Deep Reinforcement Learning in OpenAI gym

Course: Pattern Recognition

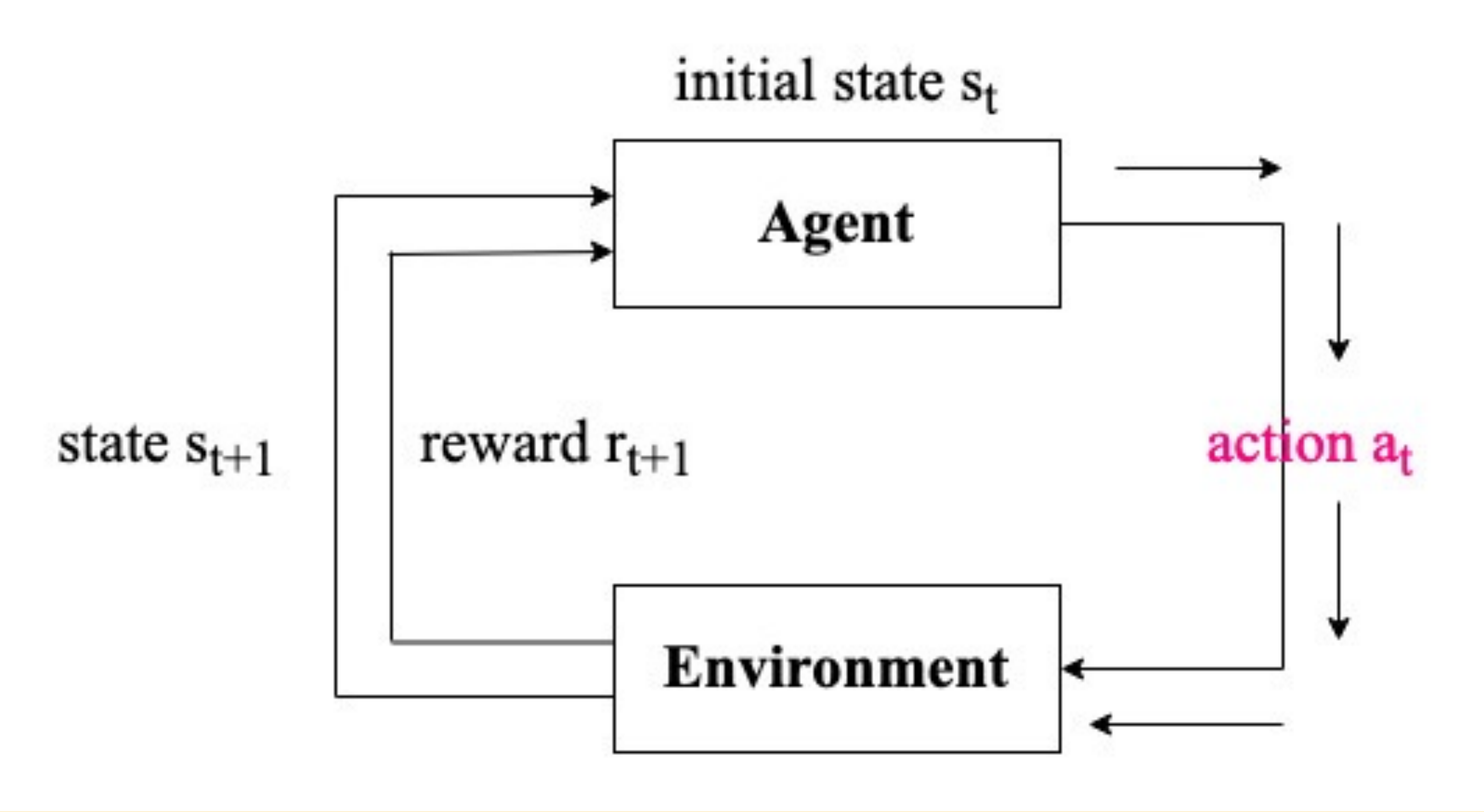
Submitted by Aishwarya Anilkumar

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.

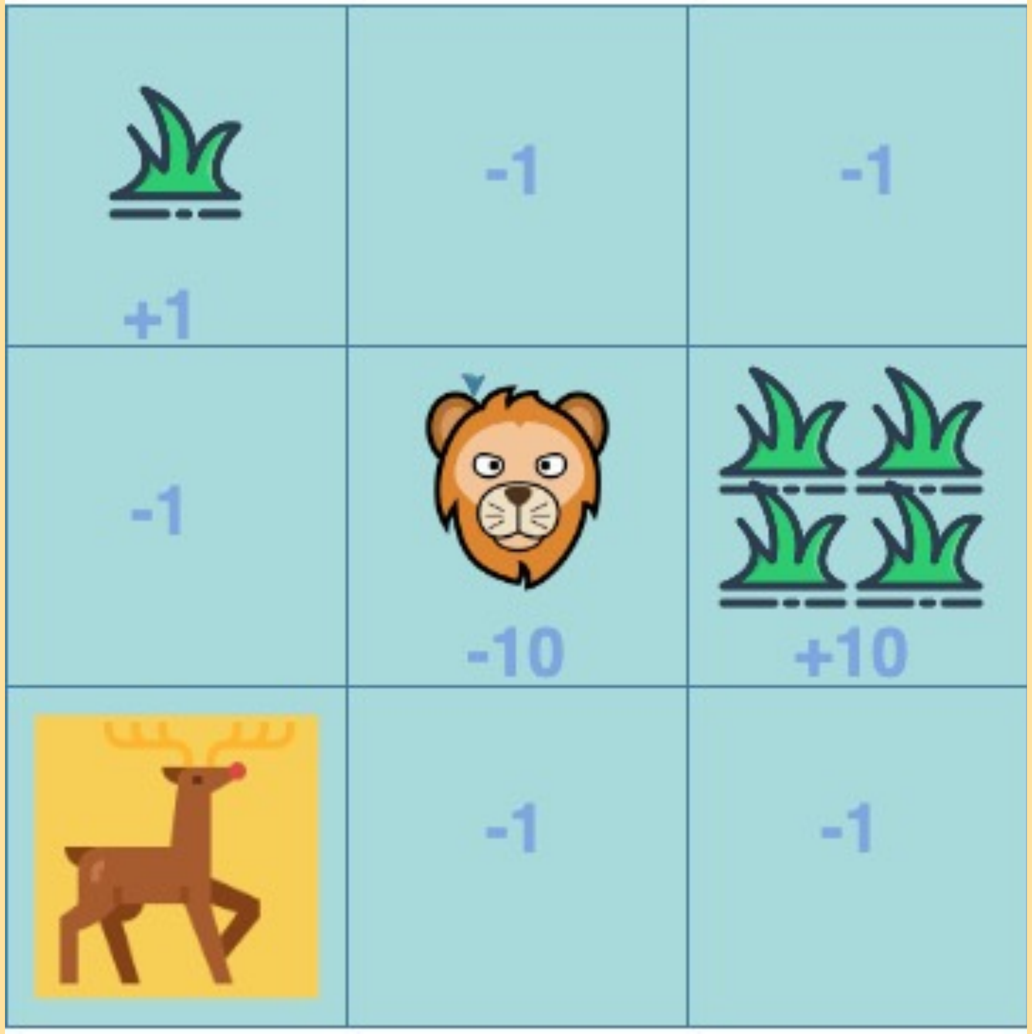
Basic Idea

Reinforcement Learning + Deep neural network

Reinforcement Learning



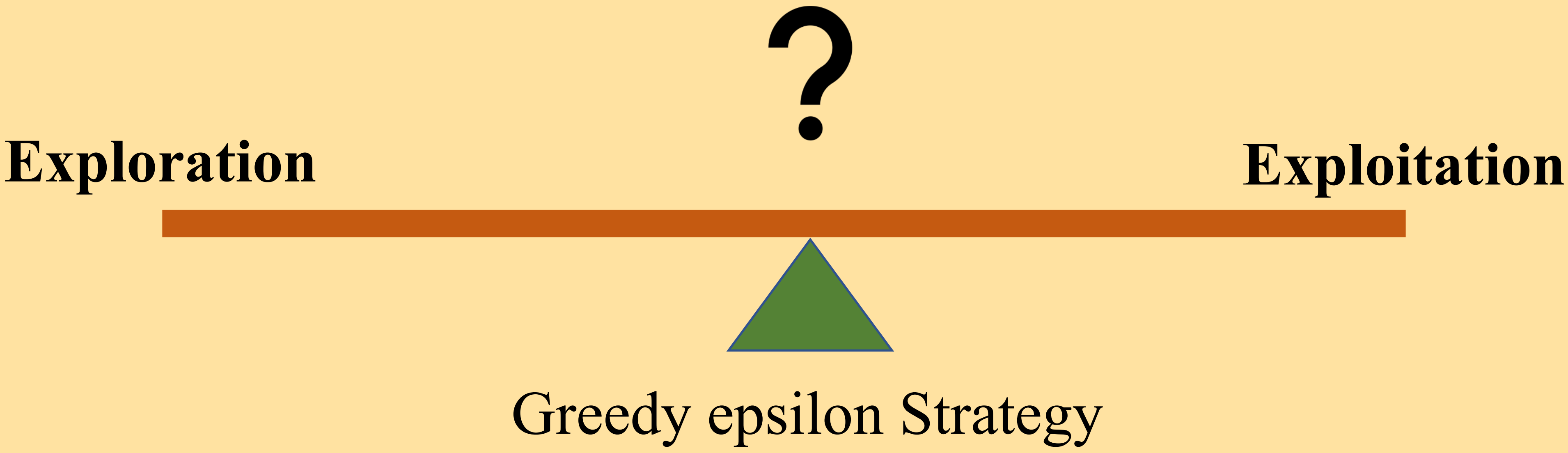
Agent Navigation



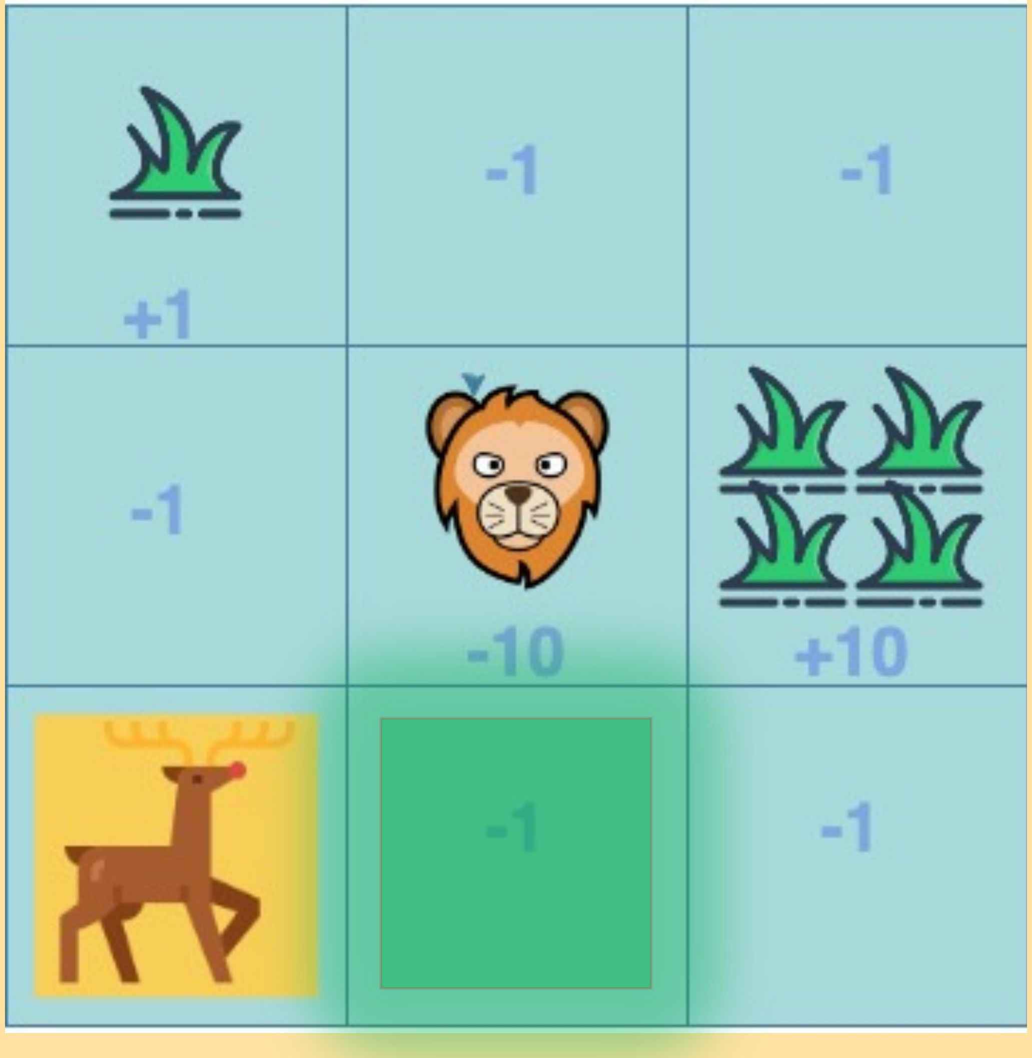
Q table

	4 Actions			
9 States	UP	DOWN	RIGHT	LEFT
	· · ·			

Dilemma for the Agent



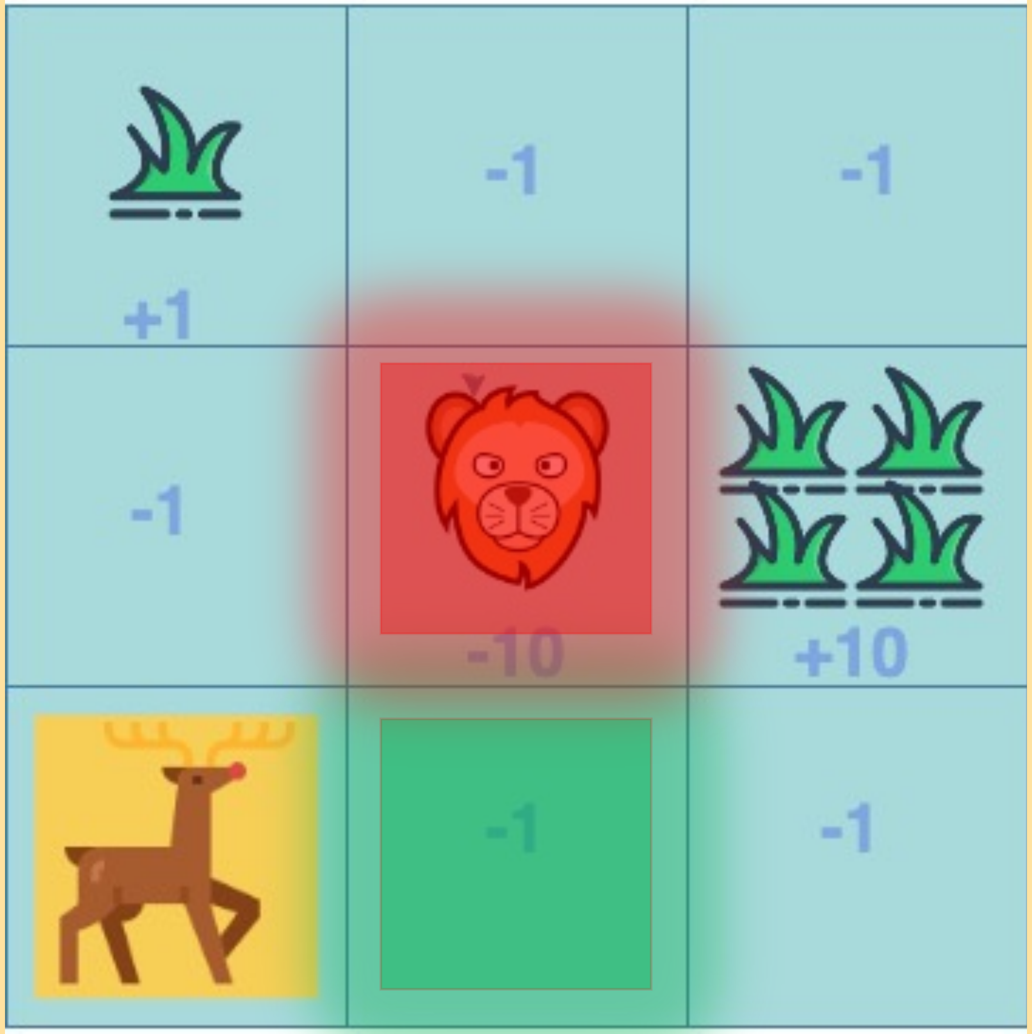
Agent Navigation



Q table

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

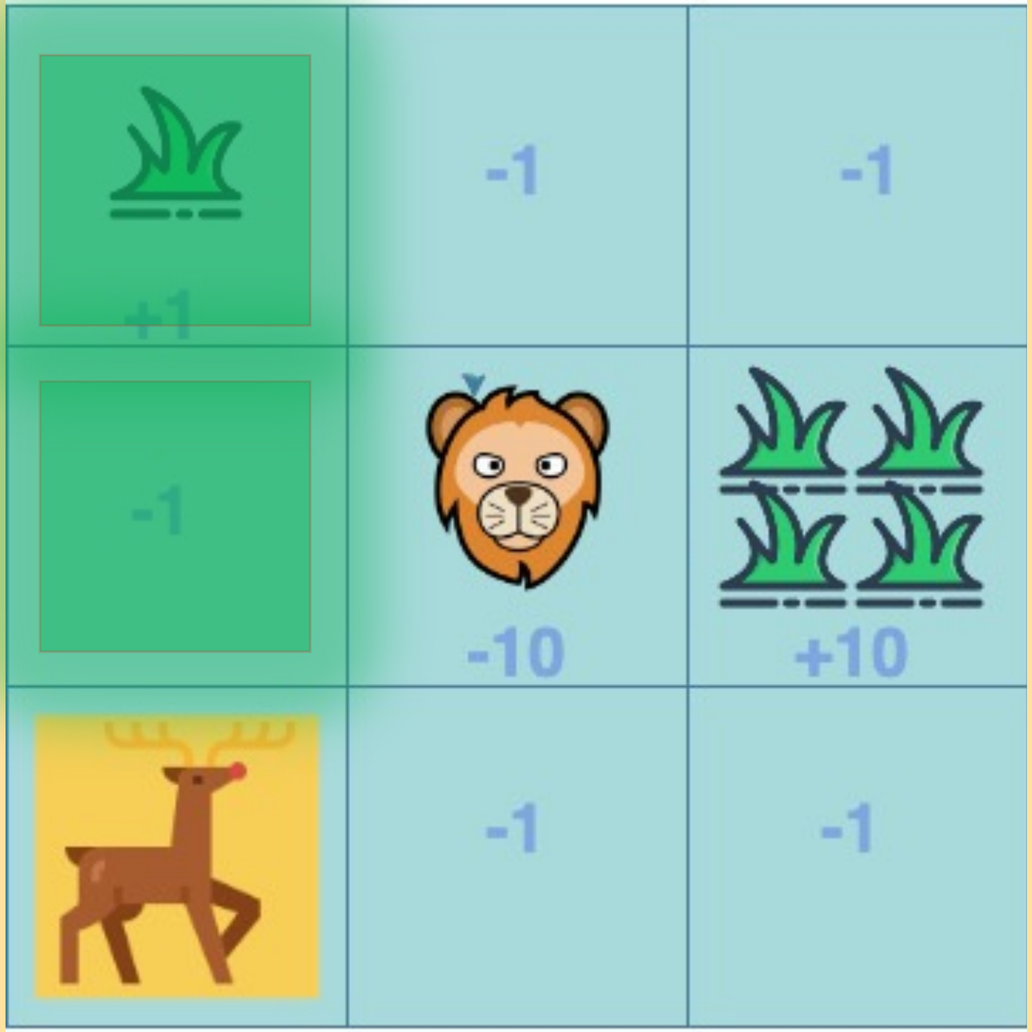
Agent Navigation



Q table

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

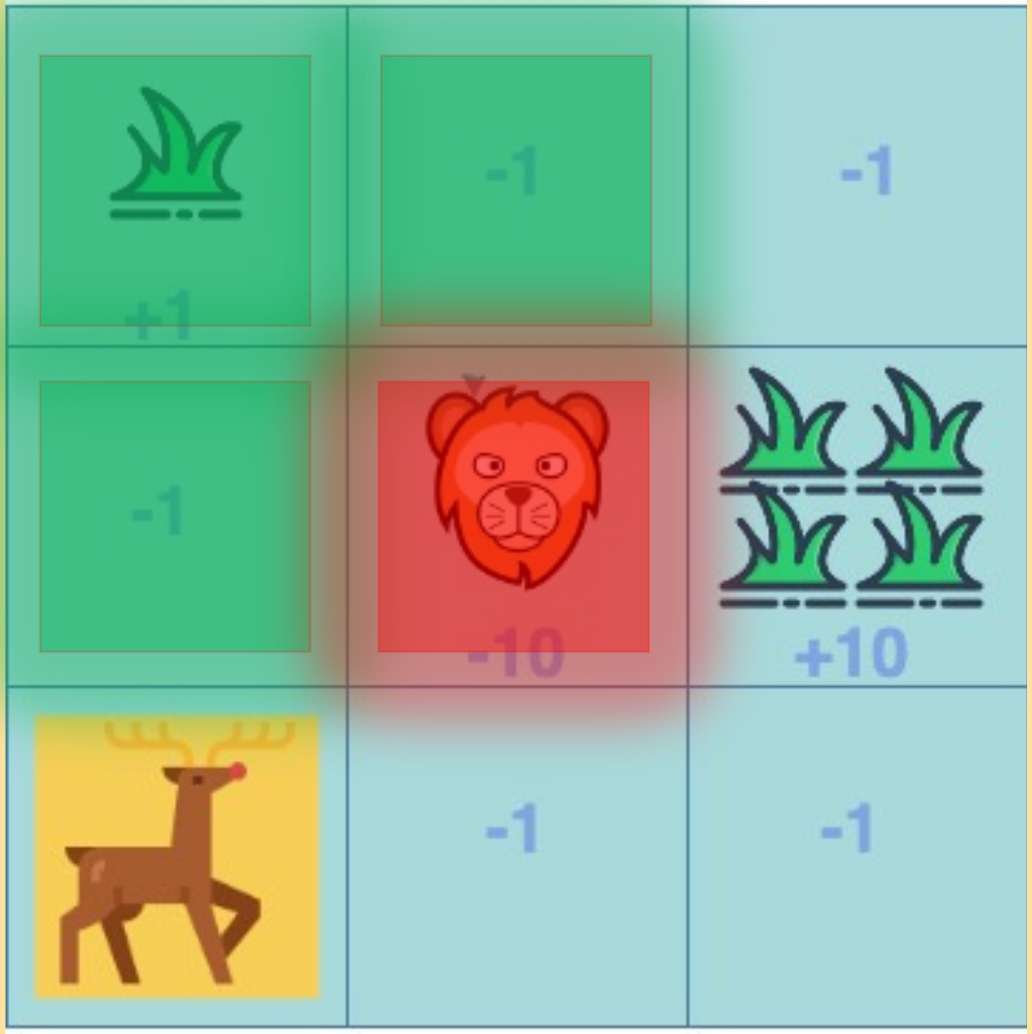
Agent Navigation



Q table

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

Agent Navigation



Q table

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

Agent Navigation

Q table

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

Important concepts

- Markov decision process: Independent of all previous interactions
- Q learning: $Q^*(s,a) \leftarrow Q^*(s,a) + \alpha_t \left[r' + (1 - \text{done})\gamma \max_{a'} Q^*(s',a') - Q^*(s,a) \right]$
- 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 (A convolutional Neural Network similar to one implemented in the paper referenced above for action-value function Q)
- 2) **Q_hat** DNN (similar model as Q DNN for target action-value function Q_hat)
  The CNN has total 6 layers:
  - 3 Convolutional 2D layers
  - 3 Dense 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 $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$

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

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

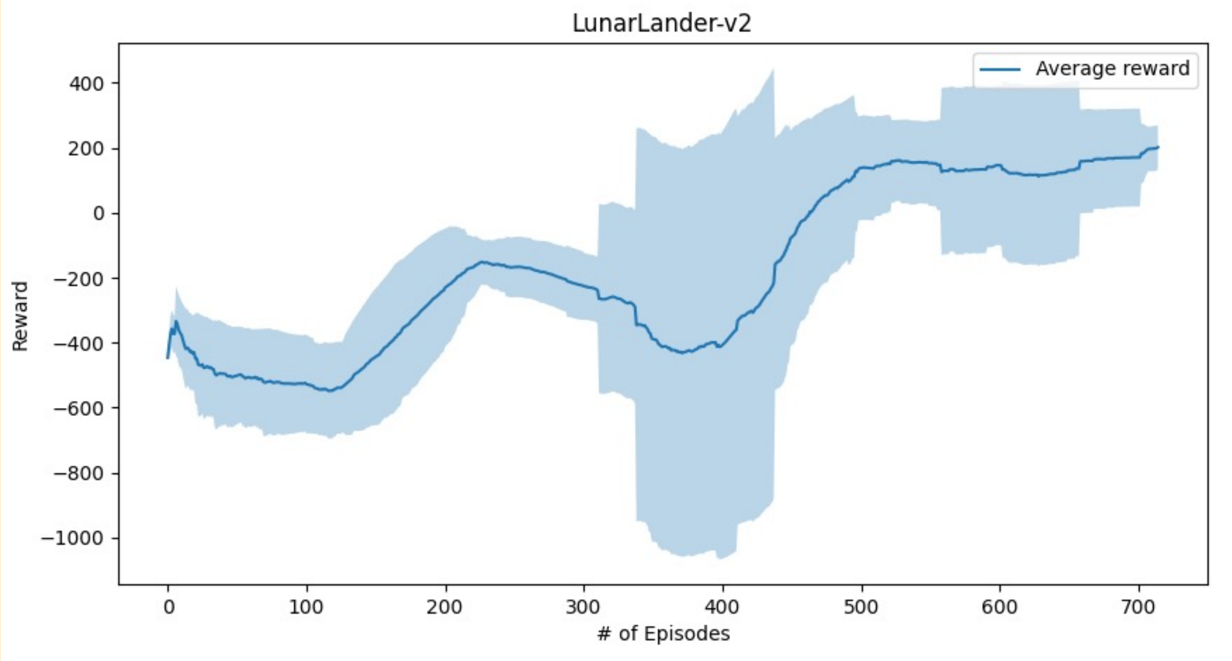
End For

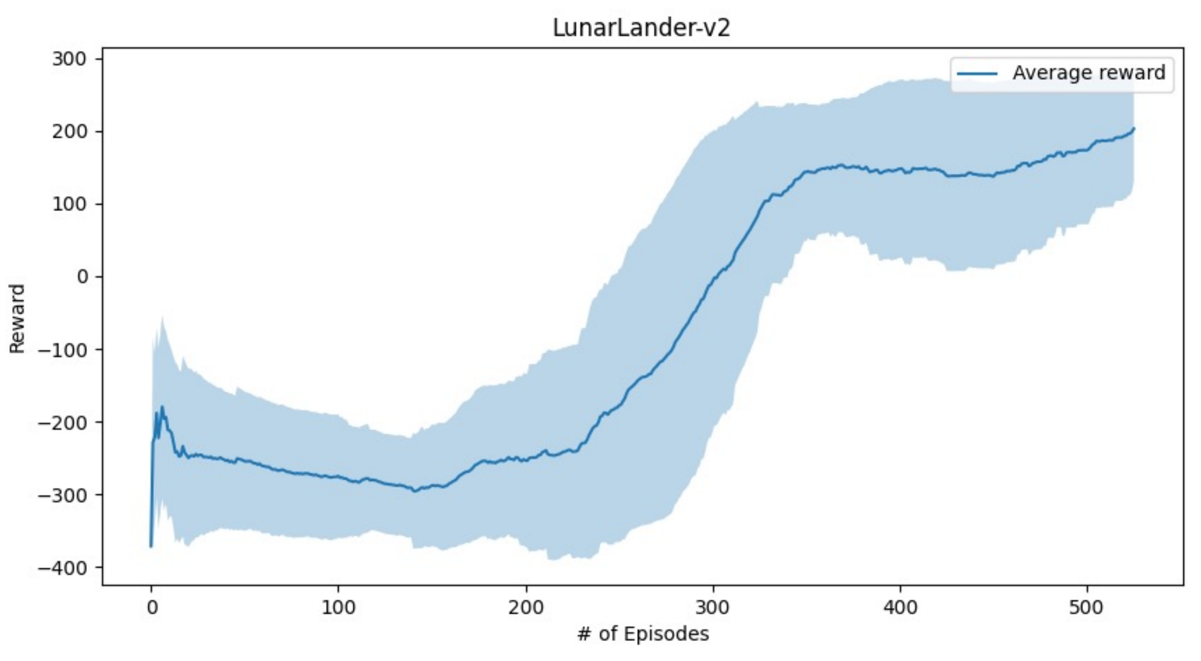
Paper reference: [1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Result

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