Deep Reinforcement Learning in OpenAI gym

Course: Pattern Recognition

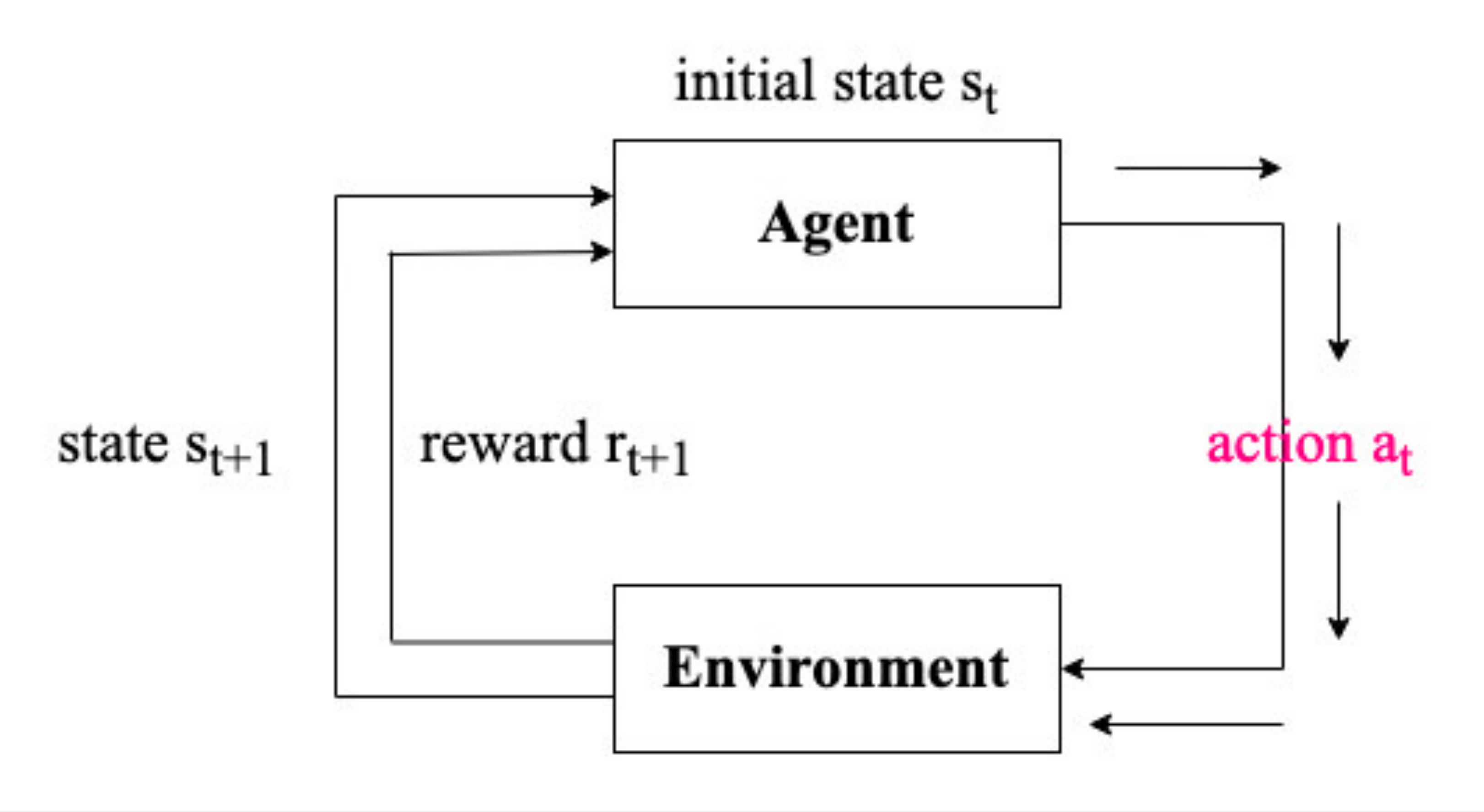
Submitted by Aishwarya Anilkumar

Paper reference: [1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

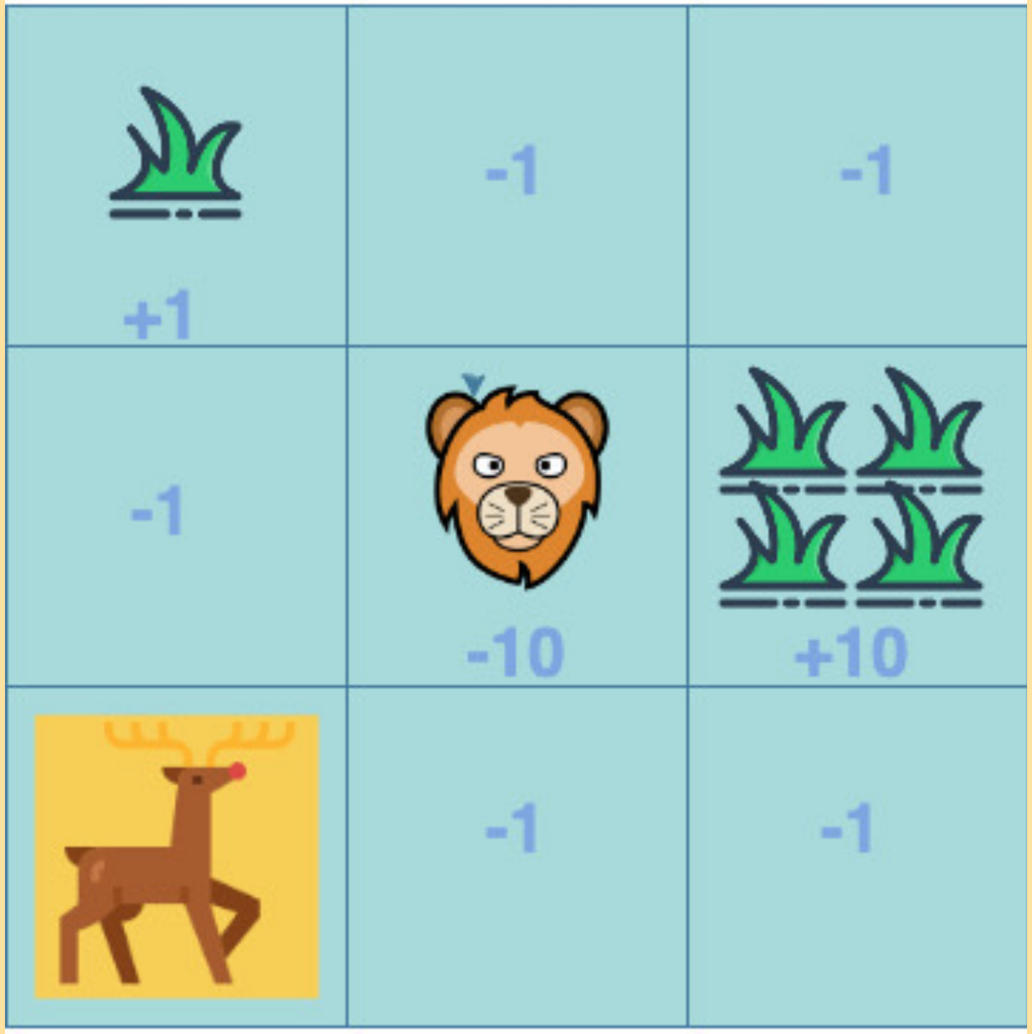
Basic Idea

Reinforcement Learning + Deep neural network

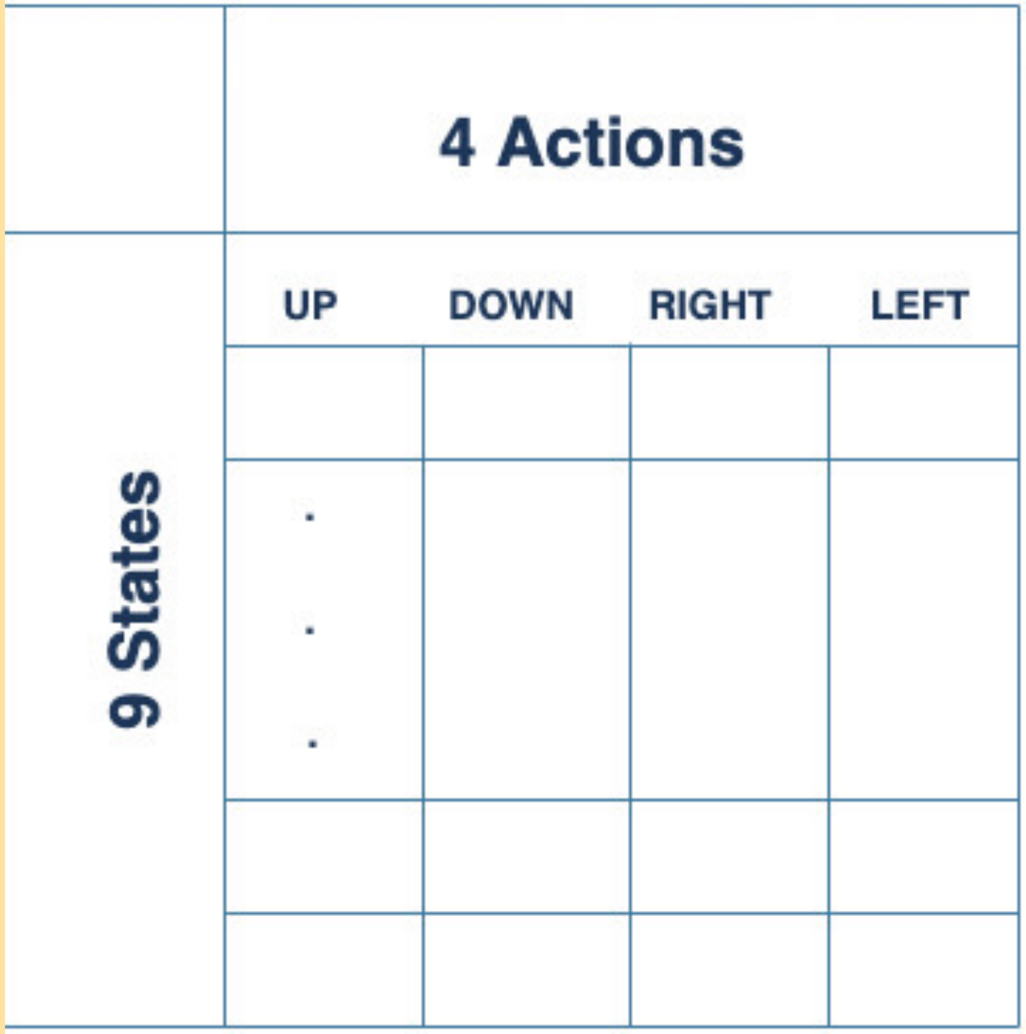
Reinforcement Learning



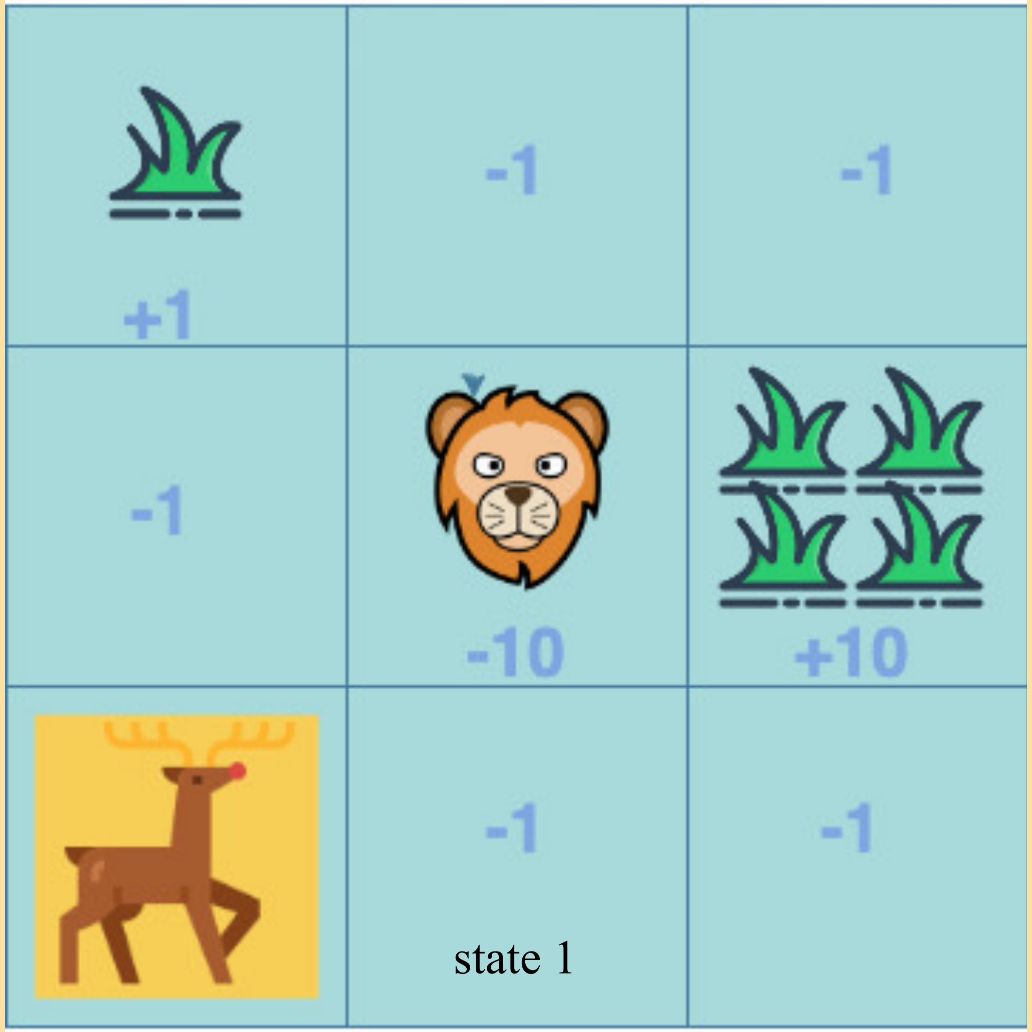
Agent Navigation



Q table



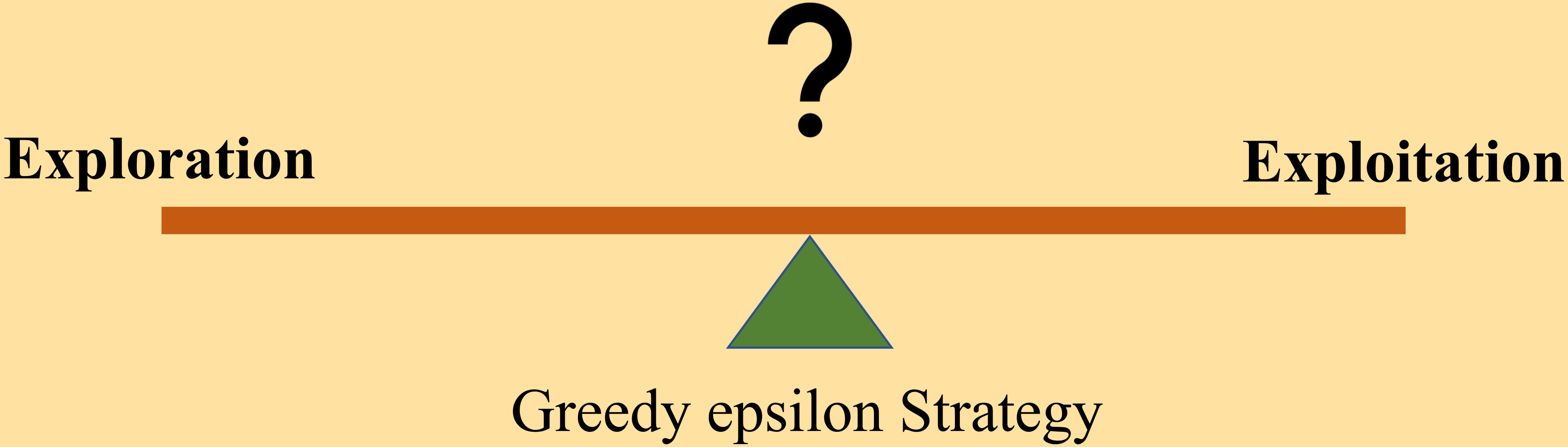
Agent Navigation



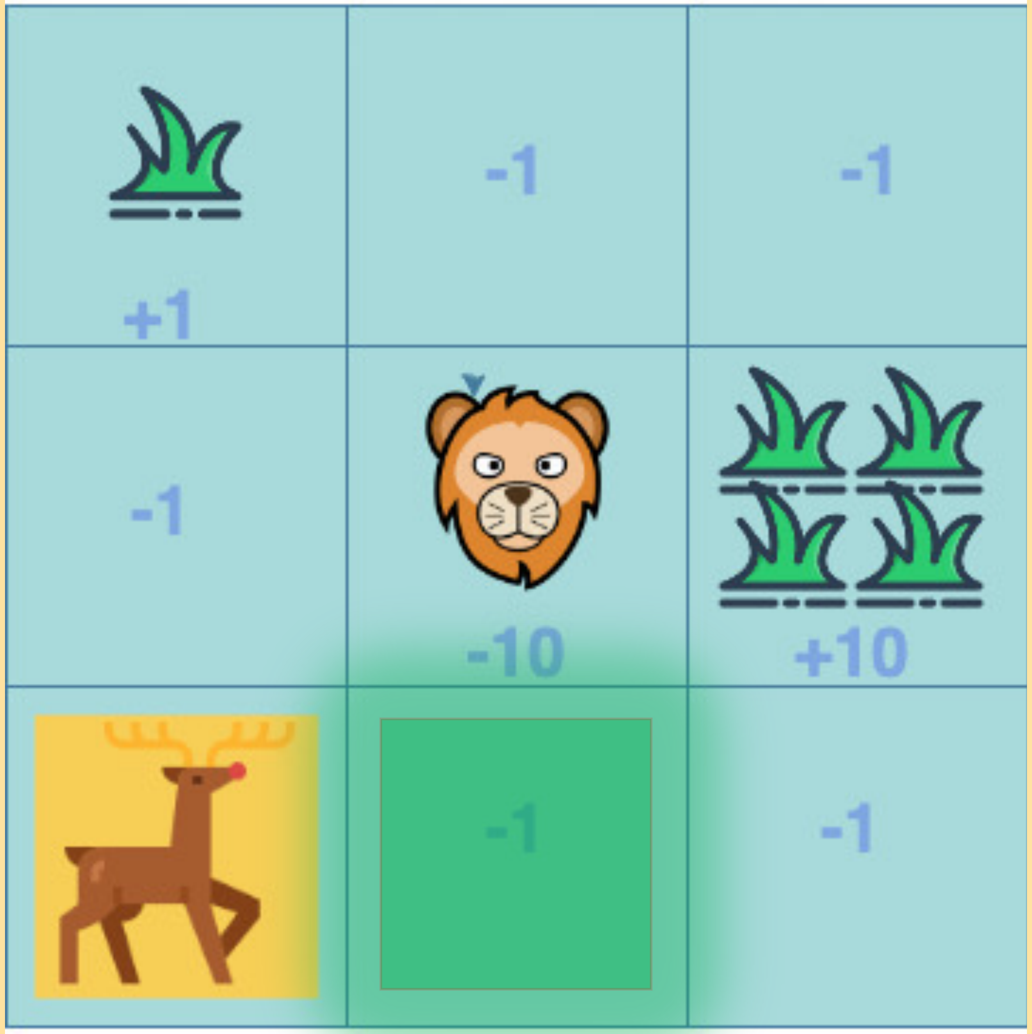
Q table

		4 Actions		
	UP	DOWN	RIGHT	LEFT
state 1			+10	
9 States	. . .			

Dilemma for the Agent



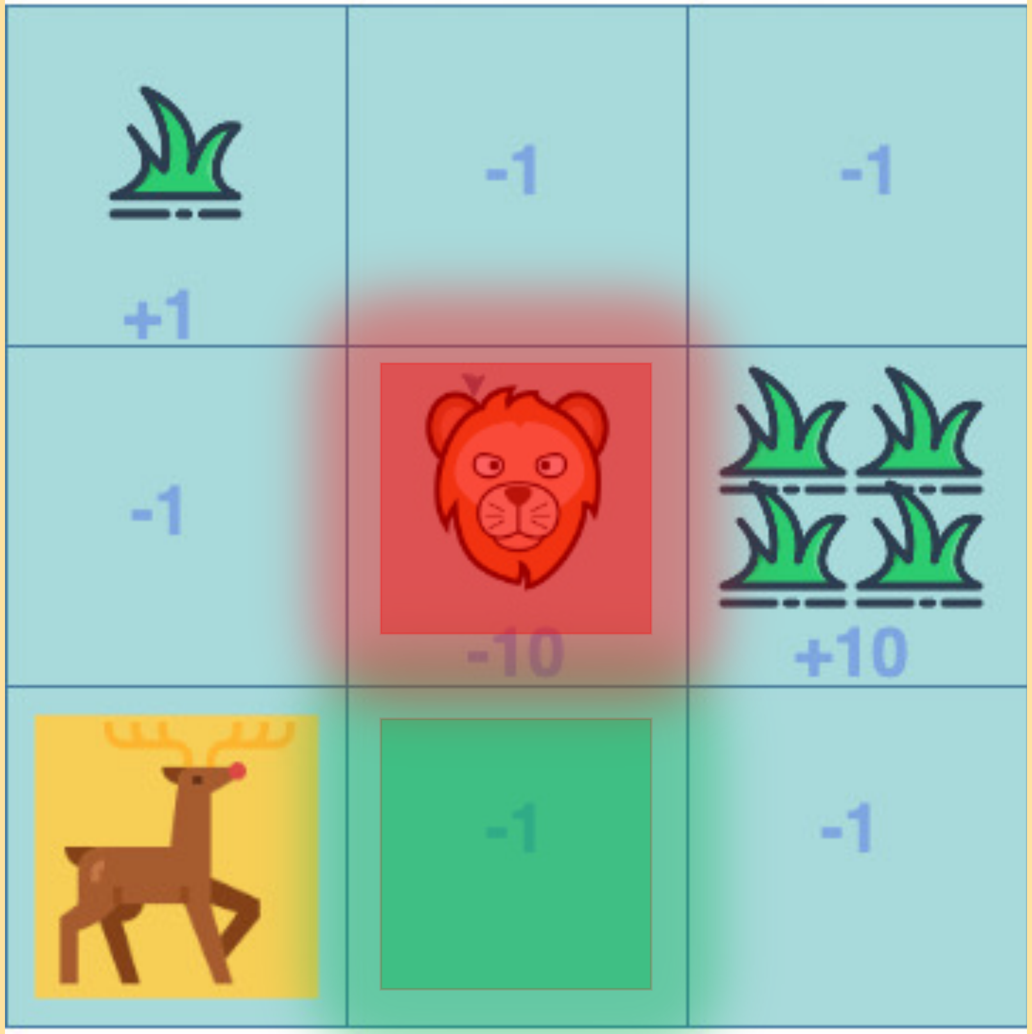
Agent Navigation



Q table

	4 Actions			
9 States	UP	DOWN	RIGHT	LEFT
			-1	
	. . .			

Agent Navigation



Q table

	4 Actions			
	UP	DOWN	RIGHT	LEFT
9 States			-1	
	. . .			
	-10			

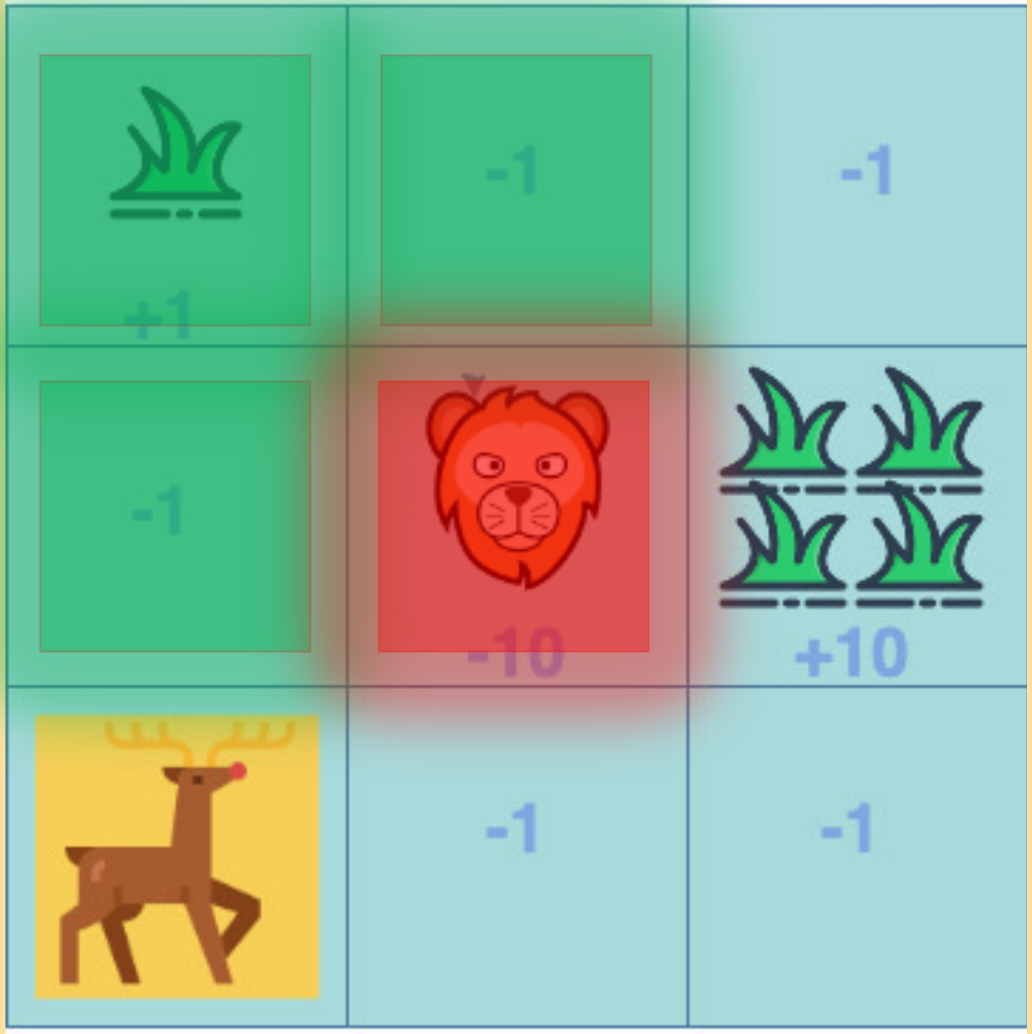
Discount = 1

Agent Navigation

Q table

	4 Actions			
9 States	UP	DOWN	RIGHT	LEFT
	-1			
	.			
	.			
	.			
	+1			

Agent Navigation



Q table

	4 Actions			
	UP	DOWN	RIGHT	LEFT
9 States	-1			
	. . .	-10	+1	
	+1			

Agent Navigation

Q table

9 States	4 Actions			
	UP	DOWN	RIGHT	LEFT
	-1			
	. . .		-1 -1	
	+1			
		+10		

Important concepts

- Markov property: The reward from s_t to s_{t+1} will only depend on St and s_{t+1}
- Q value update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[\boxed{R_{t+1} + \gamma \max_a Q(S_{t+1}, a)} - Q(S_t, A_t) \right]$$

- Policy: A strategy to navigate through the environment
- Target policy vs Running Policy

Formula reference: arXiv:1906.10025v1 [cs.LG] 24 Jun 2019

Architecture

The architecture of this project involves two models:

- 1) **Q** DNN
- 2) **Q_hat** DNN
- The DNN has total 2 fully connected linear layers:
- The final layer outputs "Action-values"(Being in a state $\mathbf{s_t}$, if we make action $\mathbf{a_t}$ how much will be the total reward)

Algorithm

For episode $= 1, M$ **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For

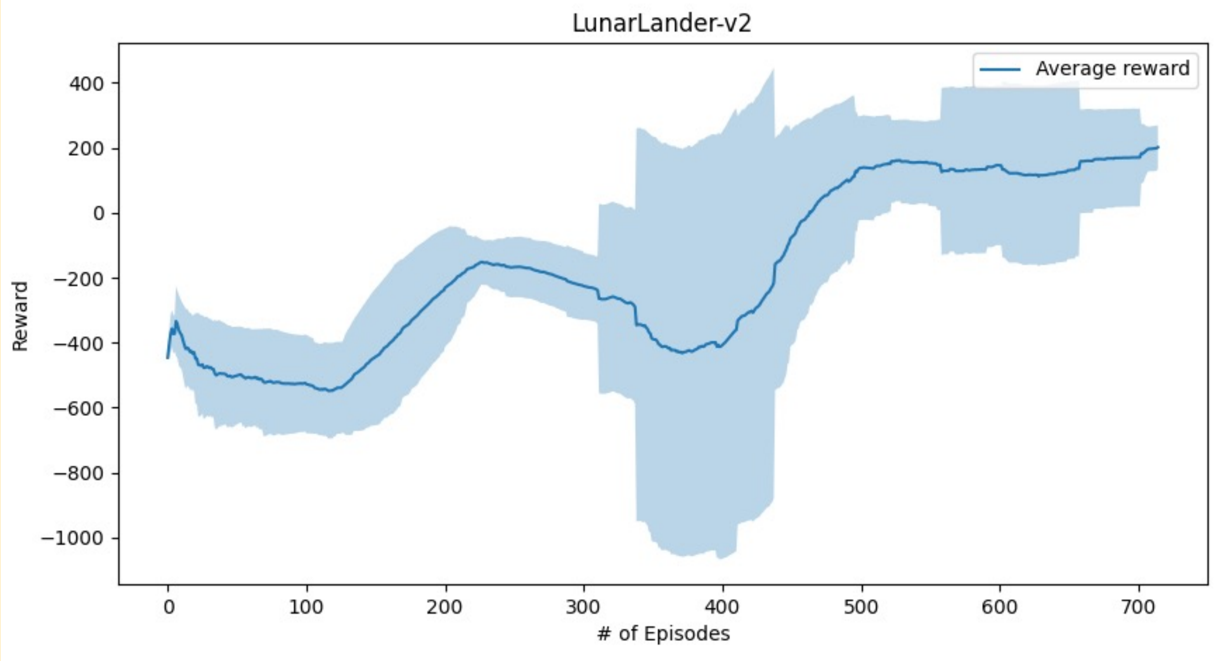
Paper reference: [1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

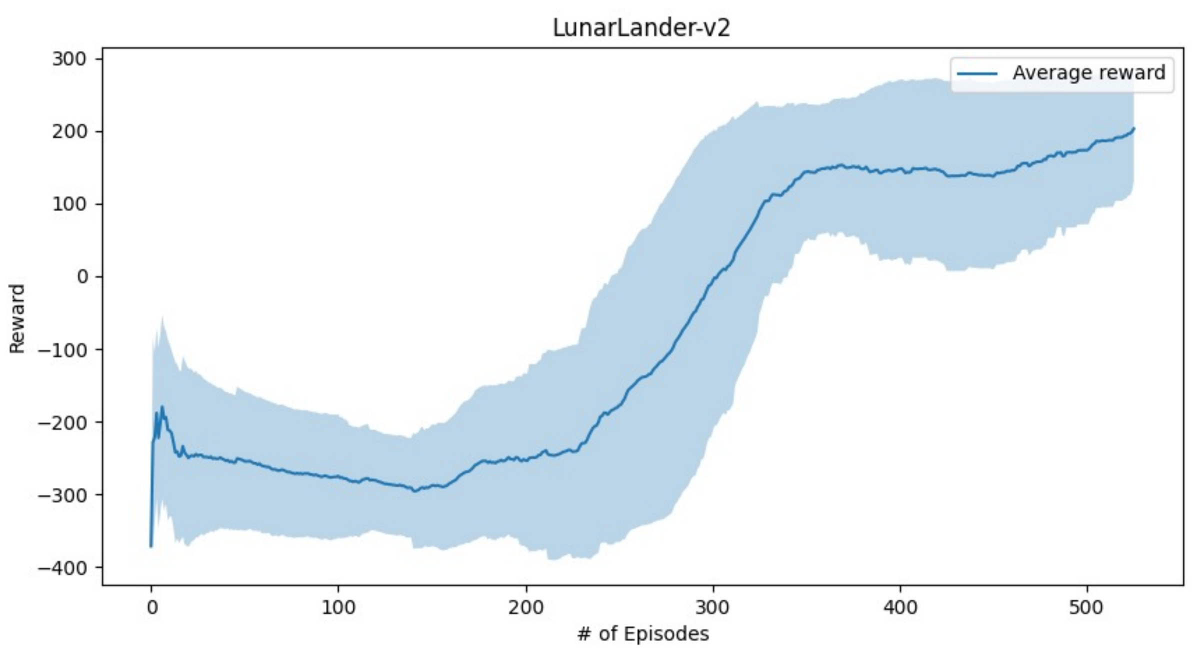
Result

Episode	Score
100	-586.05
200	-368.12
300	-255.17
400	223.26
500	255.14

Agent learning and improving

Result





Reward vs **Number of episodes plot**

Disadvantages

- Produces unsatisfactory results where data generation is expensive, since training the neural network requires huge amount of data
- Requires extensive iterations increasing the computational time cost
- Low reproducibility of same results for empirical observations

Conclusion

- This paper presented a deep learning model for reinforcement learning
- Demonstrated ability to master control policies using few pixels
- Introduces replay buffer concept
- Practical applications such as self driving cars, general AI (agent mastering multiple tasks) such as research by Dr David Silver, and Dr Peter Abbiel