UPRIGHT
6	UPLEFT	7	DOWNRIGHT	8	DOWNLEFT



Ms. Pacman Enviroment: Preprocessing

Preprocessing steps:

- *Noop Reset*
- *Frame-skipping = 4*
- *End episode on life lost (During training)*
- *Resize to 84x84*
- *GrayScale*
- *Clip Rewards*
- *Observation stacking = 4*



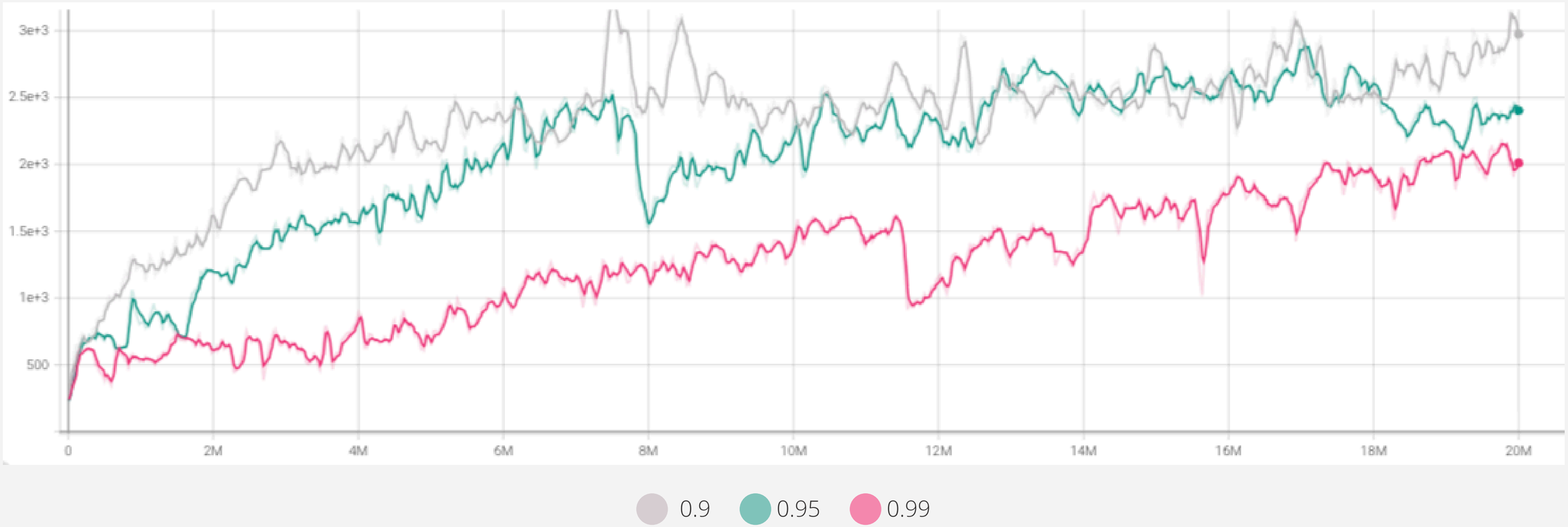
Proximal Policy Optimization

Reinforcement learning algorithm that optimizes policies by balancing exploration and exploitation with a clipped objective to ensure stable updates.

Proximal Policy Optimization

Hyperparameter Search

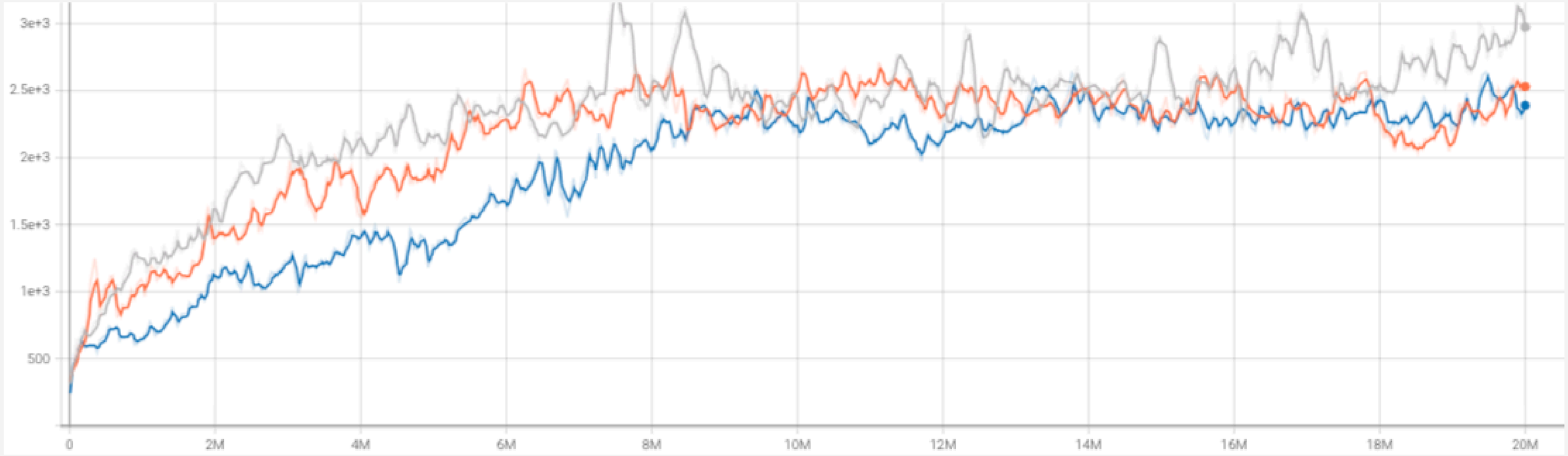
Discount factor effect:



Proximal Policy Optimization

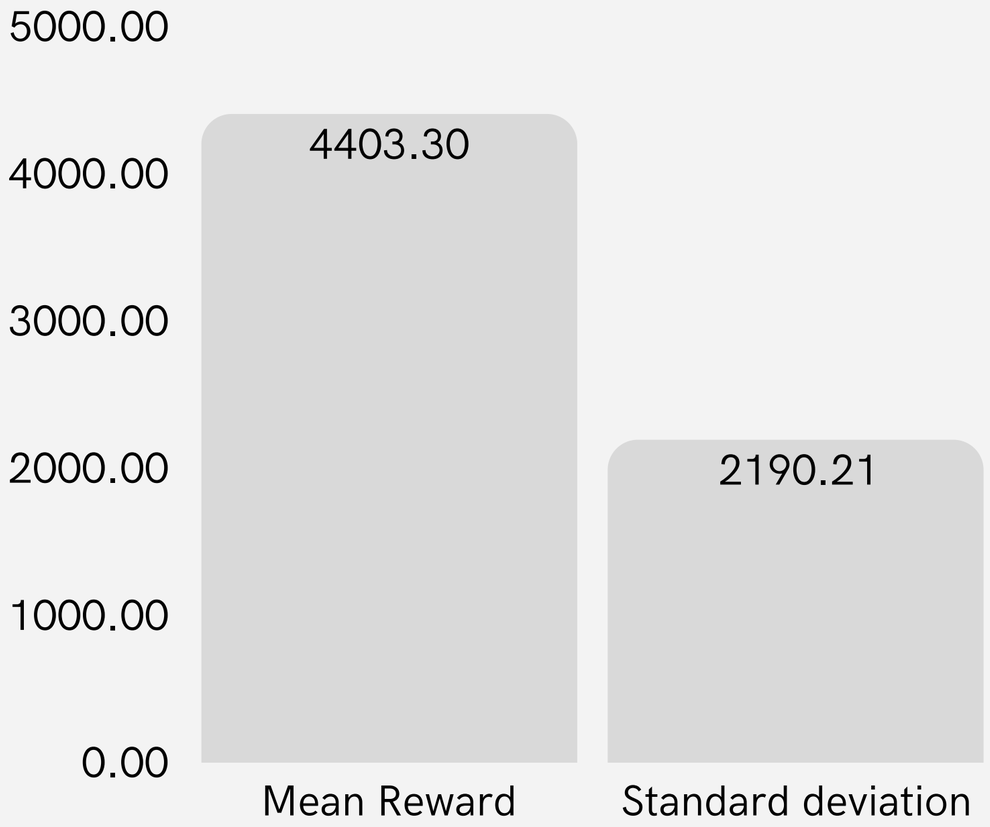
Hyperparameter Search

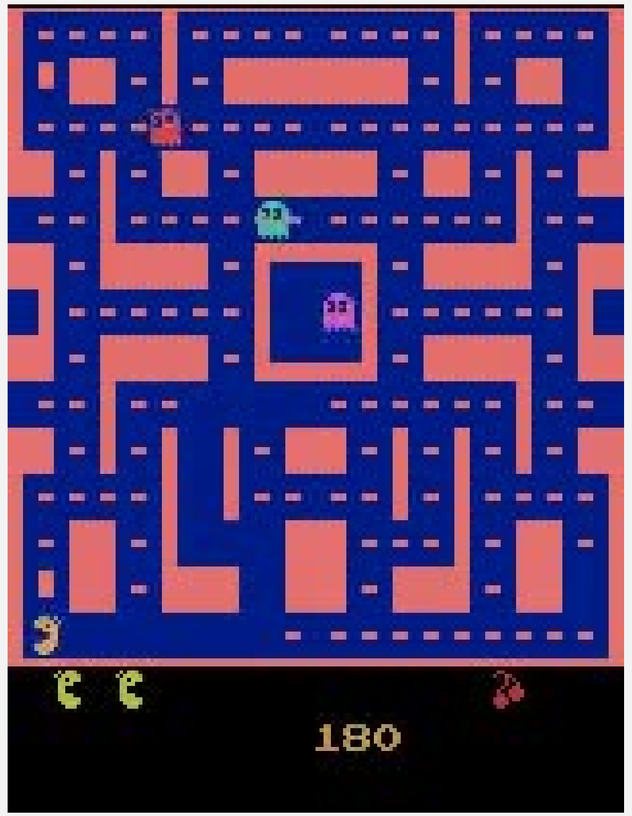
Learning rate effect:



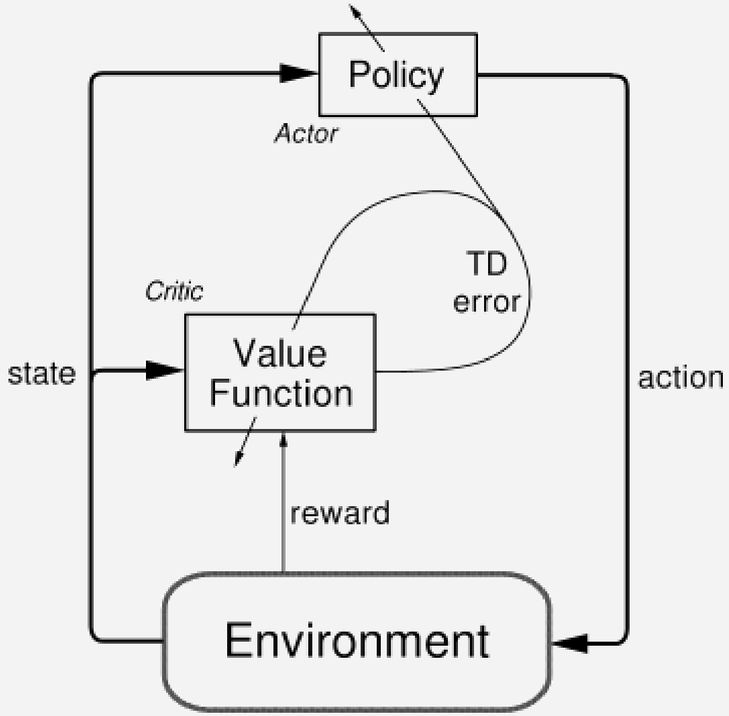
Proximal Policy Optimization

Results





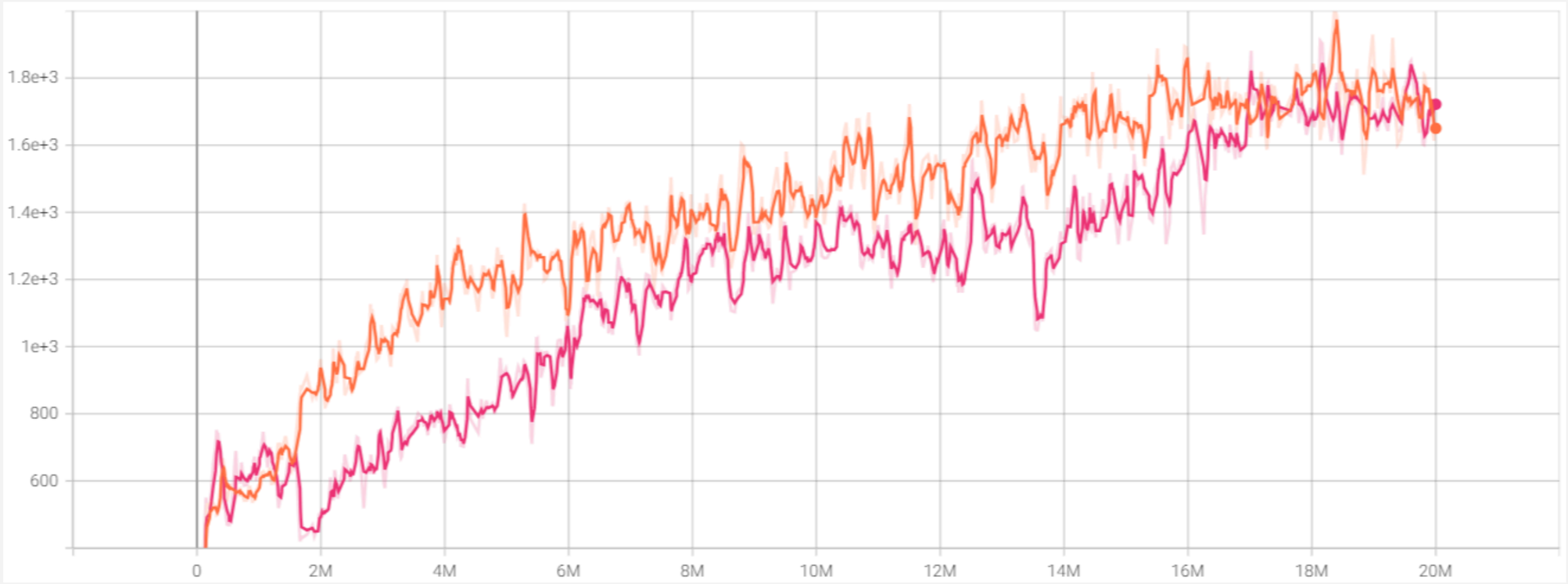
Advantage Actor Critic



Advantage Actor Critic

Hyperameter search

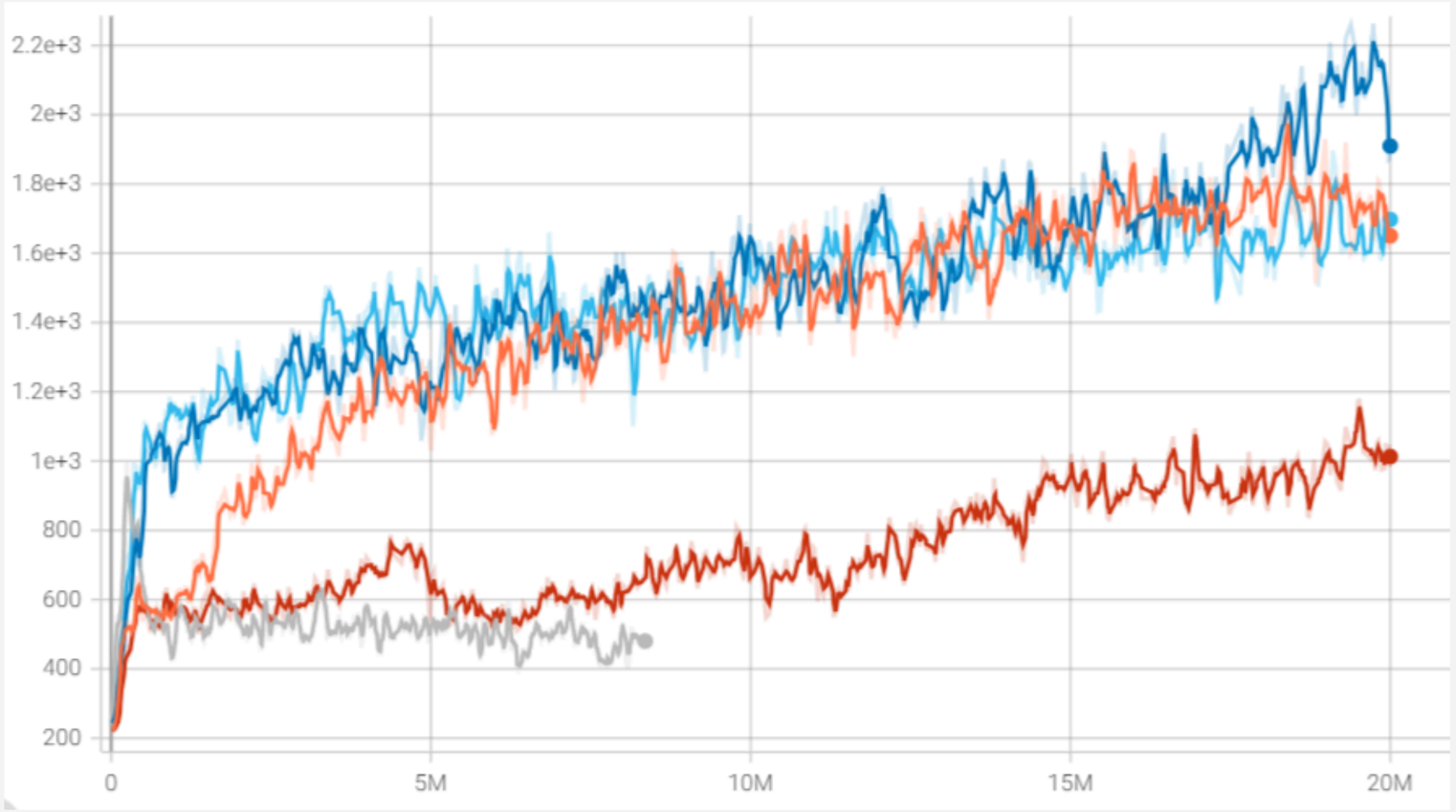
Discount factor effect:



Advantage Actor Critic

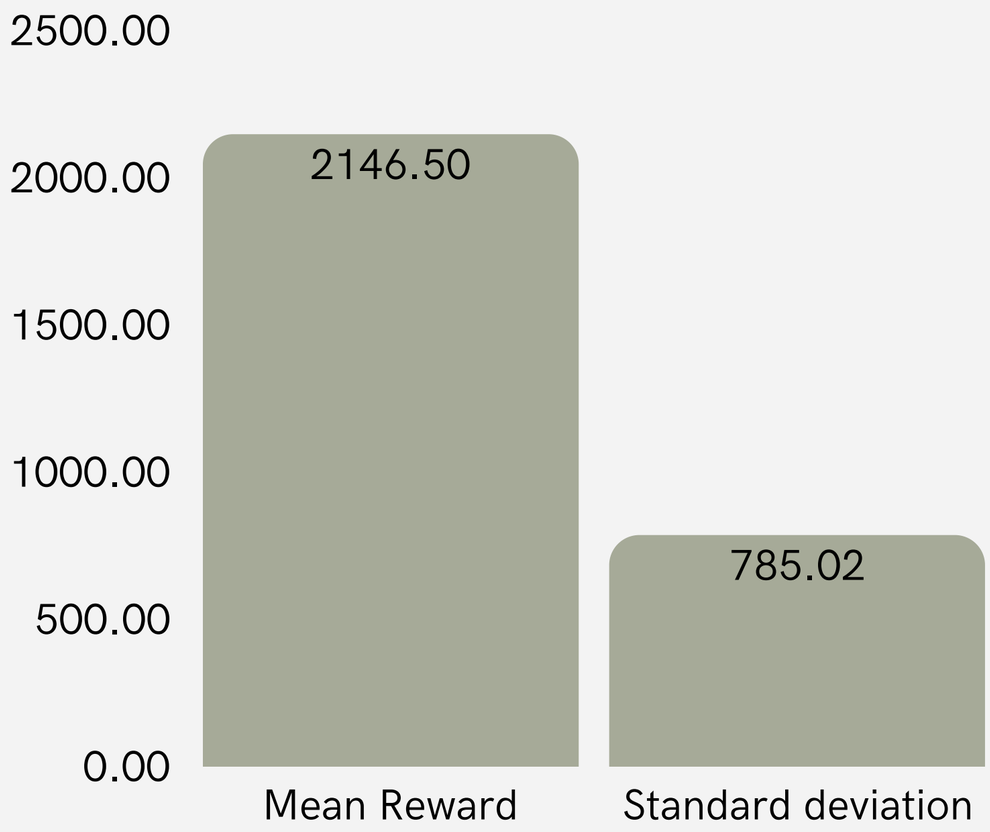
Results

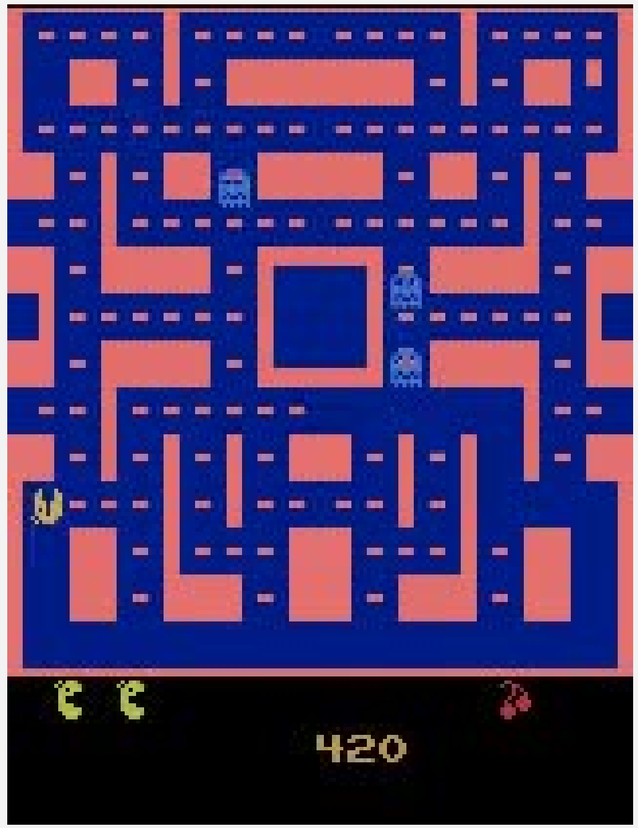
Learning rate effect:



Advantage Actor Critic

Results

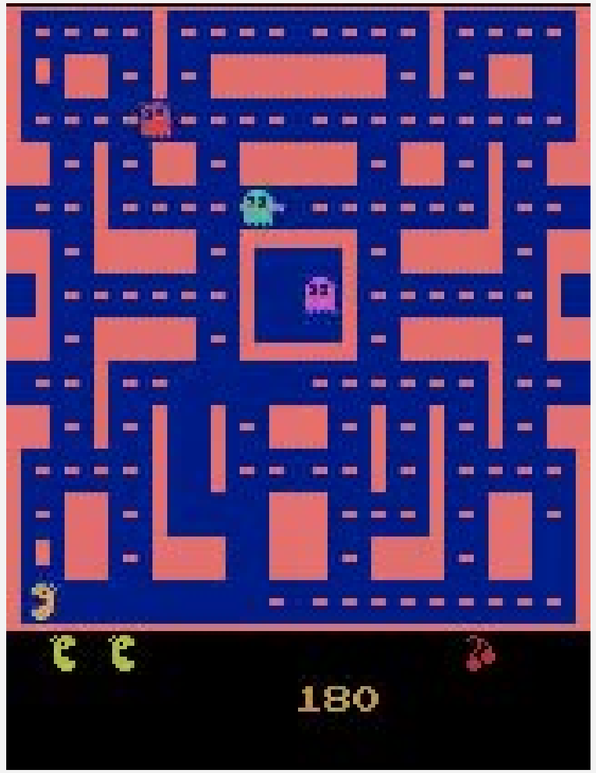


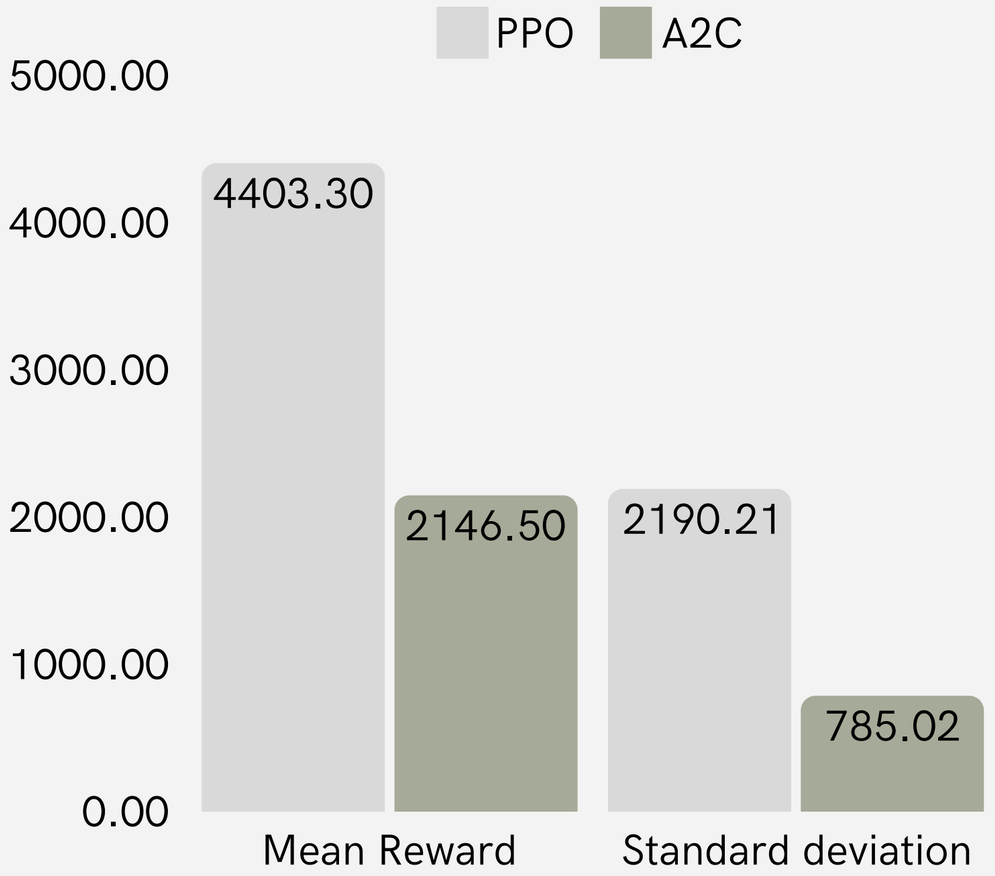


Results

Ms. Pacman

PPO





Conclusions

Ms. Pacman

Due to training dynamics and a high variance on rewards from episode to episode *Proximal Policy Optimization (PPO)* has worked better than *Advantage Actor-Critic (A2C)*, we argue that the clipping mechanism might have prevented overly large policy updates, leading to more stable learning and an overall better policy.

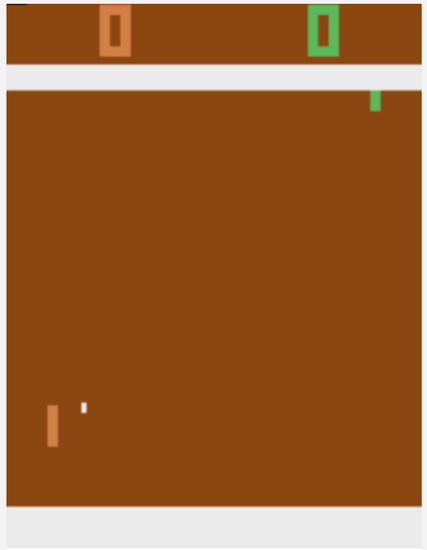
Tried penalizing lost lives** = **No major difference

Pong

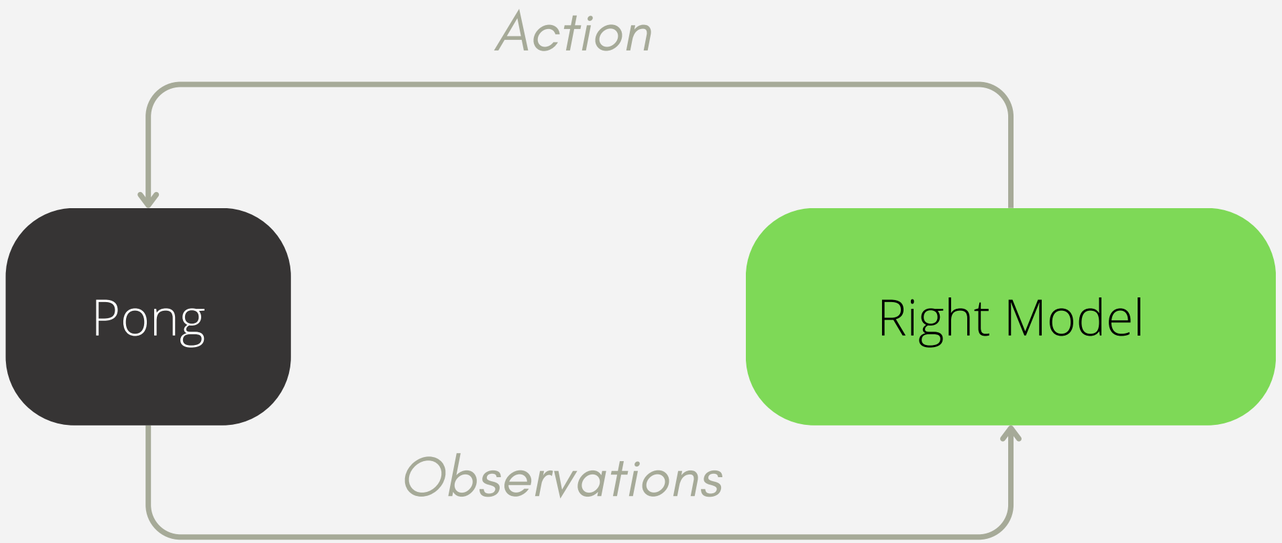
Pong

You control the right paddle, you compete against the left paddle controlled by the computer. You each try to keep deflecting the ball away from your goal and into your opponent's goal.

Value	Meaning	Value	Meaning	Value	Meaning
`0`	`NOOP`	`1`	`FIRE`	`2`	`UP`
`3`	`RIGHT`	`4`	`LEFT`	`5`	`DOWN`



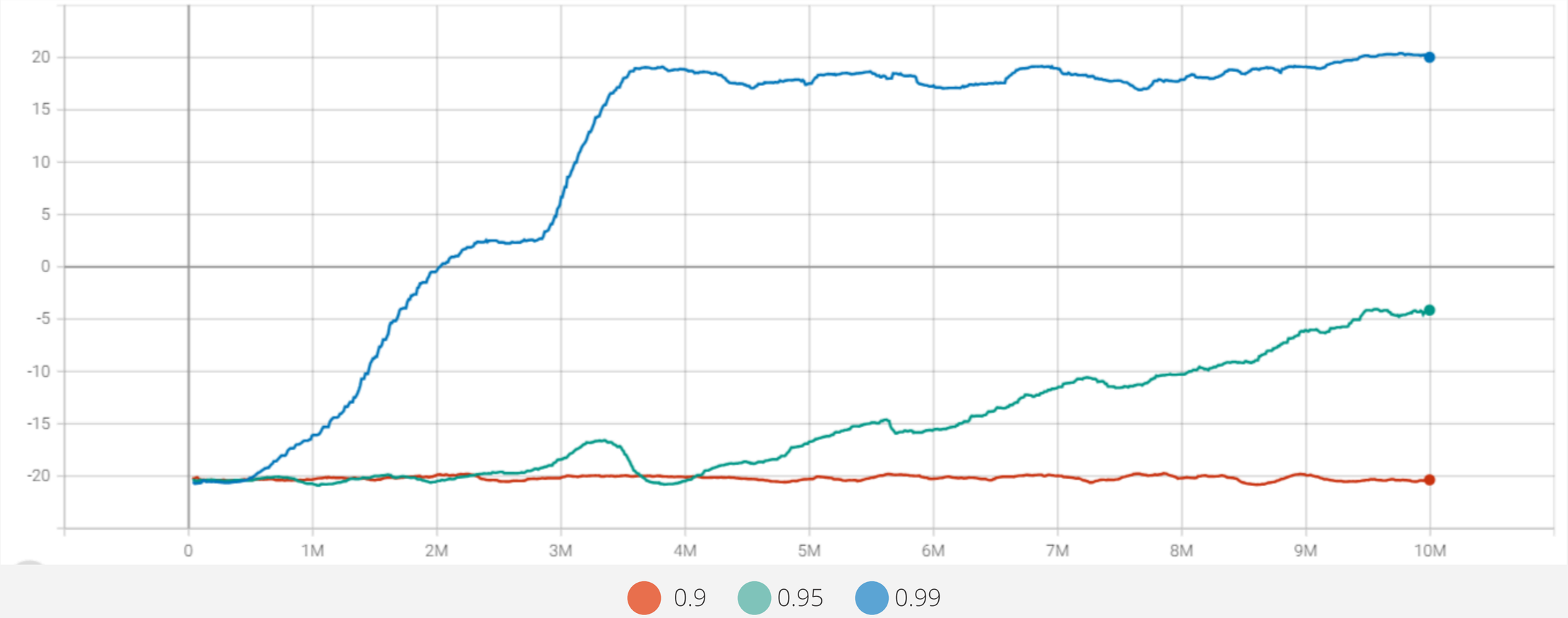
Method



Proximal Policy Optimization

Hyperparameter Search

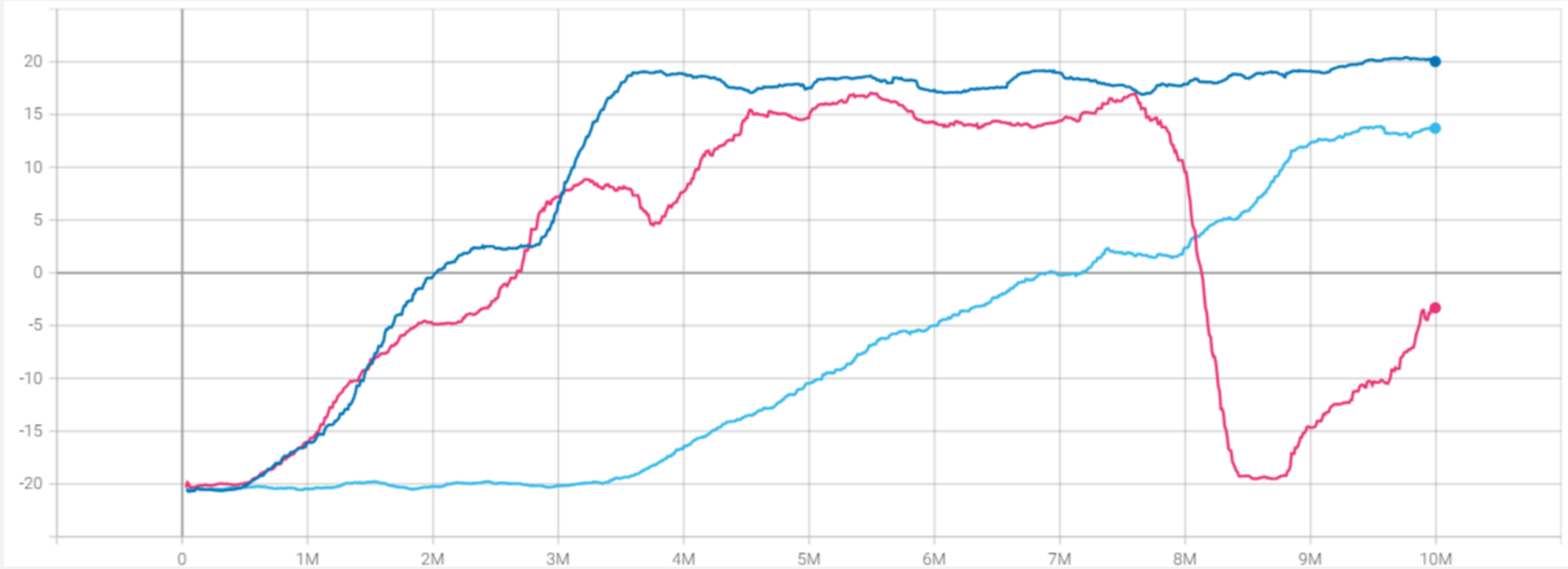
Discount factor effect:



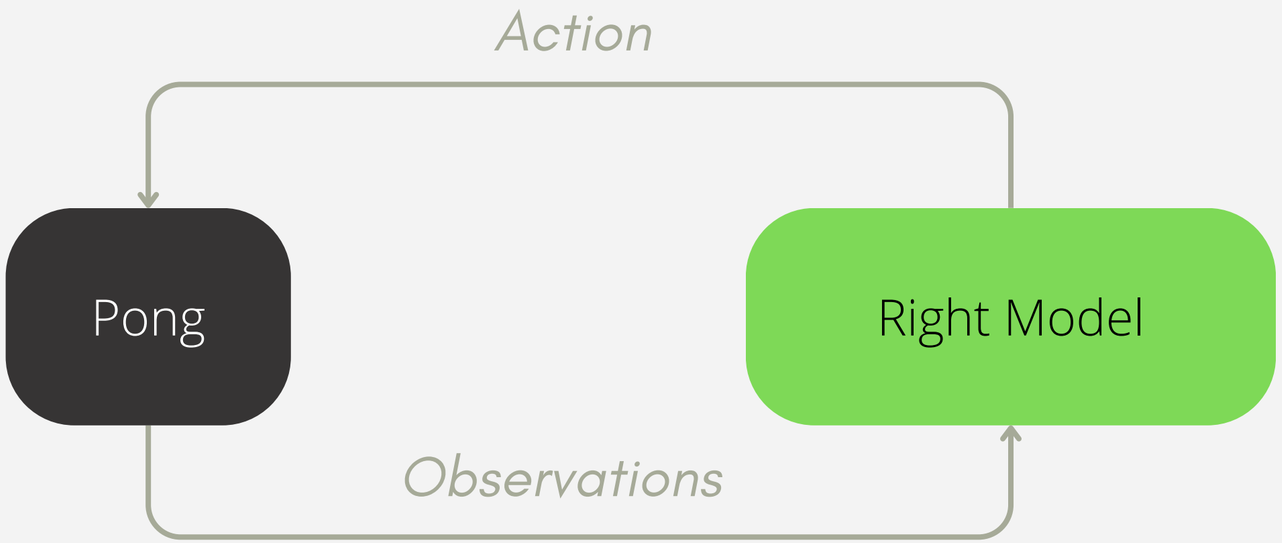
Proximal Policy Optimization

Hyperparameter Search

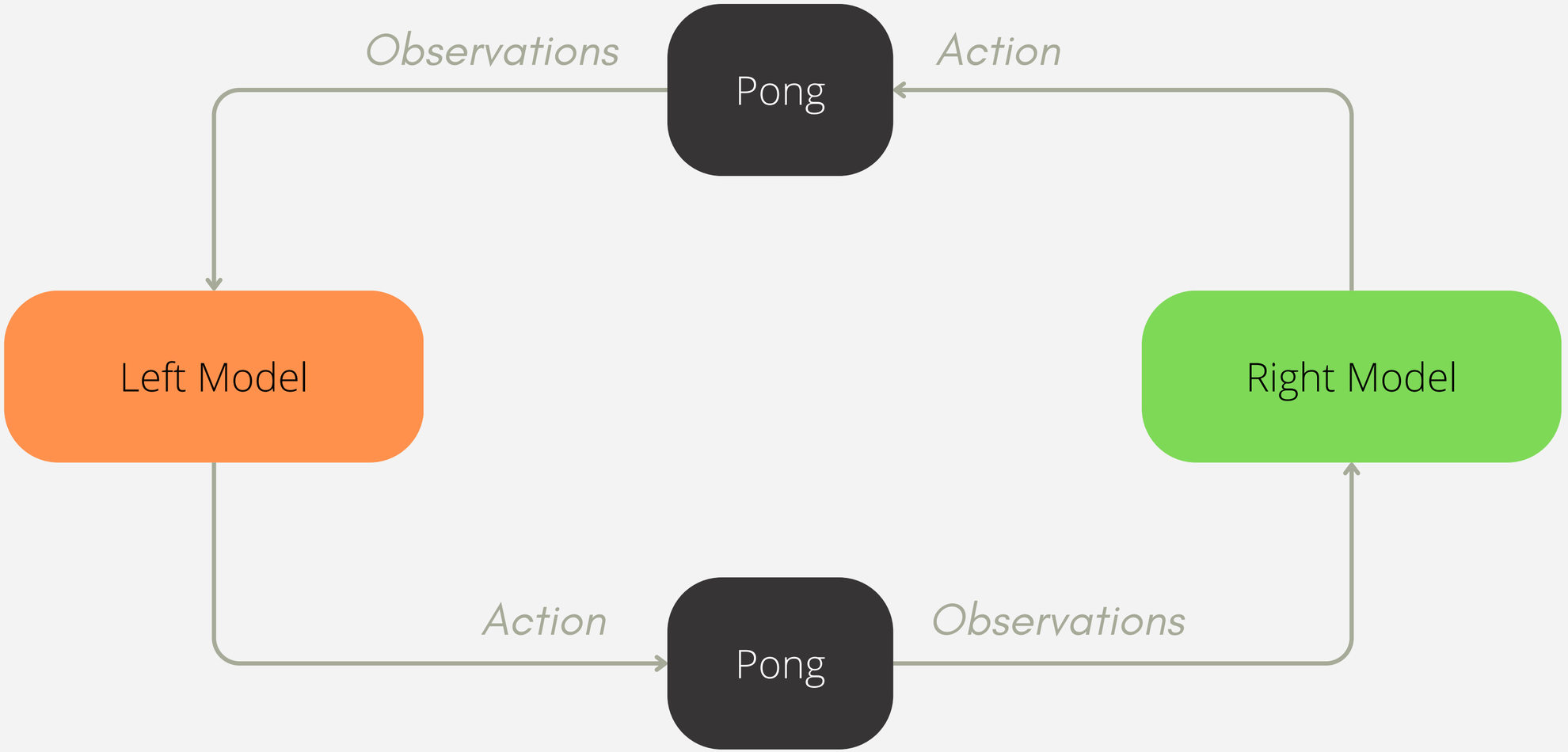
Learning rate effect:



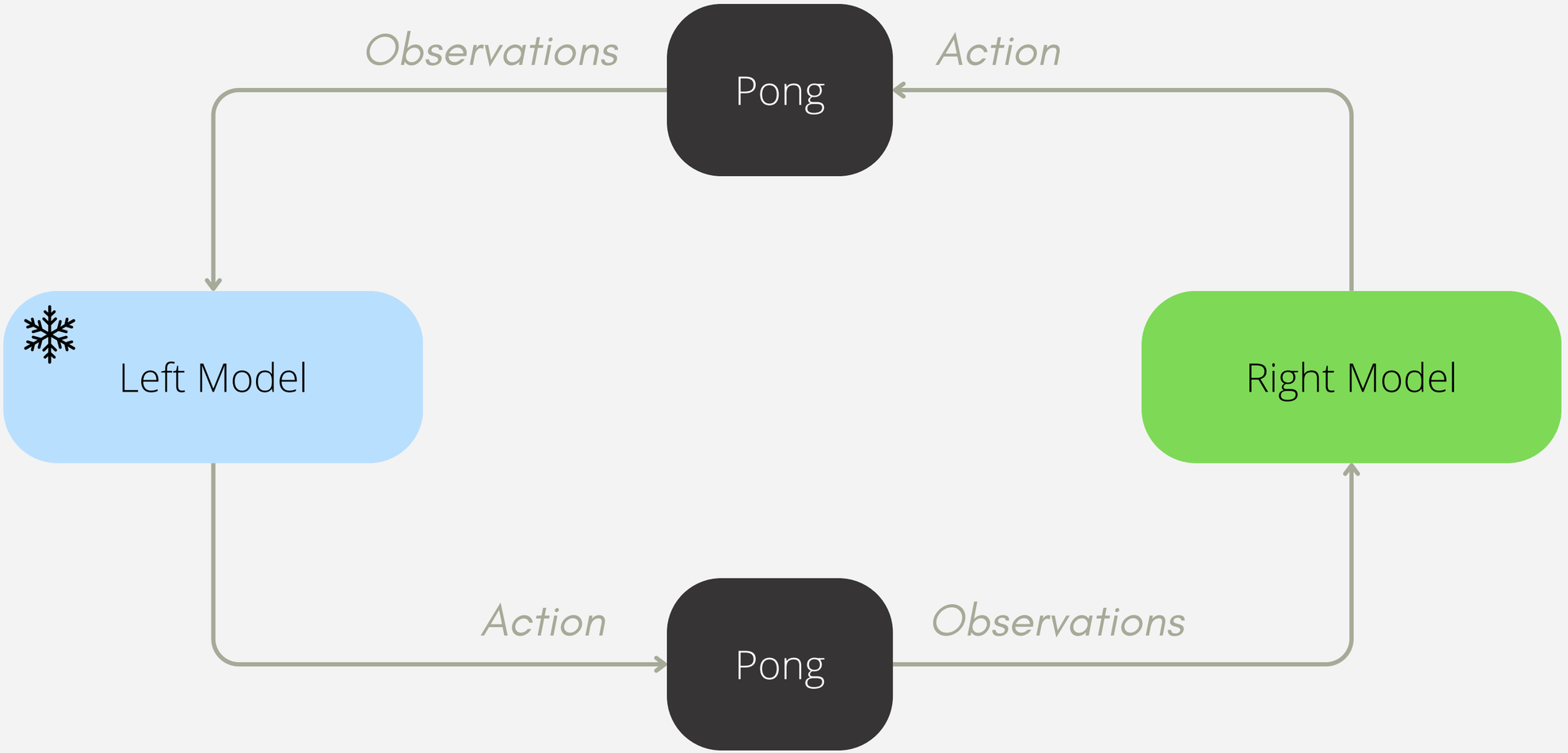
Method



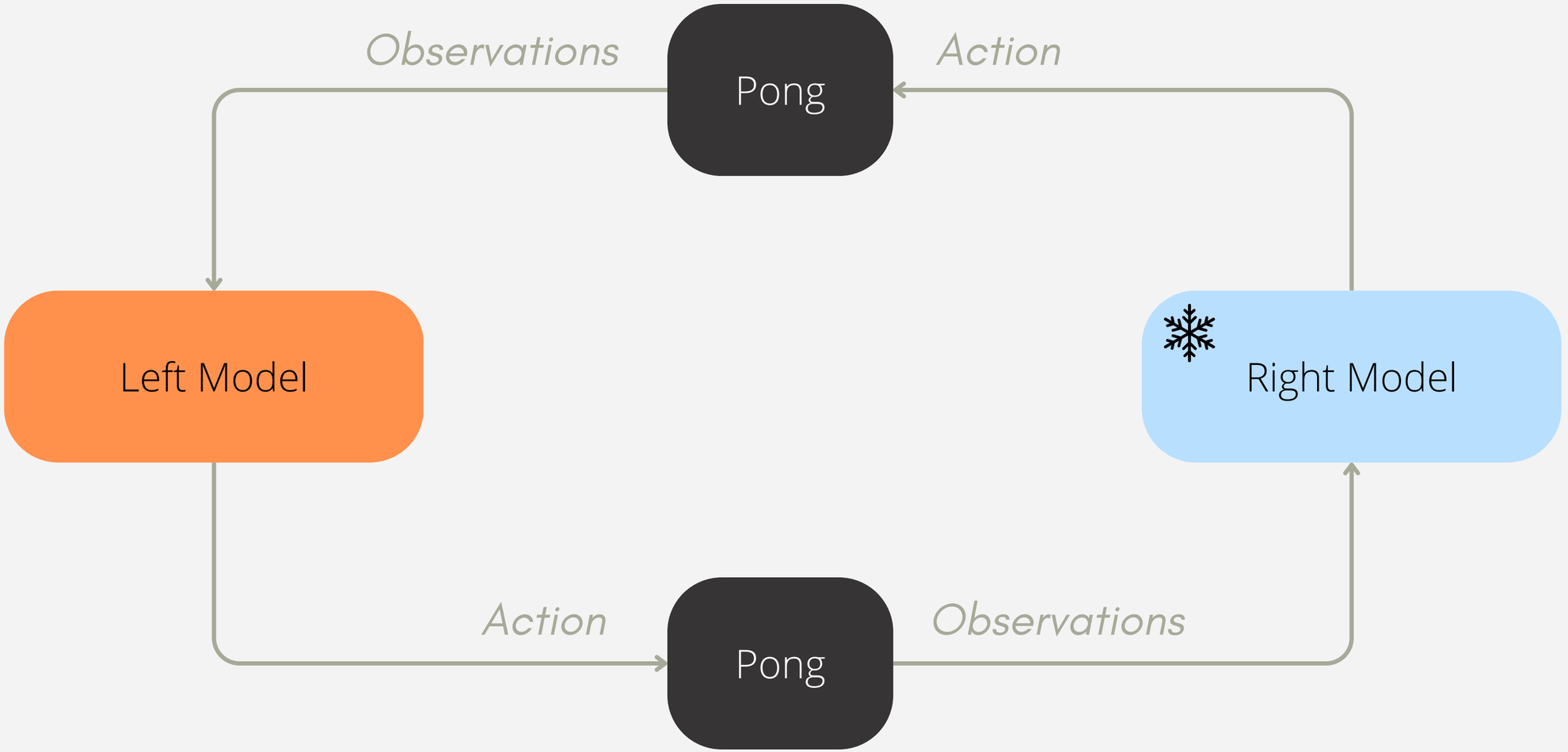
Method


Observations
Pong
Action
Left Model
Right Model
Action
Pong
Observations

Method


Observations
Pong
Action
Left Model
Right Model
Action
Pong
Observations

Method


Observations
Pong
Action
Left Model
Right Model
Action
Pong
Observations

Method

Algorithm 2 Adversarial Training with Role Switching for PPO Agents

Input: Two PPO agents A_0 and A_1 with policies π_0 and π_1, adversarial iterations n_{iter}, training timesteps per iteration T_{adv}
Initialize: Policies π_0 and π_1, alternating iteration index $i \leftarrow 0$
while $i < n_{\text{iter}}$ **do**
  if $i \mod 2 == 0$ **then** ▷ Train A_0 (Online) against A_1 (Target)
    Fix the target policy π_1 for agent A_1
    Train the online policy π_0 for T_{adv} timesteps:

$$\pi_0 \leftarrow \arg\max_{\theta_0} \mathbb{E}_{\tau \sim \pi_0, \pi_1} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

    Update agent A_0's policy with the learned parameters
  else ▷ Train A_1 (Online) against A_0 (Target)
    Fix the target policy π_0 for agent A_0
    Train the online policy π_1 for T_{adv} timesteps:

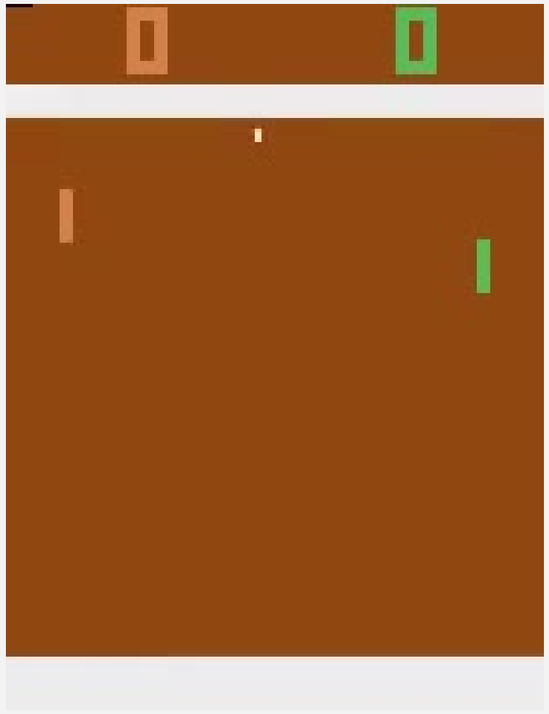
$$\pi_1 \leftarrow \arg\max_{\theta_1} \mathbb{E}_{\tau \sim \pi_1, \pi_0} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

    Update agent A_1's policy with the learned parameters
  end if
  $i \leftarrow i + 1$ ▷ Increment adversarial iteration counter
end while
Output: Final trained policies π_0 and π_1

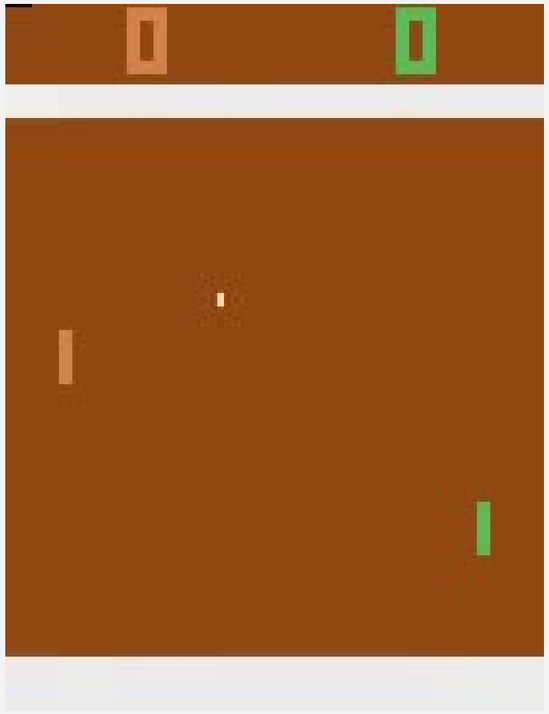
Results

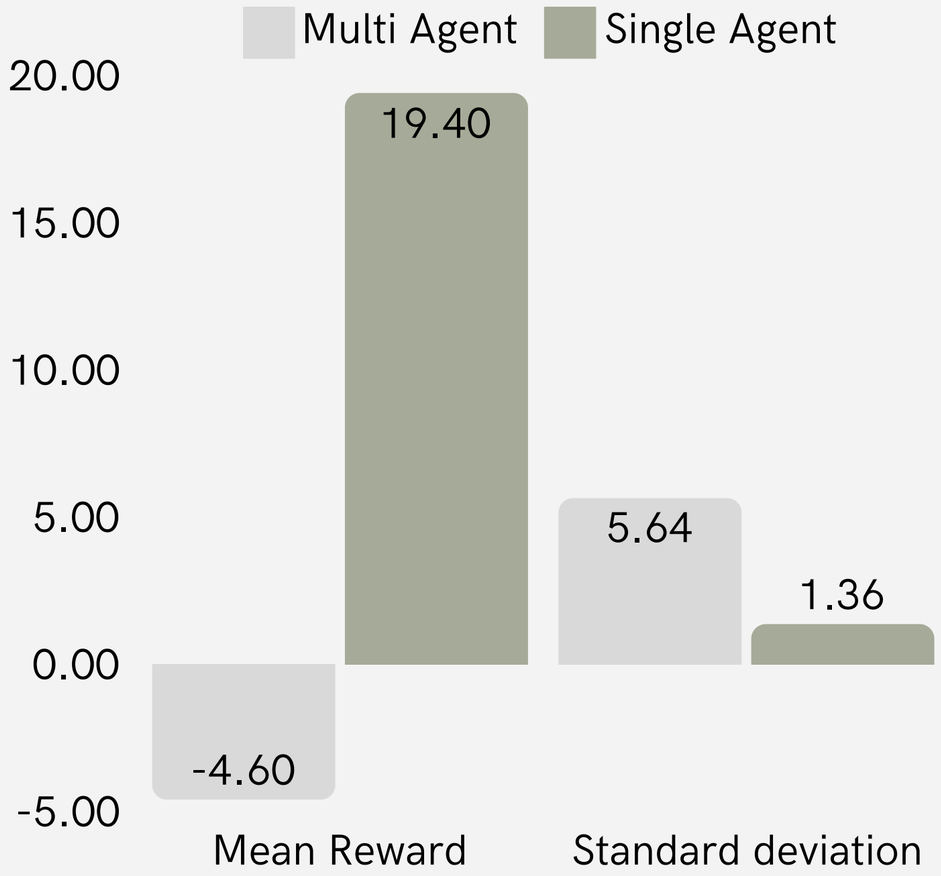
Single Agent Pong

Multi Agent Model:



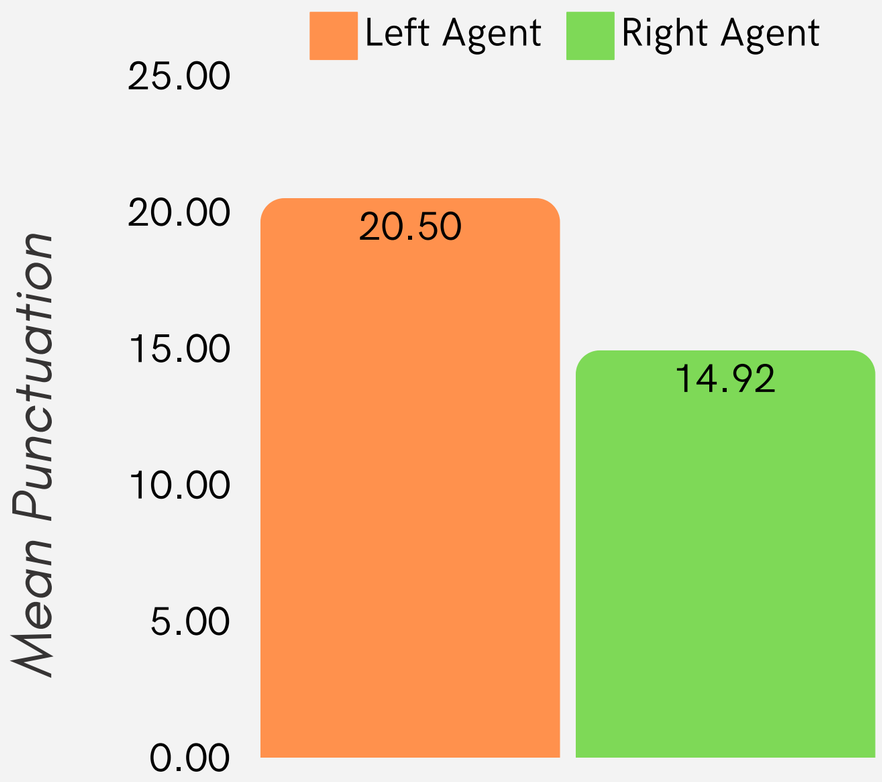
Single Agent model:

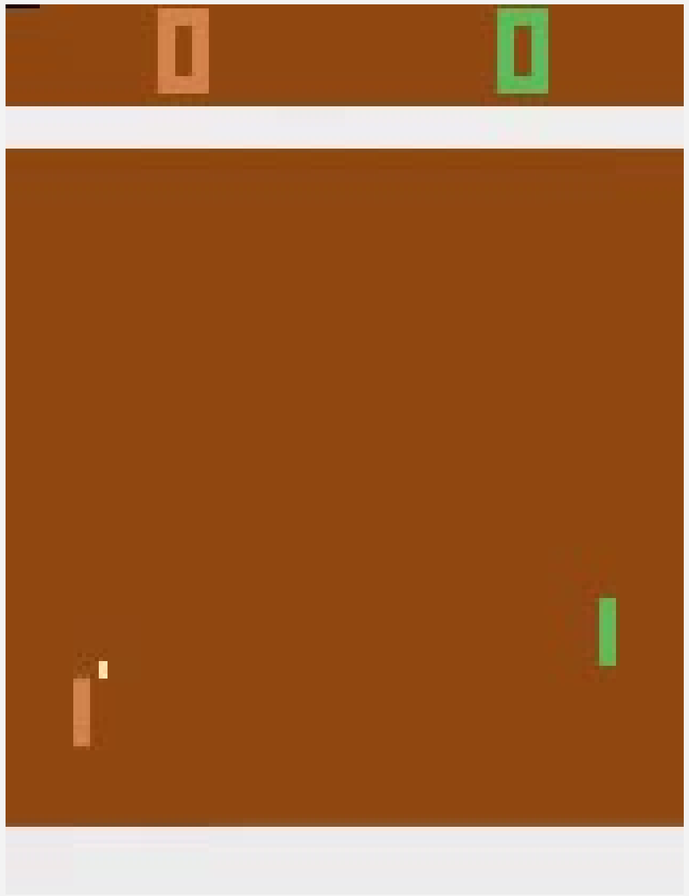




Results

Multi agent Pong





Conclusions

Ms. Pacman

The implemented adversarial agent training technique has led our agents to learn a more realistic playing policy by increasing training dynamics *"difficulty"* iteration by iteration and exploring a higher variance of pong playing dynamics compared to the default training one.

Questions?