Soft Actor-Critic Agent in MineRL

Jacob Santoni, Liam Orr, & Rohith Reddy

CSE 410: Reinforcement Learning

University at Buffalo

December 8th, 2020

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Topics for Discussion

- Project Description
- Background
- Implementation
- Demo
- Results
- Key Observations / Summary
- Thank you Page

Project Description

- We attempted to solve **MineRLNavigateDense-v0**, an environment from MineRL
- Trying to navigate from a spawn point to another point farther away
- Part of MineRL, a competition to develop **sample efficient** algorithms

Observation Space

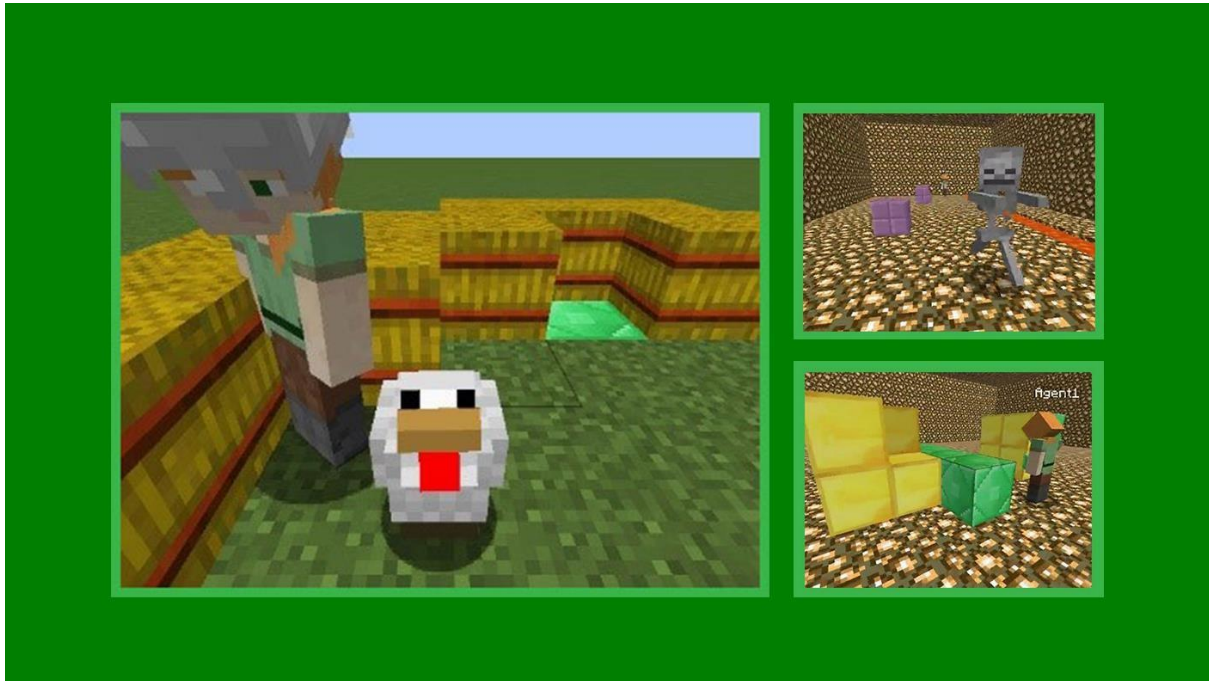
```
Dict({
    "compassAngle": "Box(low=-180.0, high=180.0, shape=())",
    "inventory": {
            "dirt": "Box(low=0, high=2304, shape=())"
    },
    "pov": "Box(low=0, high=255, shape=(64, 64, 3))"
})
```

Action Space

```
Dict({
    "attack": "Discrete(2)",
    "back": "Discrete(2)",
    "camera": "Box(low=-180.0, high=180.0, shape=(2,))",
    "forward": "Discrete(2)",
    "jump": "Discrete(2)",
    "left": "Discrete(2)",
    "place": "Enum(dirt,none)",
    "right": "Discrete(2)",
    "sneak": "Discrete(2)",
    "sprint": "Discrete(2)"
})
```

Background: Malmö, Minecraft, and MineRL

- **Minecraft**: It’s a game you all probably know well
- **Malmö**: Reinforcement learning backend for Minecraft, by Microsoft
- **MineRL**: A competition framework built on top of Malmö which gives us a set of challenge environments and user generated data



https://www.microsoft.com/en-us/research/uploads/prod/2018/11/MalmoCompetition_AI_Site_11_2018_1400x788.png

- Overall goal of the MineRL competition is to produce an agent capable of mining diamonds, displaying **sample efficiency**

Background: MineRLNavigateDense-v0

- Environment from MineRL
- Agent has to navigate from a spawn point to another point on the map
- Has a compass which always points at the goal
- 3x64x64 representation of the POV of the agent

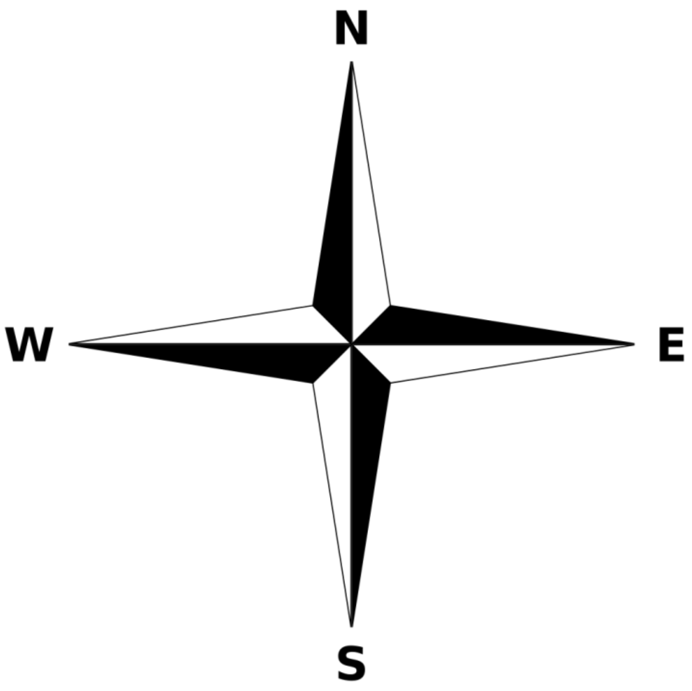
- Dense version gives small positive rewards for getting closer to the goal and negative for going farther away

Implementation: Soft Actor-Critic

- Spaces

Observation Space=Compass Angle + POV

Action Space = Yaw (Continuous)



- Objective - **Maximize Expected Return & Maximize Entropy**
- Exploration vs. Exploitation - Controlled by alpha parameter which scales entropy
- Large alpha -> Large entropy -> Large exploration

Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
Compute targets for the Q functions:

$$y(r, s', d) = r + \gamma(1-d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') \boxed{- \alpha \log \pi_\theta(\tilde{a}'|s')}\right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s,a) - y(r,s',d)\right)^2 \qquad \text{for } i = 1, 2$$

Update policy by one step of gradient ascent using

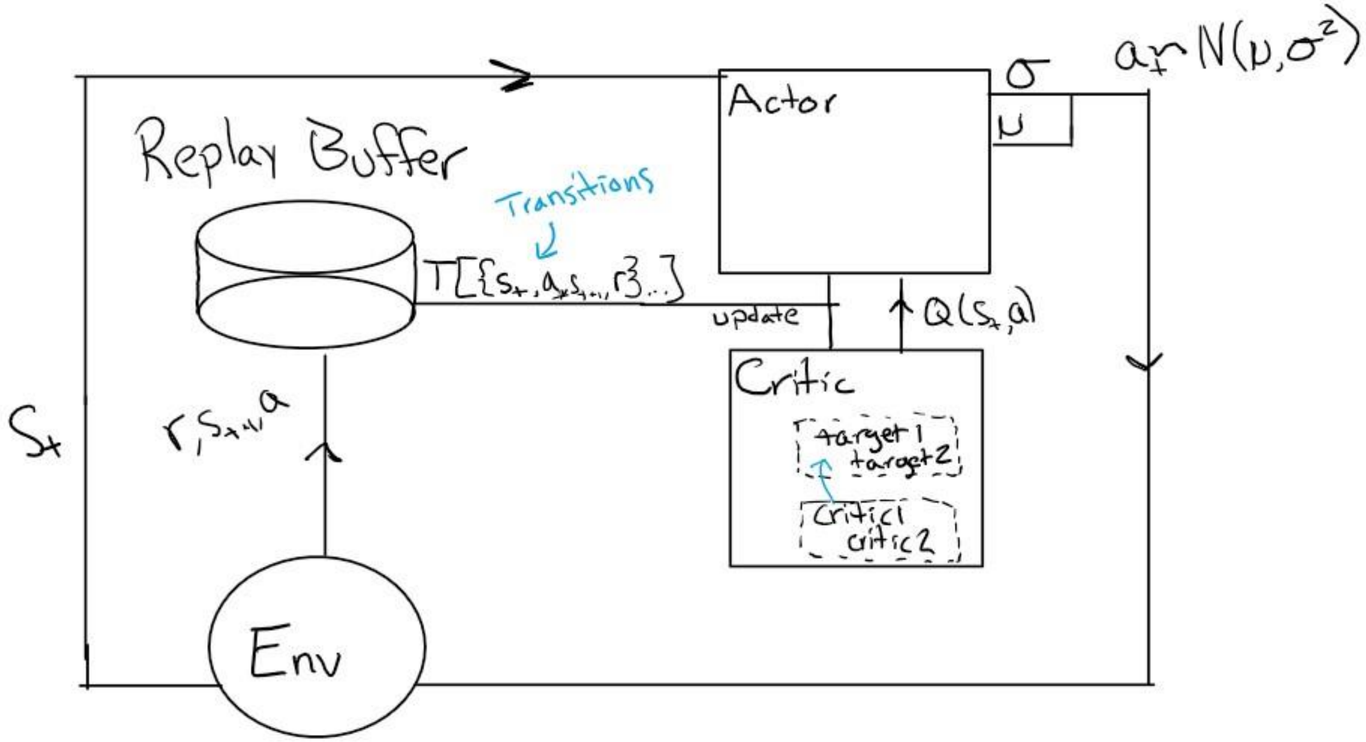
$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \boxed{\alpha \log \pi_\theta\left(\tilde{a}_\theta(s) | s\right)}\right),$$

where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.
Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho\phi_{\text{targ},i} + (1-\rho)\phi_i \qquad \text{for } i = 1, 2$$

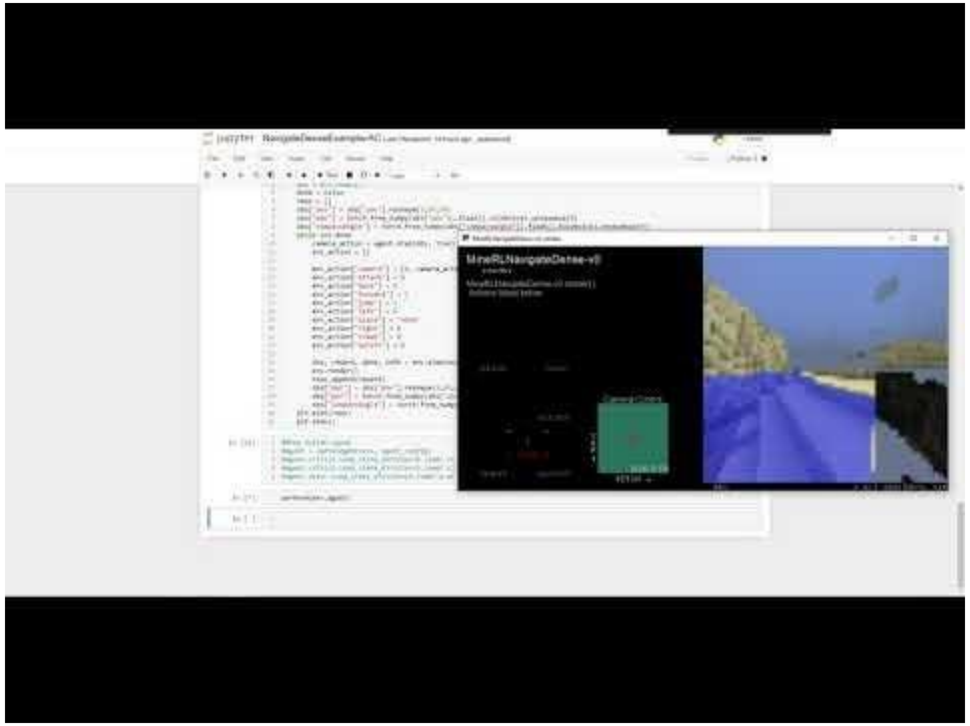
Implementation Details

- **Replay Buffer**
  - Improve Sample Efficiency
- **Double Q-Network**
  - Makes Learning More Stable
- **Freeze Target Critics**
  - Breaking Critic/Target Correlations
- **Stochastic Policy**

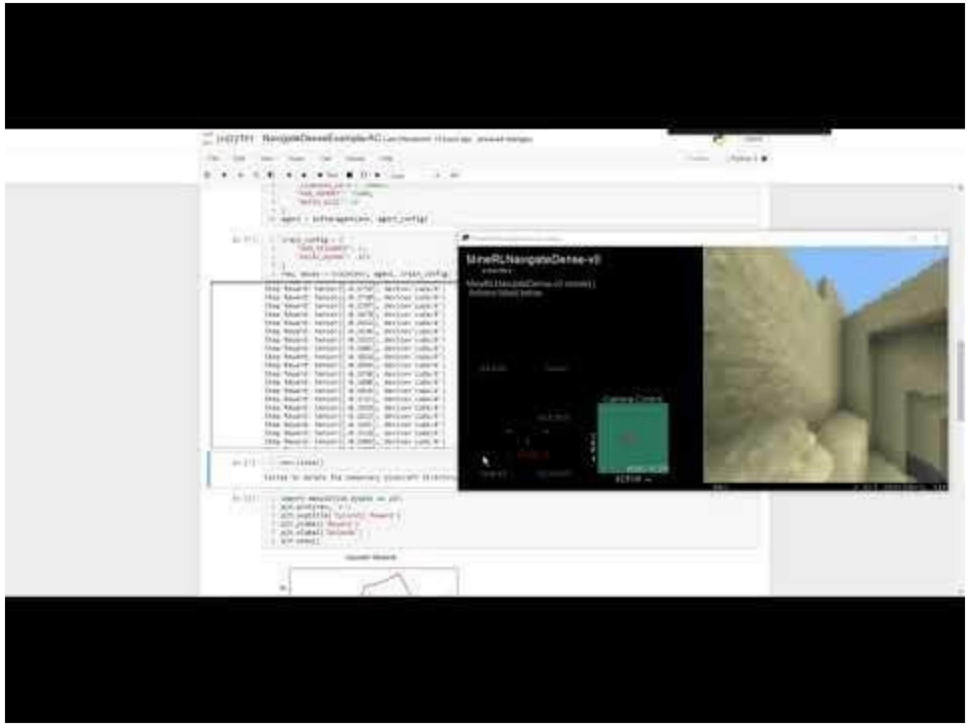


Demo (Using POV)

[Deterministic Testing](#) [Trapped Agent During Evaluation](#)



https://youtu.be/s7ab244Iwp0



https://youtu.be/bBk45epnjn8

Results Contextualized

- When training our agent, we found that it would start off likely achieving high reward or even **finishing the episode successfully**, but after many episodes it would average out to good (i.e. positive) but less than ideal reward
- Due to the complexity of the environment and the relatively small training, it is hard to assess if this is due to the randomness involved or if the agent is forgetting or ignoring the optimal policy.

Results Benchmark

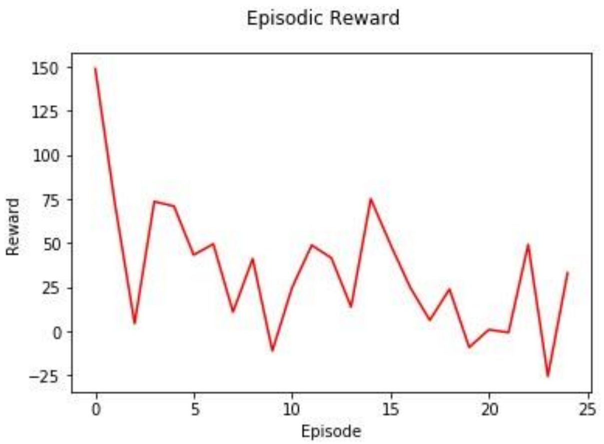
	Treechop	Navigate (S)	Navigate (D)
DQN (Minh et al., 2015[13])	3.73 ± 0.61	0.00 ± 0.00	55.59 ± 11.38
A2C (Minh et al. 2016[14])	2.61 ± 0.50	0.00 ± 0.00	-0.97 ± 3.23
Behavioral Cloning	**43.9 ± 31.46**	4.23 ± 4.15	5.57 ± 6.00
PreDQN	4.16 ± 0.82	6.00 ± **4.65**	**94.96 ± 13.42**
Human	64.00 ± 0.00	100.00 ± 0.00	164.00 ± 0.00
Random	3.81 ± 0.57	1.00 ± 1.95	-4.37 ± 5.10

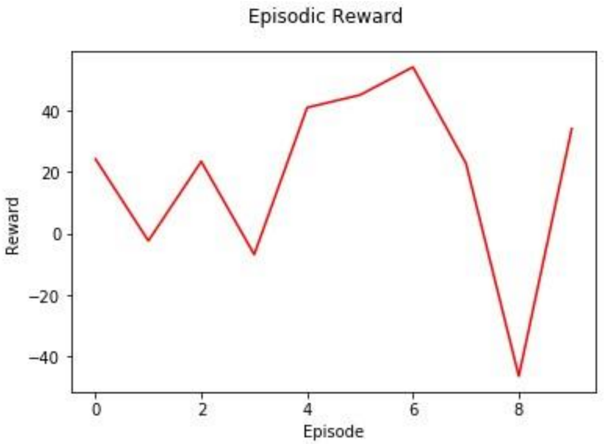
Table 2: Results in Treechop, Navigate (S)parse, and Navigate (D)ense, over the best 100 contiguous episodes. ± denotes standard deviation. Note: humans achieve the maximum score for all environments shown.

Results

Without POV Observation With POV Observation

	No POV	POV
Average Reward	34	19
Number Episodes	25	10
Best Episode	151	54





Episode Renders

-18.1 Reward

https://youtu.be/DcEFFxwzV44

8.9 Reward

https://youtu.be/jQ5ZU--qJPA

148.4 Reward

https://youtu.be/0K3tUOaLSJ0

Key Observations / Summary

- Minecraft has an incredibly large Observation Space
  - POV: Less sample efficient, but **more robust agent**.
- Training = **Resource Intensive**

  With CNN ~10 minutes per episode

  Larger replay buffer -> more stable training

  Malmo rendering incomplete / crashes

- Hyper-Parameter Sensitive
- Ideas for Improvement

  Distributed Learning

Use MineRL Dataset

References

1. Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018, August 08). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. Retrieved December 08, 2020, from https://arxiv.org/abs/1801.01290

2. “Soft Actor-Critic¶.” Soft Actor-Critic - Spinning Up Documentation,
https://spinningup.openai.com/en/latest/algorithms/sac.html

Thank You!!!