MINIGRID PROJECT

Pradumna Awasthi 200693,
Shreenaga Tejas 200296
Lavesh Mangal 200546

Important Links

1. REINFORCE:https://colab.research.google.com/drive/1OcrIJxWlAVYSH9n32HXB8wK9AmOyb6zr?usp=sharing
2. Rl-starter files :https://drive.google.com/drive/folders/1ncJky-G7CIQUBhRpoOrTE7MuHCDOnCIb?usp=sharing
3. PPONotebook:https://colab.research.google.com/drive/1YuYP-cvUVpd7DbJmSg-g3uMHzaXGsRhZ?usp=sharing
4. Video Recording : https://drive.google.com/drive/folders/1tY-_ZnAAfy8gs6saGo6E5KCMttTkmCW3?usp=share_link

Wrapper Functions:

https://github.com/rohitrango/gym-minigrid/blob/master/gym_minigrid/wrappers.py

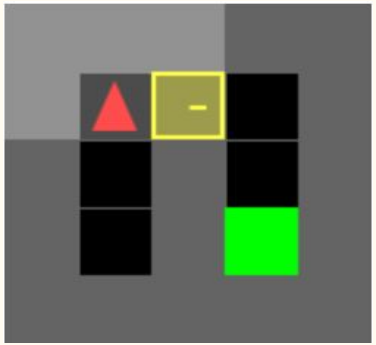
Online version of these slides:

https://docs.google.com/presentation/d/1dVo81oUygH5ygMoYCVWBWreb-F10L24YqVE5tNj-eq0/edit?usp=sharing

STEP 1: UNDERSTANDING THE ENVIRONMENT

Door Key Environment:

This environment has a key that the agent must pick up in order to unlock a goal and then get to the green goal square. This environment is difficult, because of the sparse reward, to solve using classical RL algorithms. It is useful to experiment with curiosity or curriculum learning.



ACTION SPACE :

NUM	NAME	ACTION
0	Left	Turn Left
1	Right	Turn Right
2	Forward	Move Forward
3	Pickup	Picking Object
4	Drop	Unused
5	Toggle	Activate/Toggle Object
6	Done	Unused

Door Key Environment:

OBSERVATION ENCODING:

- Each Tile is encoded as a 3-Dimensional Tuple (OBJECT_IDX, COLOR_IDX, STATE)
  - Here 7x7 is the partial view of tiles hence we get a 7x7x3 matrix as the observation.

Object to Index Mapping:

- 0 - empty
- 2 - wall
- 1 - floor
- 4 - door
- 5 - key
- 8 - goal state

Color to Index Mapping:

- 0 - Black
- 1 - Green
- 4 - Yellow
- 5 - Gray

State Space:

- 0 - open
- 1 - closed
- 2 - locked

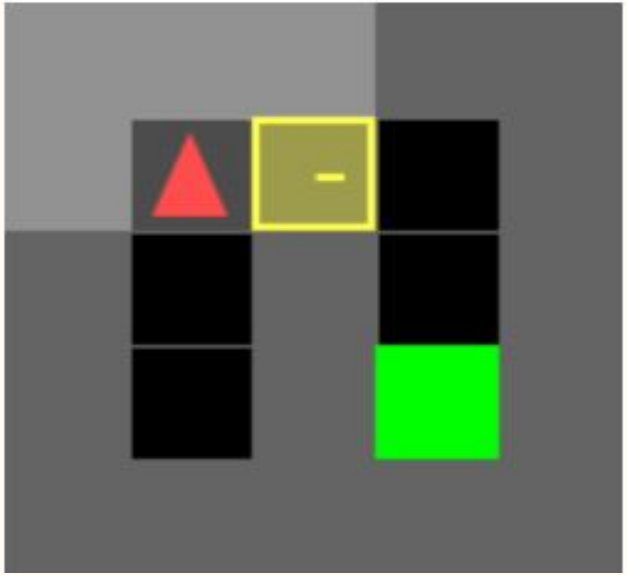
Door Key Environment:

Reward Value Mechanism:

- ‘1 - 0.9 * (step_count / max_steps)’ for success.
- ‘0’ for failure.

Termination Protocol:

- The agent reaches the goal.
- Timeout: A maximum number of steps (max_steps) have been defined.



ISSUES FACED:

- The gym environment provided to us has been changed to gymnasium environment with an incomplete implementation of libraries. As the Gym environment was outdated hence the newer libraries were not supported.

- As the Action Set is large and the reward is received only at the end. The possibility that the agent will reach the final state for at least once through random actions is very small and hence the reward mechanism was inadequate.
- We found that as such the (7,7,3) observation space was incompatible to use with convolution networks.

OUR SOLUTIONS:

- Implementing Policy Gradient Reinforce Algorithm.
- Changing the Reward Mechanism to motivate it to clear milestones.
- Limiting the action space of the agent to 5 actions.
- Using wrappers such as RGBImgPartialObs wrapper for handling the 7x7x3 observation space converting and resizing it into 56x56x3 for suitable use in CNN.

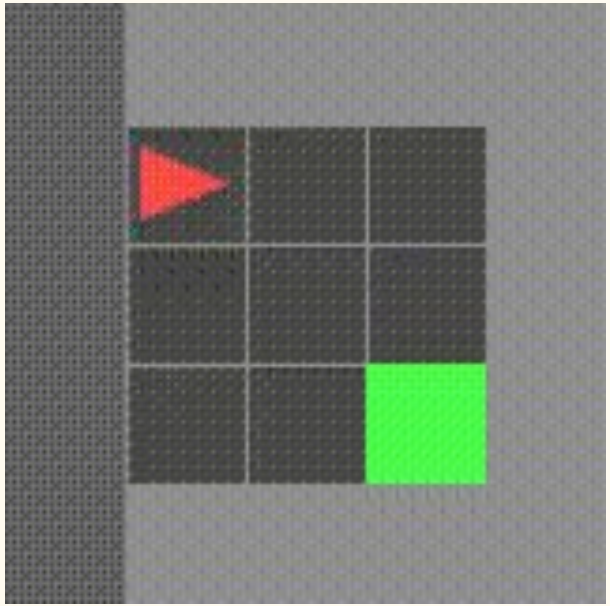
STEP 2: IMPLEMENTING THE REINFORCE ALGORITHM

Reinforce Algorithm:

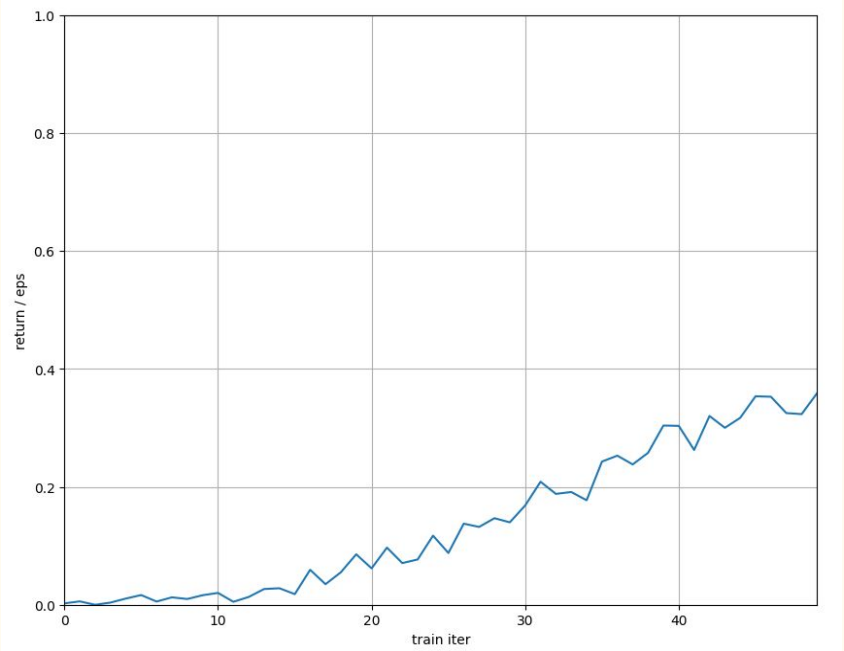
- We wanted to test the mini grid environment. So, we choose to apply a simple algorithm on a simpler mingrid.
- We first chose `MiniGrid-Empty-5x5-v0` .
- We tested it for 50 iterations only to get used to the working of the mini grid environment.
- We enabled **full view** instead of partial for this and converted the dimension of observation from **2D to 1D** and trained a **neural network** on it.

Results

After Training video



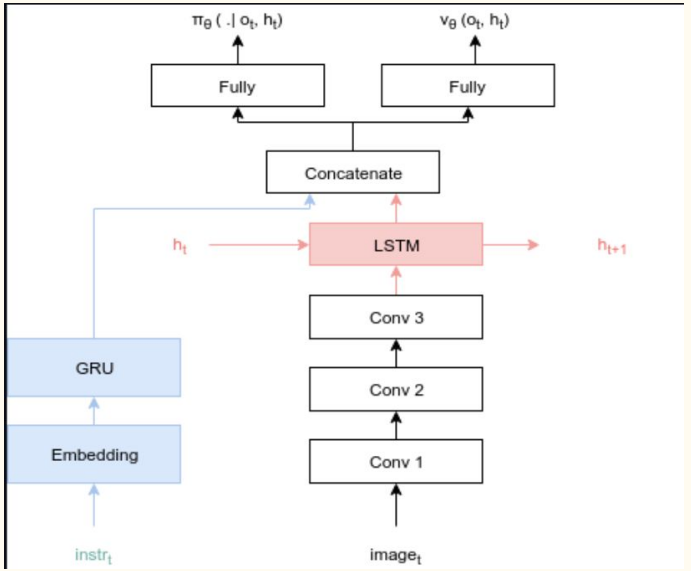
Return Plot



STEP 3: CHANGING REWARD MECHANISMS

PPO Algorithm and Code Implementation:

- We utilized the "rl-starter-files" repository to implement the Proximal Policy Optimization (PPO) algorithm for our problem.
- To modify the reward mechanism, we designed a custom wrapper on the environment.
- This repository was particularly suitable for our task because it already supported environments similar to DoorKey, such as Memory-Random, and resolved various dependency conflicts by employing the latest version of Gym and MiniGrid.
- Our implementation employed CNN architectures and implicitly incorporated memory by incorporating LSTM cells into the network.



Implementation Architecture

Experimentation and Approach Finalisation:

Approaches Tried	Results
Used the reward mechanism as mentioned in The Minigrid task (1- (num_steps)/(max_steps)) on completion.	✗
Discretized the reward mechanism on reaching goal giving +10 reward on reaching the goal.	✓
Motivated the agent to collecting the key and opening the door by giving it reward +3.0,+6.0	✓
Demotivated the agent for colliding into walls and obstacles	✓
Demotivated the agent from doing the toggle action at aimless obstacles.	✗
Motivated the agent to perform the forward action after opening the door	✗
Adding the negative reward for every step executed of exploration	✓

Final Reward Mechanism:

+3.0 For collecting Key(One time)

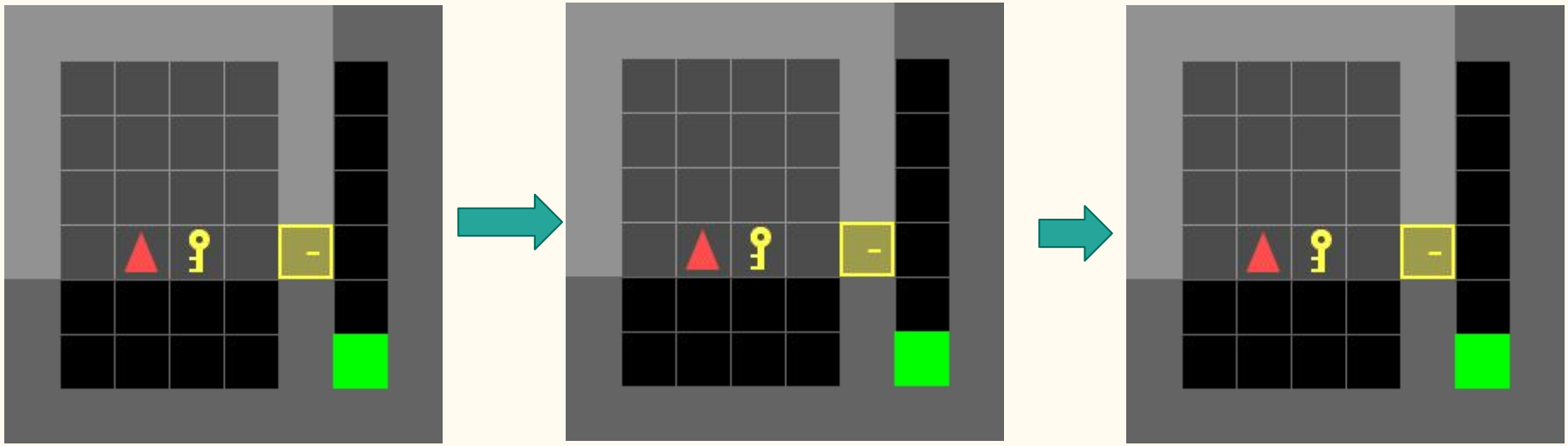
-1.0 for colliding with wall

+6.0 For opening door(One time)

+10.0 for reaching goal

-0.10 otherwise

Improvement Journey

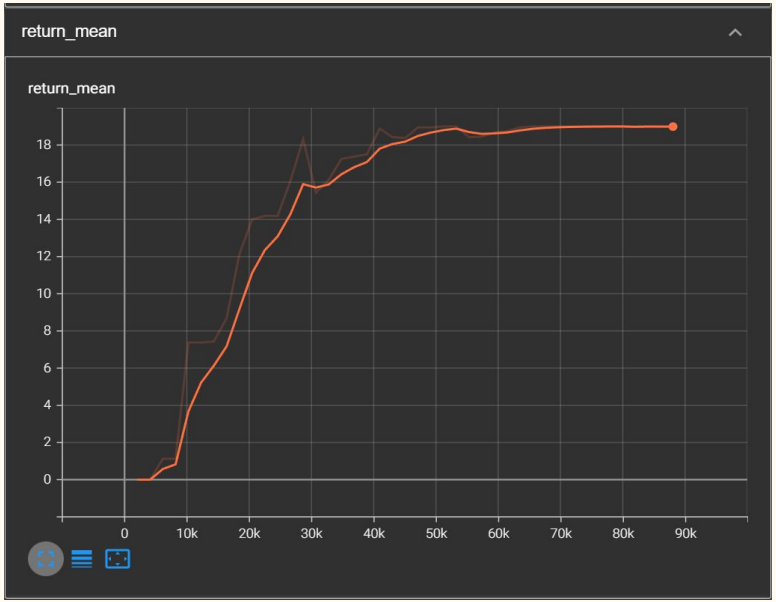


Proximal Policy Optimization Results.

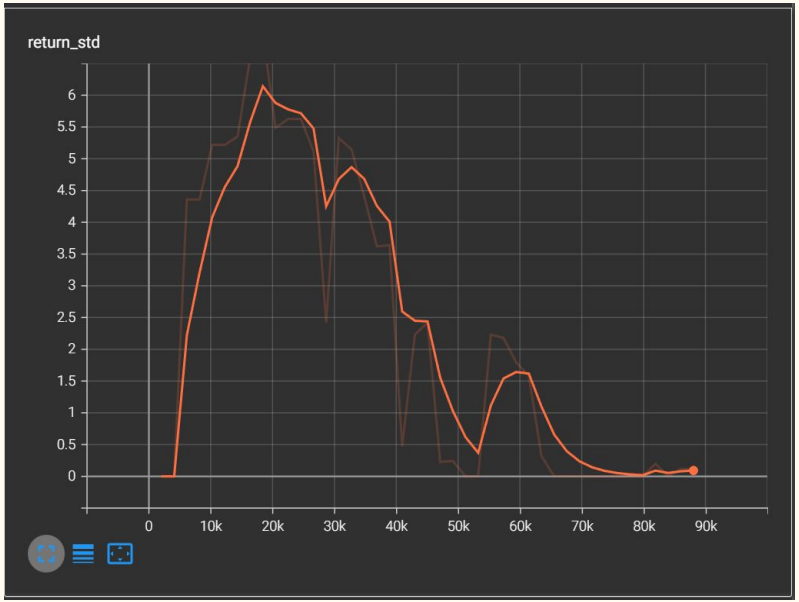
Minigrid-Door Key- 8X8-v0 Environment

Training Statistics:

Return Mean

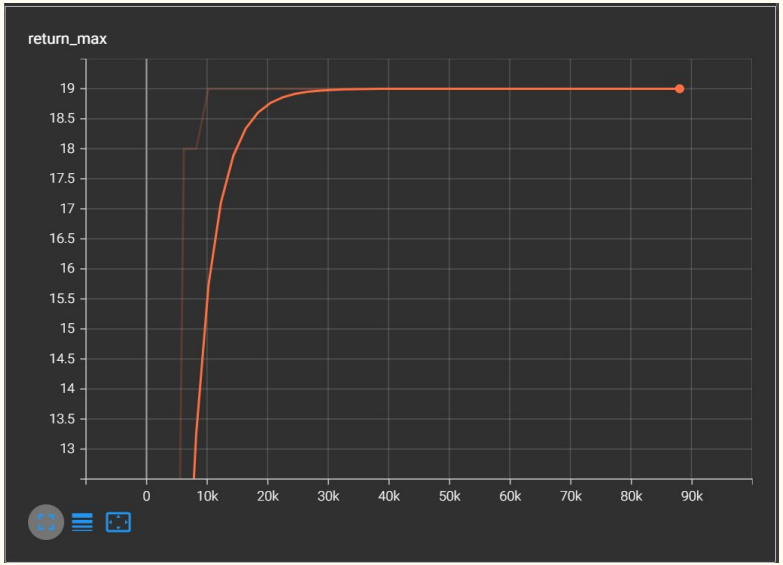


Return Std

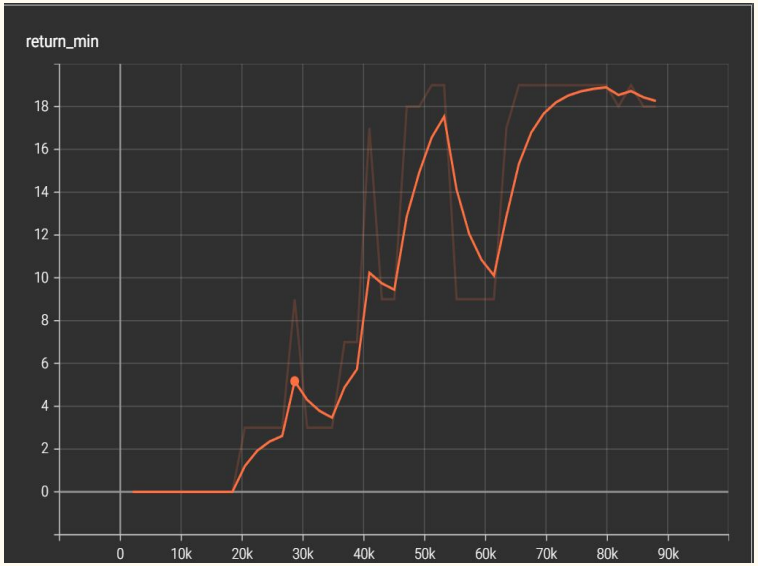


Training Statistics:

Return Max



Return Min



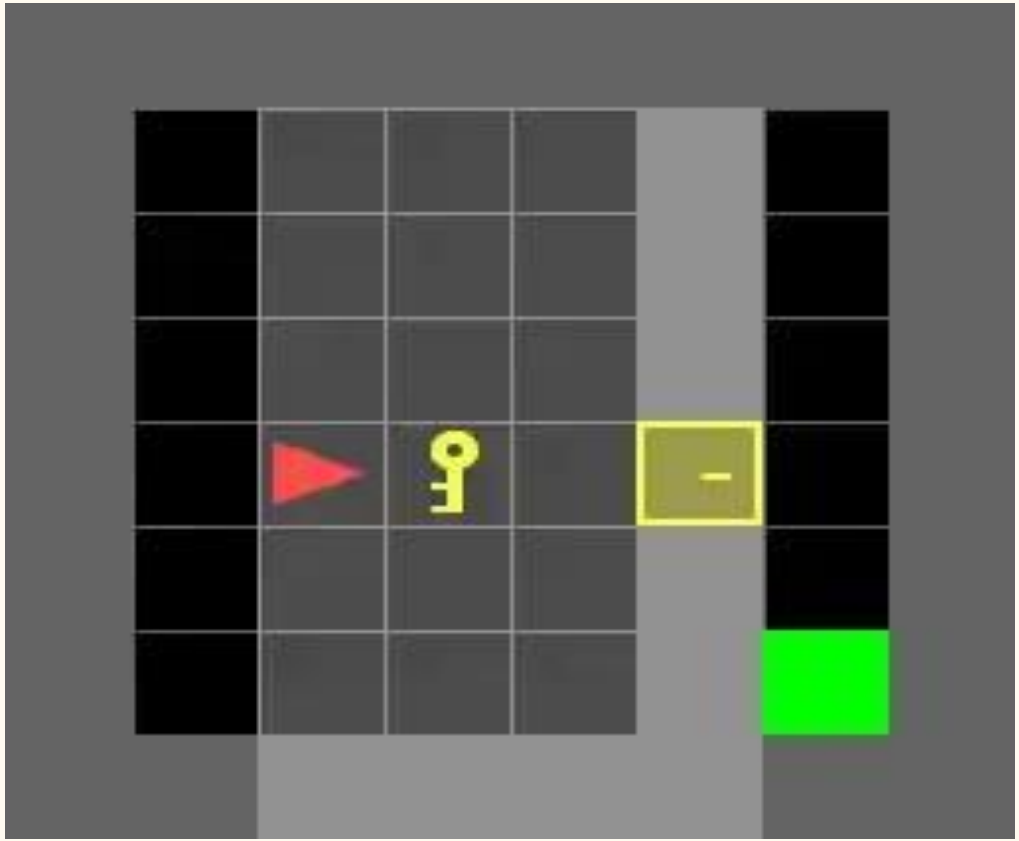
Evaluation Statistics:

```
pygame 2.3.0 (SDL 2.24.2, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
Device: cuda

Environments loaded

Agent loaded

F 2501.0 | FPS 207 | D 12 | R:μσmM 19.00 0.00 19.00 19.00 | F:μσmM 25.0 14.8 10.0 105.0

10 worst episodes:
- episode 0: R=19.0, F=10.0
- episode 1: R=19.0, F=13.0
- episode 2: R=19.0, F=16.0
- episode 3: R=19.0, F=18.0
- episode 4: R=19.0, F=20.0
- episode 5: R=19.0, F=22.0
- episode 6: R=19.0, F=13.0
- episode 7: R=19.0, F=24.0
- episode 8: R=19.0, F=26.0
- episode 9: R=19.0, F=27.0
```

Model Architecture:

```
ACModel(
  (image_conv): Sequential(
    (0): Conv2d(3, 16, kernel_size=(2, 2), stride=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(16, 32, kernel_size=(2, 2), stride=(1, 1))
    (4): ReLU()
    (5): Conv2d(32, 64, kernel_size=(2, 2), stride=(1, 1))
    (6): ReLU()
  )
  (actor): Sequential(
    (0): Linear(in_features=40000, out_features=64, bias=True)
    (1): Tanh()
    (2): Linear(in_features=64, out_features=5, bias=True)
  )
  (critic): Sequential(
    (0): Linear(in_features=40000, out_features=64, bias=True)
    (1): Tanh()
    (2): Linear(in_features=64, out_features=1, bias=True)
  )
)
```

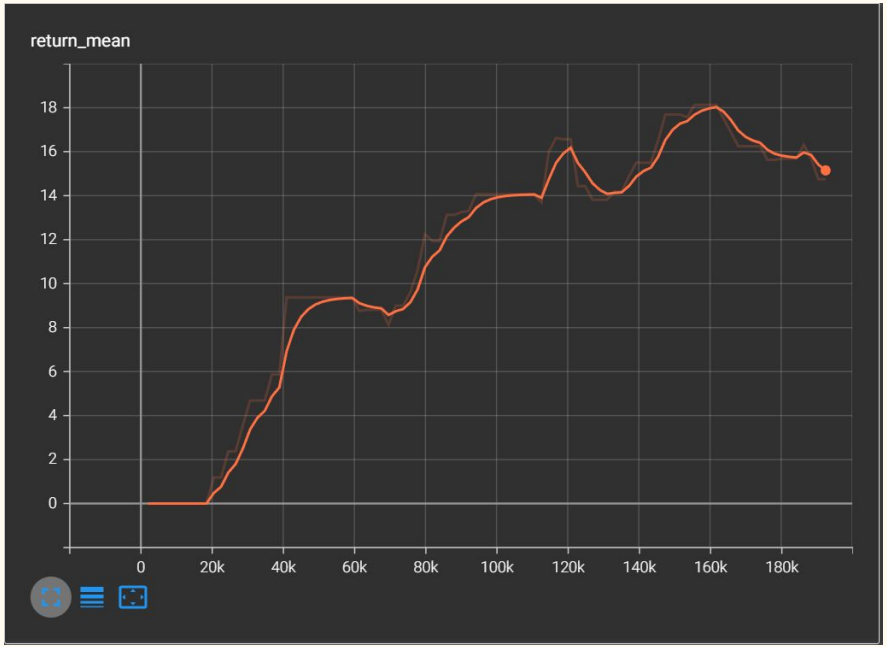
Agent Performance Video:



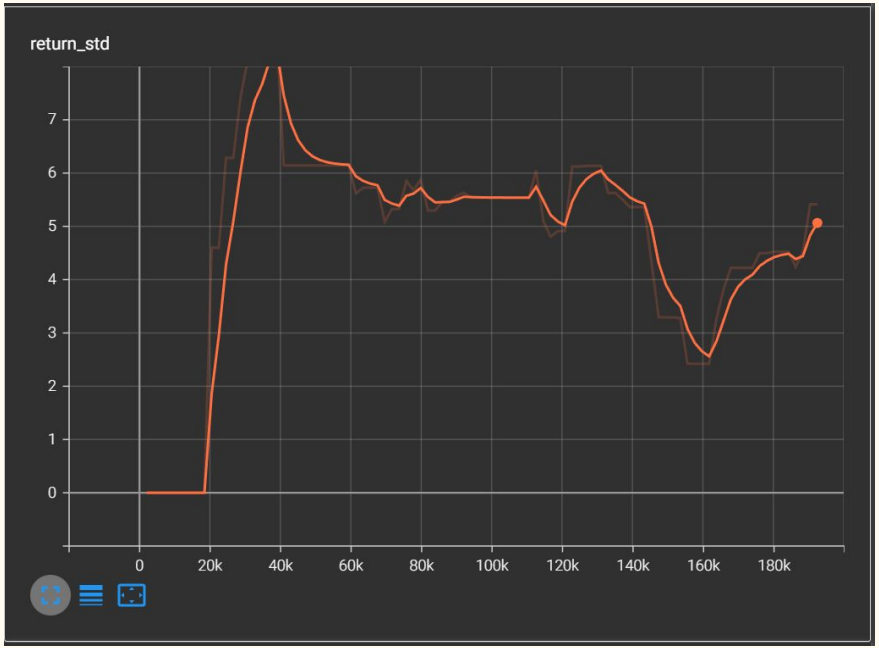
Minigrid-Door Key- 16X16-v0 Environment

Training Statistics:

Return Mean

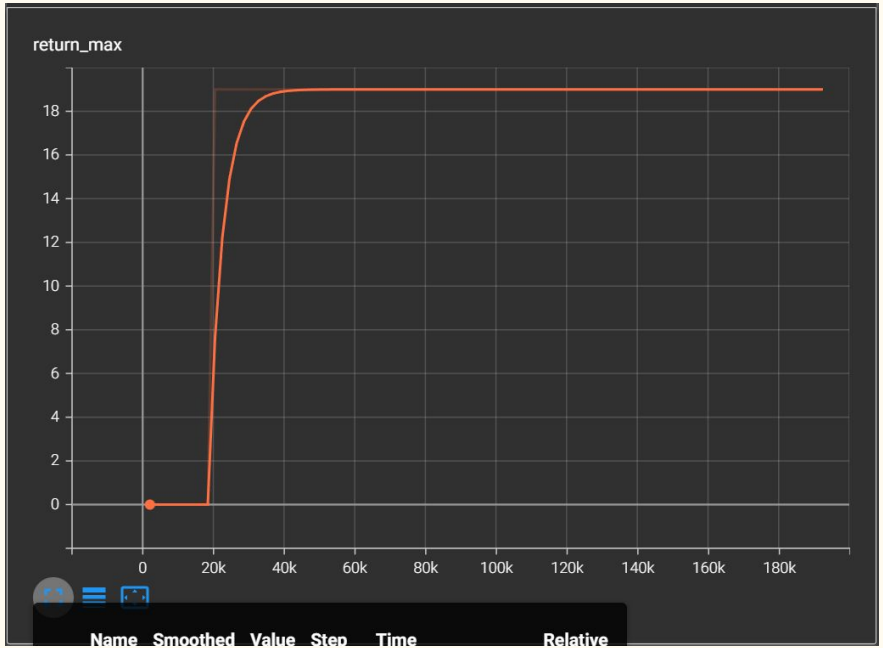


Return Std

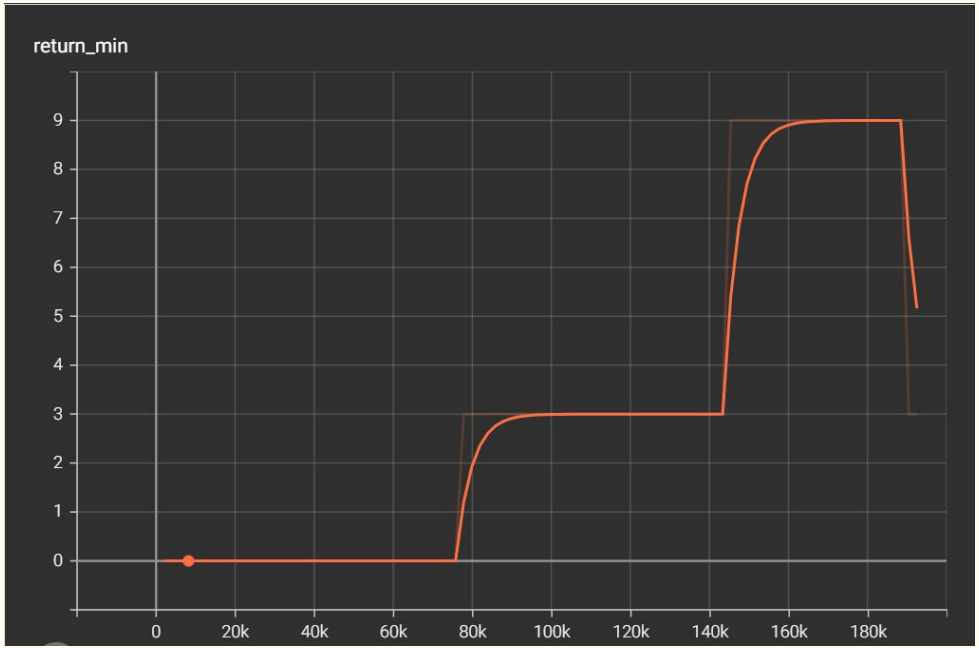


Training Statistics:

Return Max



Return Min



Evaluation Statistics:

```
%matplotlib inline
!python -m scripts.evaluate --env MiniGrid-DoorKey-16x16-v0 --model new

pygame 2.3.0 (SDL 2.24.2, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
Device: cuda

Environments loaded

Agent loaded

F 164160.0 | FPS 433 | D 378 | R:μσmM 14.71 5.53 3.00 19.00 | F:μσmM 1641.6 919.5 27.0 2560.0

10 worst episodes:
- episode 20: R=3.0, F=2560.0
- episode 33: R=3.0, F=2560.0
- episode 42: R=3.0, F=2560.0
- episode 48: R=3.0, F=2560.0
- episode 54: R=3.0, F=2560.0
- episode 72: R=3.0, F=2560.0
- episode 85: R=6.0, F=2560.0
- episode 16: R=8.0, F=2560.0
- episode 39: R=8.0, F=2560.0
- episode 41: R=8.0, F=2560.0
```

Model Architecture:

```
Observations preprocessor loaded
Model loaded

ACModel(
  (image_conv): Sequential(
    (0): Conv2d(3, 8, kernel_size=(2, 2), stride=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(8, 16, kernel_size=(2, 2), stride=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(16, 32, kernel_size=(2, 2), stride=(1, 1))
    (7): ReLU()
    (8): Conv2d(32, 64, kernel_size=(2, 2), stride=(1, 1))
    (9): ReLU()
  )
  (memory_rnn): LSTMCell(7744, 7744)
  (actor): Sequential(
    (0): Linear(in_features=7744, out_features=64, bias=True)
    (1): Tanh()
    (2): Linear(in_features=64, out_features=5, bias=True)
  )
  (critic): Sequential(
    (0): Linear(in_features=7744, out_features=64, bias=True)
    (1): Tanh()
    (2): Linear(in_features=64, out_features=1, bias=True)
  )
)
```

FUTURE DIRECTIONS

Findings and Directions:

Found that the use of the Pixel Observation wrapper greatly helped in reaching the convergence of the object and making it easy for CNN's to learn to extract meaningful representations from the view.

Found that the use of giving -0.10 and tuning the discount factor helped in reaching convergence faster of the algorithm.

Using Memory in the form of LSTM significantly improved convergence per update but the training times and GPU RAM utilization was increased exponentially.

Need to perform Hyperparameter tuning for getting the best reward values assigned to each milestone using frameworks such as Optuna.

Need to explore more architecture frameworks in CNN and ways to explicitly encode the current next milestone [By training three separate models conditionally].

THANK YOU!