AI DS251

Quadrons

RL Spec Ops

Gaddey Hemanth Chowdary	12140660
Anumula Chaitanya Sai	12140240
Rahul	12141300
Deva Surya Prasad	12140550

Environment

Agents: Terrorist & Soldier

Environment: They are placed at random positions in 15 x 15 grid environment with walls, and both have weapons and ability to kill each other.

Multi Agent Setup: Both Agents Have different policies and also Different Reward systems they need to learn.

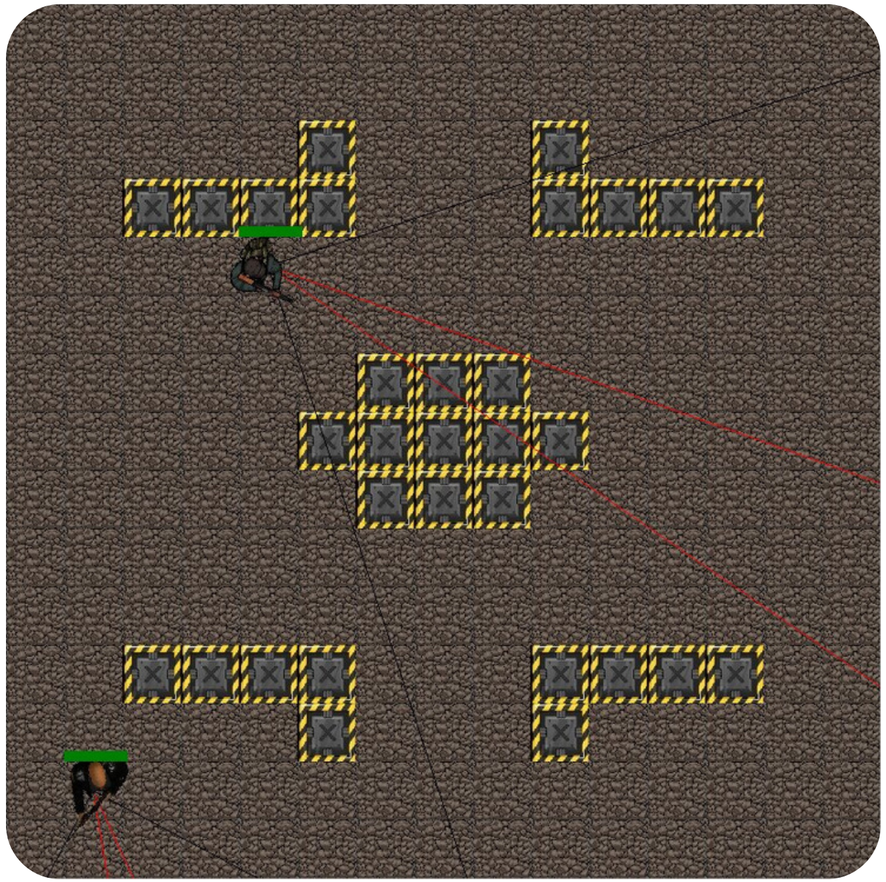
FOV: 90 degrees (We used shadow casting algorithm for fast computation which is faster than ray tracing)

Shooting Angle: 15 degrees (Agent can immediately shoot if the other agent enters it’s shooting angle)

HP: 2 (Agent can survive in shooting field for only 2 steps)

Action Space: Discrete(6) - Up, Down, Left, Right, Turn +10°, Turn -10°.

Observation Space: MultiDiscrete(8) - agent(x,y) gets info about 2 cells in all cardinal directions



Agent Reward System

Common to both

Common Rewards

Kill the other Agent: +30
Spot the other Agent: +10
Shoot the Agent: +20

Common Punishments

Get Killed by the other Agent: -20
Seen by the other Agent: -10
Get shot by the other Agent: -15
Slam Into Wall/ Go outside Region: -1

Terrorist Specific

Exploration Reward

r += 1 for visiting each latitude/ longitude

Soldier Specific

Exploration Reward

r += 1/(n_visit+1) ~ 0.022

Terrorist Tracking Reward

r += 30/Manhattan(Sol, Terr)
[Here Soldier coordinates are not live and are updated only if terrorist comes into soldier's fov]

Environment

PettingZoo

- **PettingZoo** is a pythonic interface built specifically for general multi-agent reinforcement learning (**MARL**) environments with Gymnasium like API. PettingZoo includes a wide variety of reference environments, utilities, and tools for creating custom environments.

Visualization

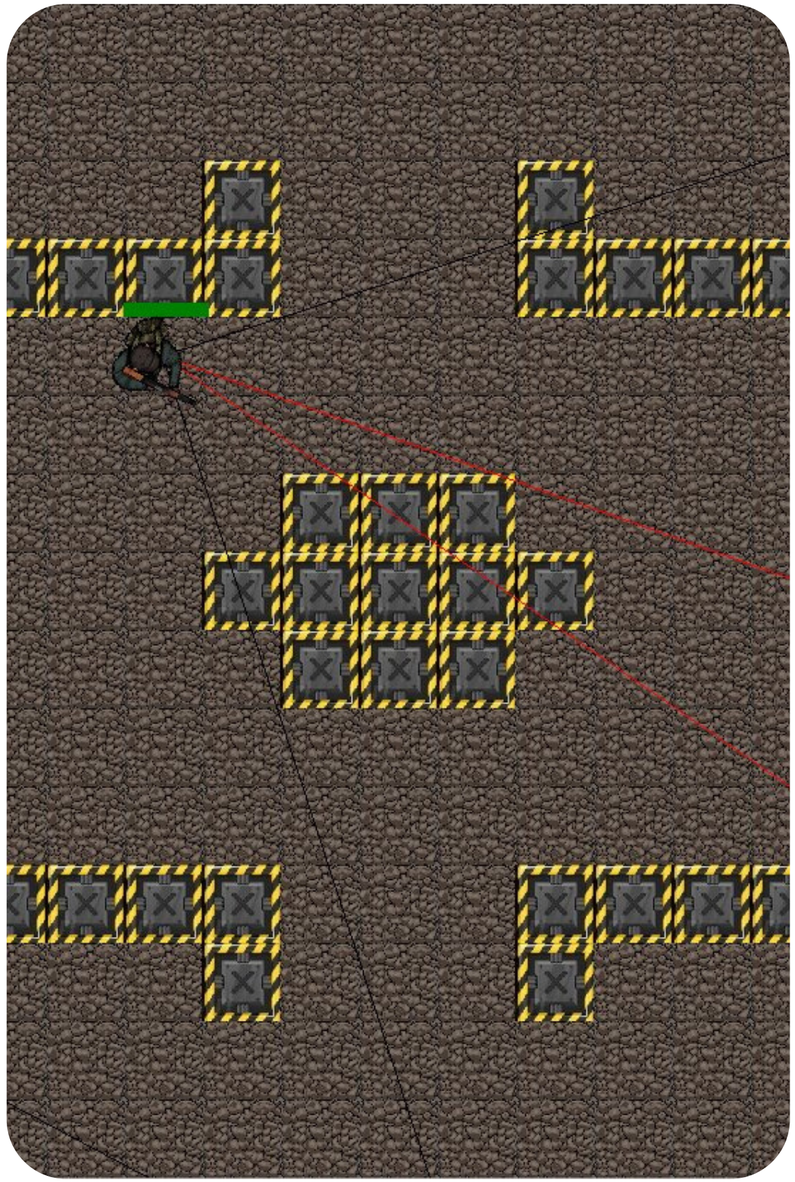
Pygame

- Python Library for primarily for 2D graphics and game development which we used for visualization.

RL Library

RLLib (Ray)

- Reinforcement learning library by Ray offering extensive suite of RL algorithms.
- **Pros:** Library is build for industrial distributed systems and hence massively parallelizes training leading to faster convergence.
- **Cons:** Is highly resource intensive and highly device centric



Training Results: DQN

Results:

DQN:

Hyper-parameters:

Learning rate: 4e-3

gamma(exploration): 0.90

Replay Buffer Size: 50,000 (with replay sequence of 1)

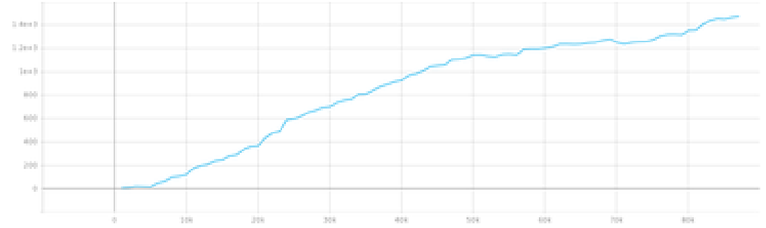
Training Batch Size: 512

Number of cpu workers: 7+1

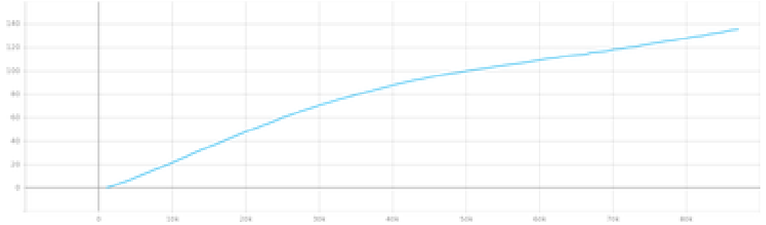
Number of gpus: 1

Framework used: Tensorflow (pytorch was not allowing use of GPU due to some RLLib bug)

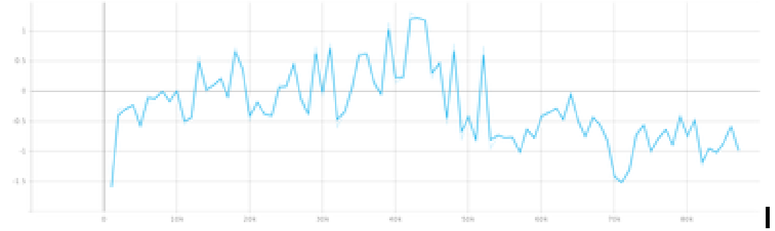
Episode Mean Reward:



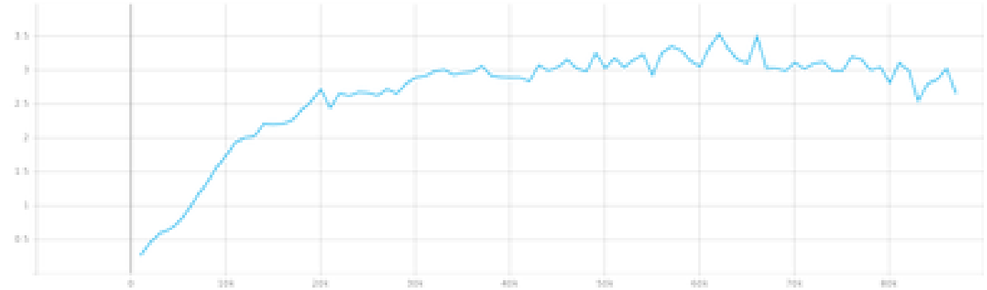
Soldier Mean Q:



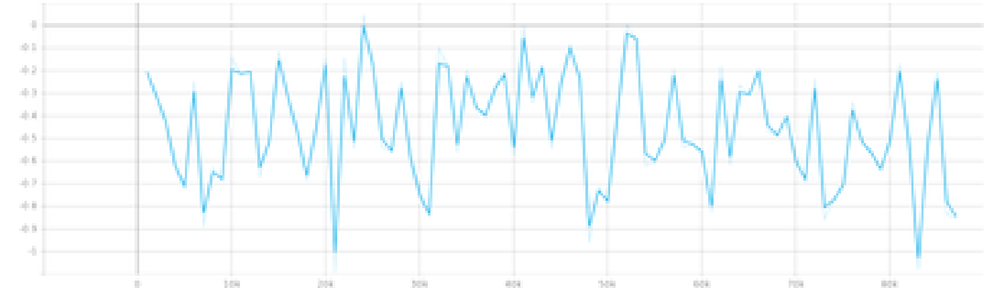
Soldier Mean TD Error:



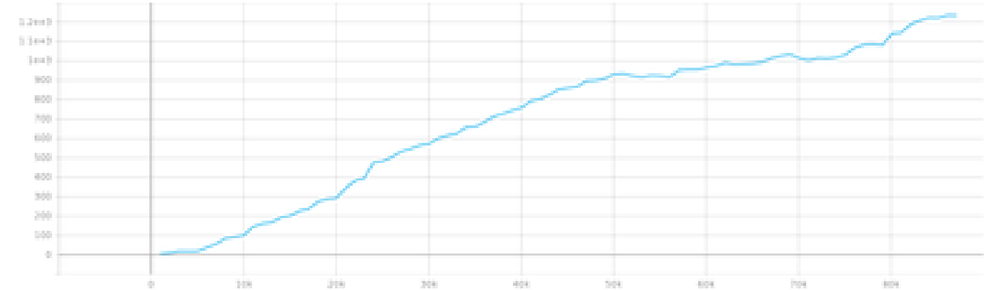
Terrorist Mean Q:



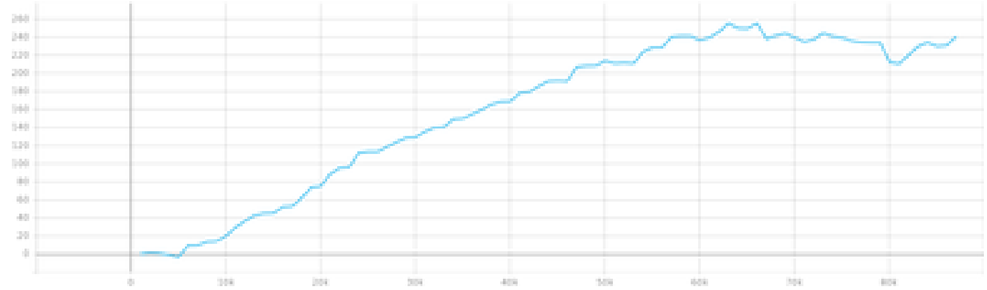
Terrorist Mean TD Error:



Soldier Policy Mean Reward:



Terrorist Policy Mean Reward:



Training Results: PPO

PPO:

Hyper-parameters:

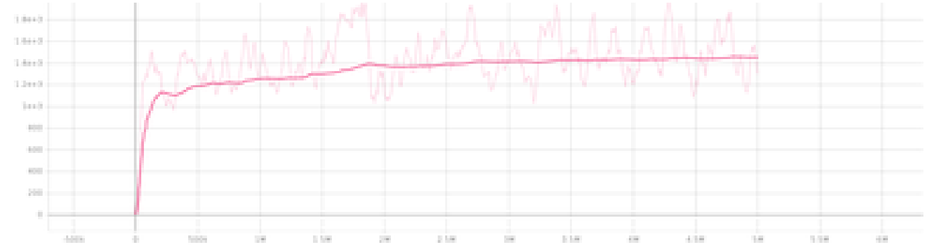
- **Learning rate:** 3e-5
- **gamma(exploration):** 0.99
- **Entropy coefficient:** 0.1
- **Vf loss coefficient:** 0.25
- **Mini batch size (for SGD):** 64
- **Number of iterations (for SGD):** 10
- **Training Batch Size:** 512

Number of cpu workers: 7+1

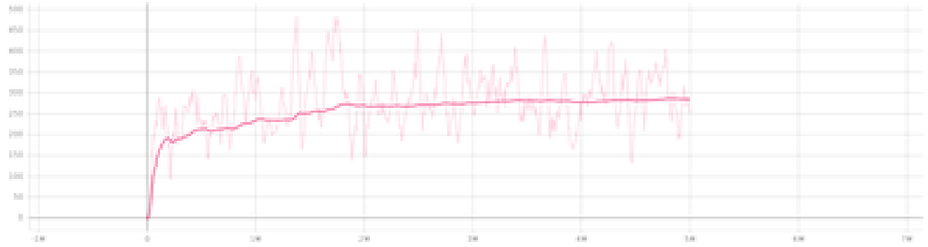
Number of gpus: 1

Episode Mean Reward:

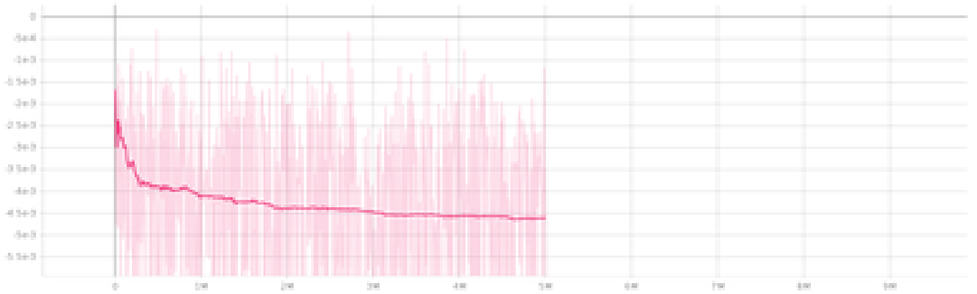
Soldier Mean Reward:



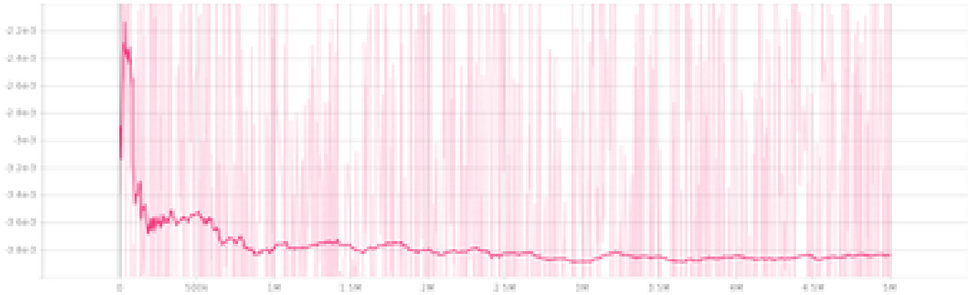
Terrorist Mean Reward:



Soldier Policy Loss:



Terrorist Policy Loss:



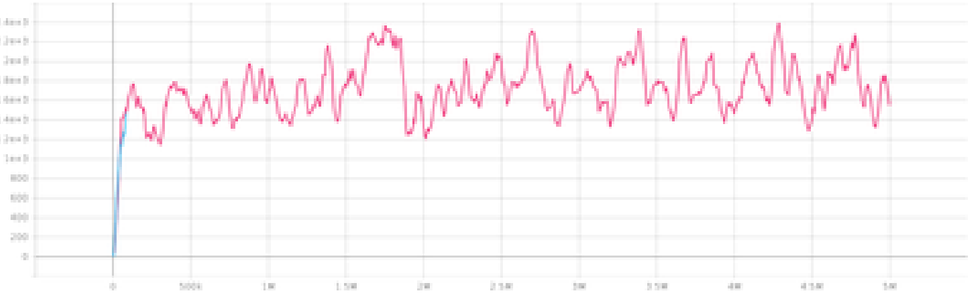
Framework used: Tensorflow (pytorch was not allowing use of GPU due to some RLLib bug)

Combined Analysis:

Episode Mean Reward:

Pink - PPO

Blue - DQN



We ran PPO algorithm for much longer time than DQN and also tried with many other hyperparameters for both DQN and PPO all of which (along with results and metrics for each iteration like KL loss, VF Loss, mean and max Q for agents, min and max rewards etc.) are stored in the given google drive link to make the report less cumbersome:

We ran PPO algorithm for much longer time than DQN and also tried with many other hyperparameters for both DQN and PPO all of which (along with results and metrics for each iteration like KL loss, VF Loss, mean and max Q for agents, min and max rewards etc.) are stored in the given google drive link to make the report less cumbersome: https://drive.google.com/drive/folders/1DbA5KpQSyQzwb3uD-XCBraDn1yYd21Sh?usp=sharing

Algorithms

DQN

Can Handle only DIscrete Action Space

It is not very stable while learning and is sensitive to reward system (especially sparse rewards)

Offline Learning

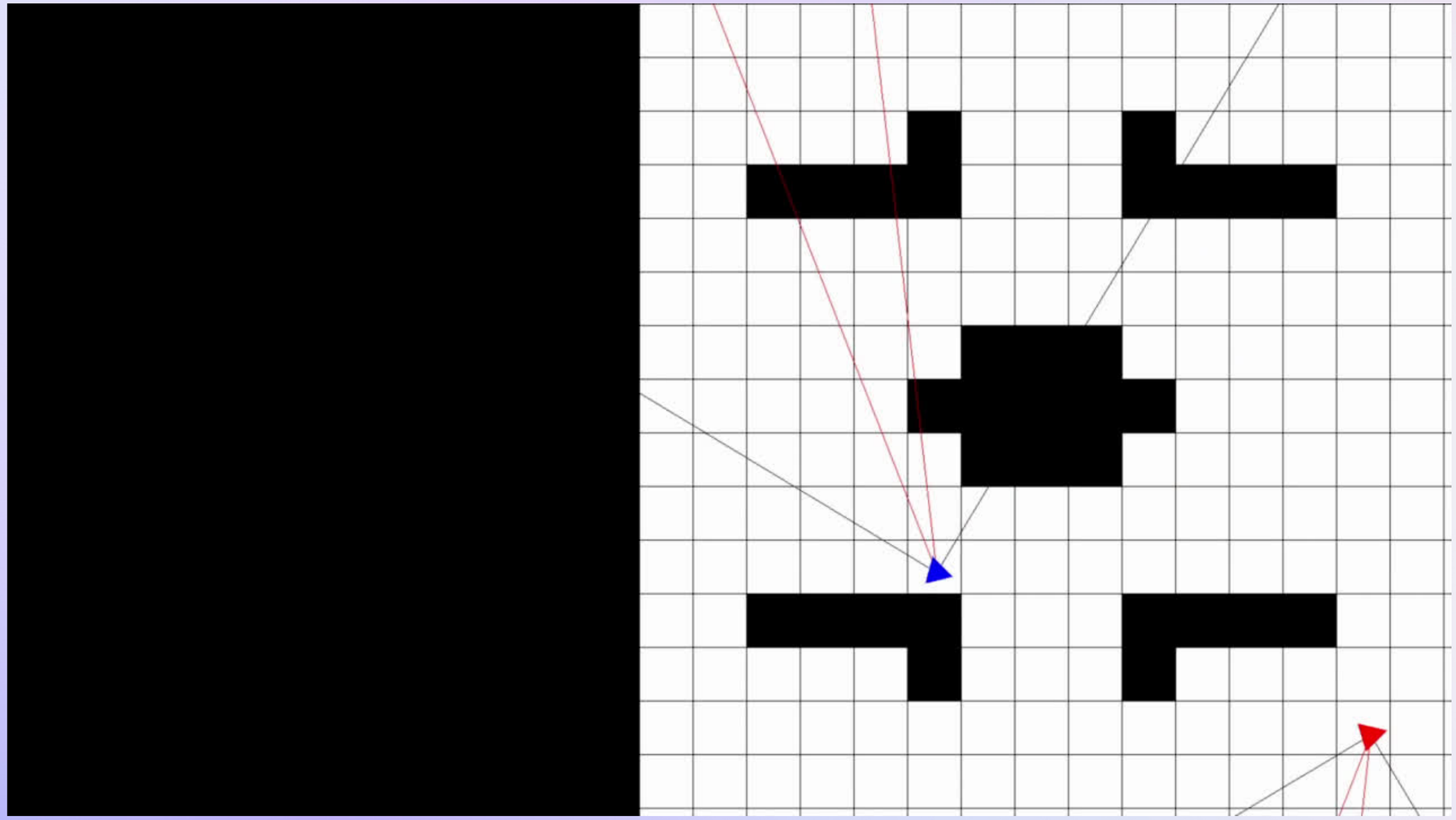
PPO

Can Handle Both Discrete and Continuous Action Spaces

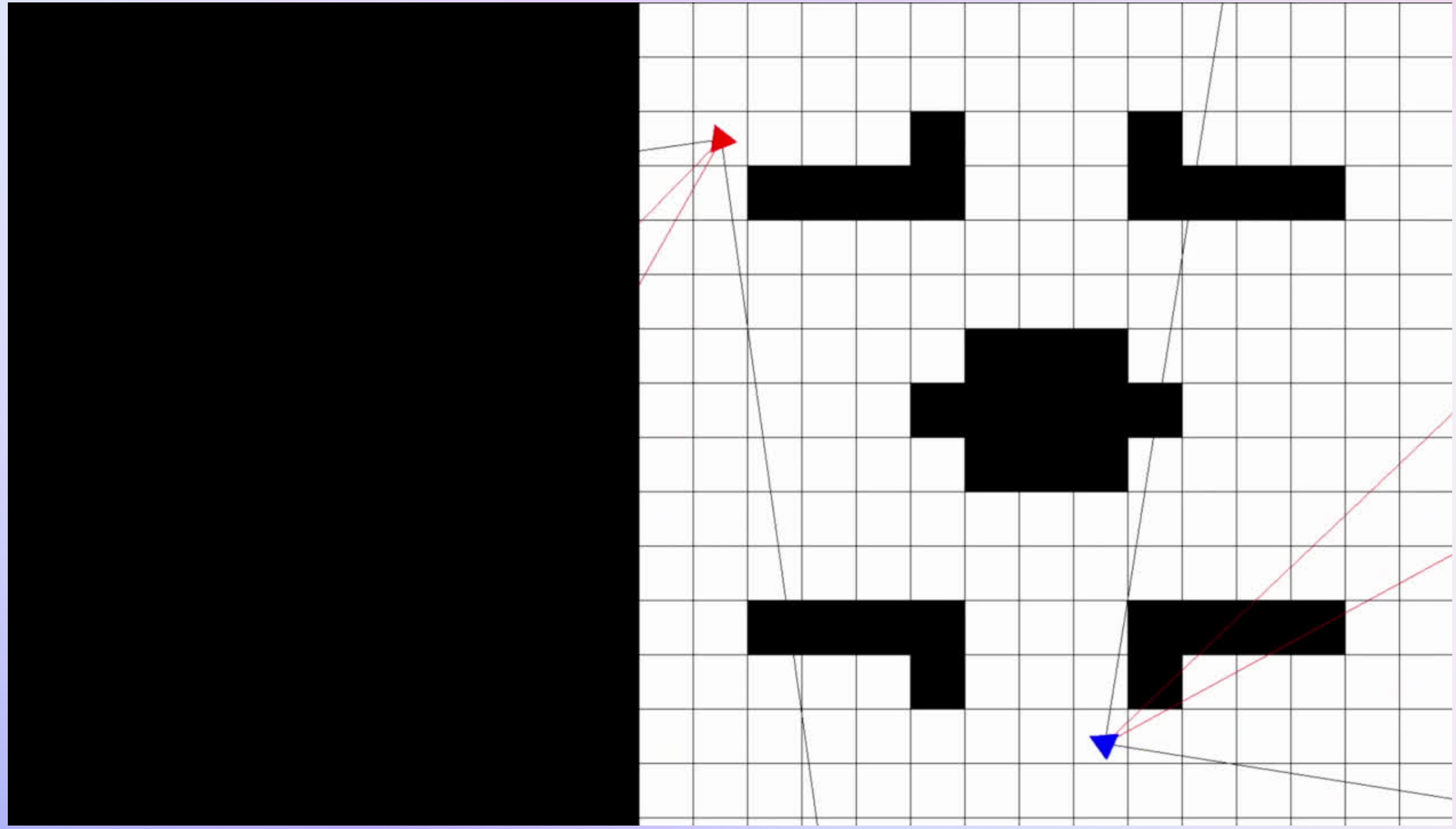
PPO’s clipped policy function allows stable convergence.

Online Learning

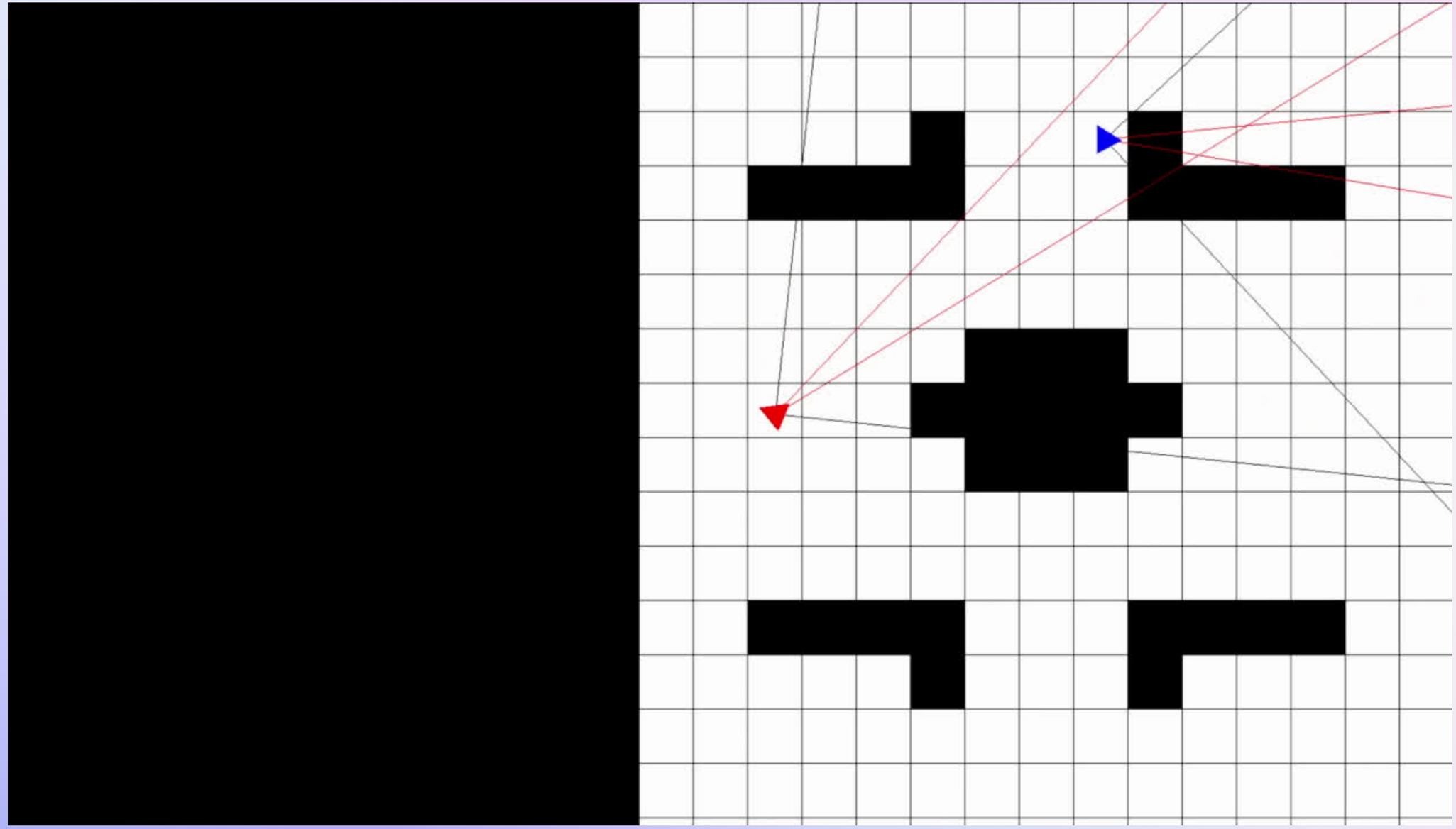
DQN



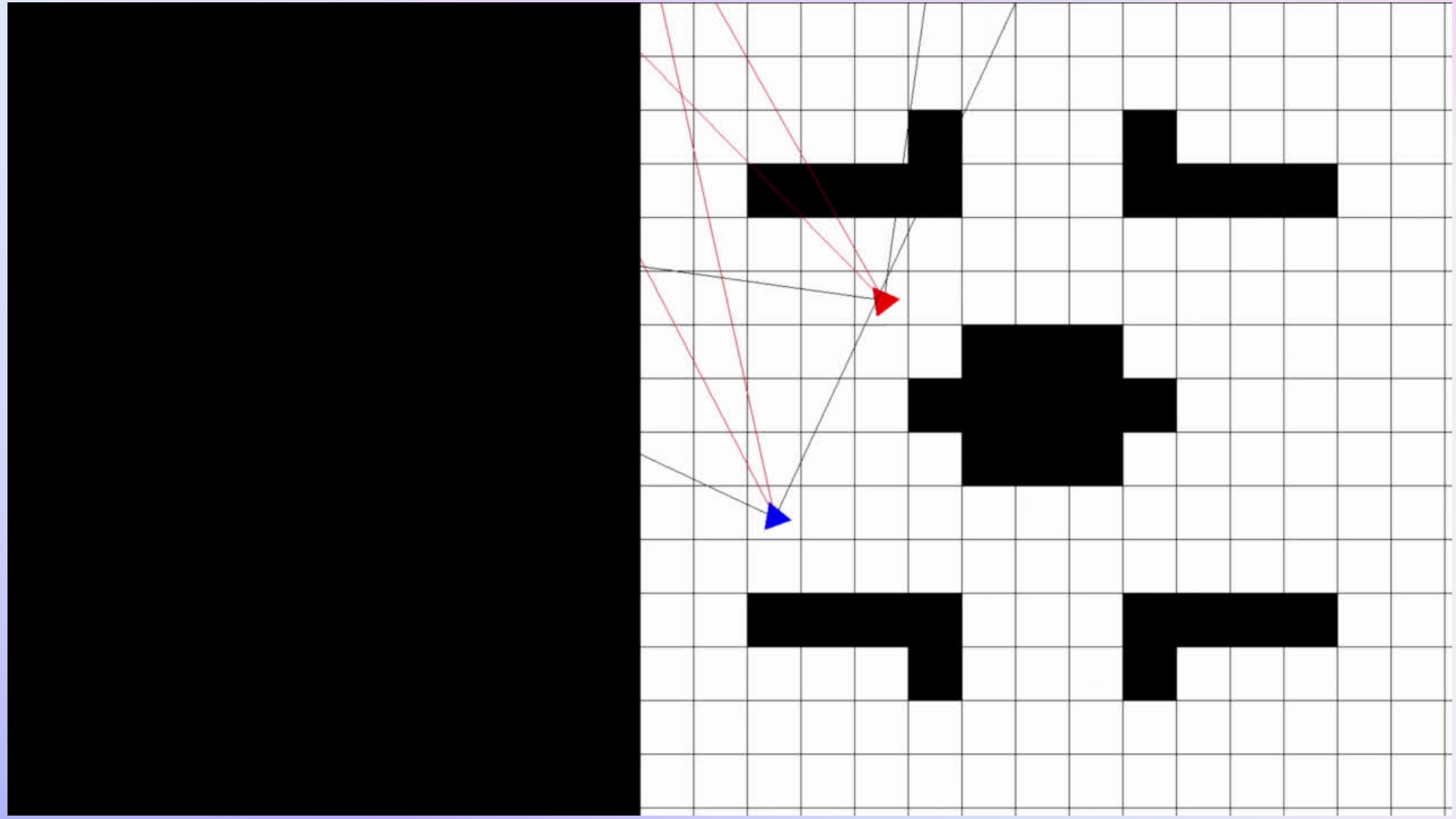
PPO



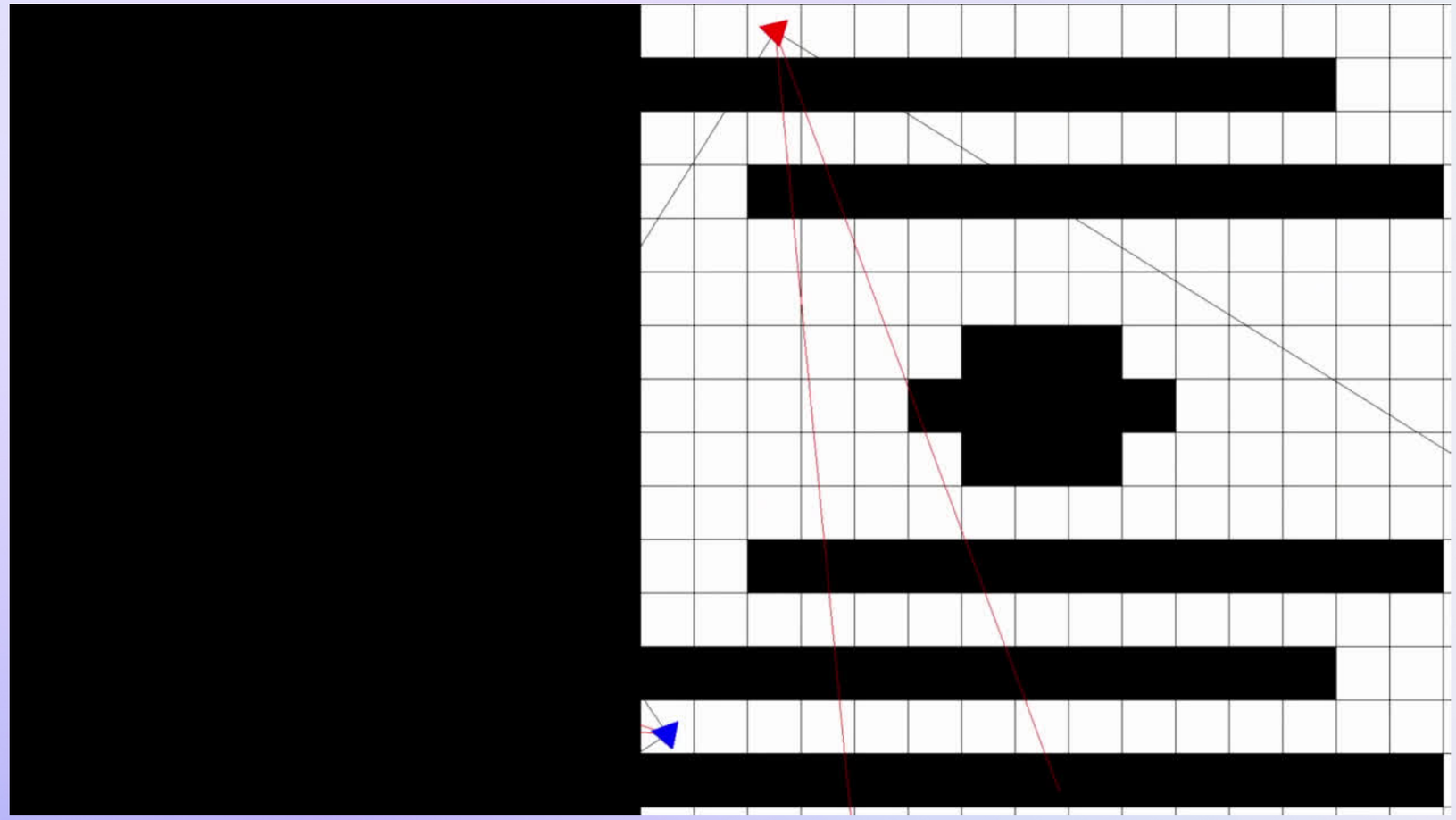
PPO



PPO



Testing in Unseen Environment



Video DEMO

- **To view the video demos, please go to https://www.canva.com/design/DAF0-S3SBmQ/5U71k_sMmyi731AOyvBuWw/edit?utm_content=DAF0-S3SBmQ&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton**

Future Work

- **Enhance Multi-Agent Spec Ops with more agents although same policy for all the soldiers and another for all terrorists.**
- **Introduce diverse and dynamic maps to challenge agent adaptability.**
- **Overcome computational constraints using distributed reinforcement learning and model compression for improved performance on powerful hardware.**