CSE 4/546: Reinforcement Learning

Self-driving cars

Team 23:
Arjun Ramesh Kaushik (kaushik3)
Ritesh Manchikanti (vmanchik)

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

DDQN Agent after 250 episodes of training

Car Racing - OpenAI Gym

Action Space	:	Discrete(Throttle, Left, Right, Brake, No movement)
Observation space	:	(96x96) RGB images
Each episode	:	1000 frames
Reward	:	1000 - 0.1*(x)

Q-learning

Loop for each step of episode:

Choose A from S using policy derived from Q (e.g., ε-greedy)

Take action A, observe R, S'

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$$

$S \leftarrow S'$

until S is terminal

Target

Prediction

Immediate Reward

loss

Deep Q-Network(DQN)

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

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 $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

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

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

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

  End For

End For

Double Deep Q-Network(DDQN)

Algorithm 1: Double DQN (Hasset et al. 2015

Initialize primary network Q_θ, target network $Q_{\theta'}$, replay buffer $\mathcal{D}$, $\tau < 1$, C
for *each iteration* **do**
  for *each step* **do**
    Observe state s_t and select $a_t \sim \pi(s_t)$
    Execute action a_t and observe next state s_{t+1} and reward r_t;
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$.
  end
  for *each update step* **do**
    Sample minibatch of transitions $(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$
    Compute target Q value:

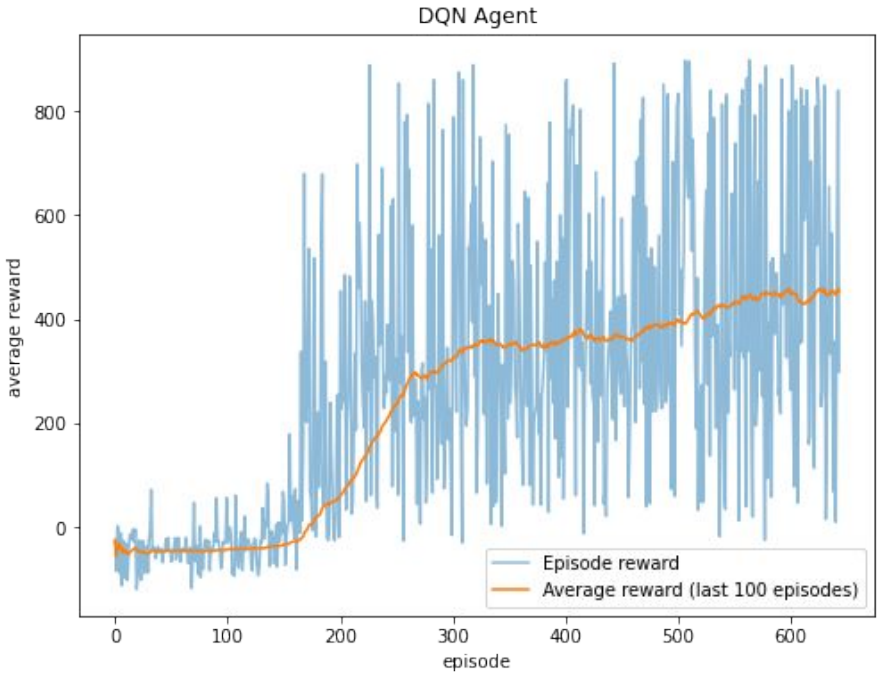
$$Q^*(s_t, a_t) = r_t + \gamma Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a'))$$

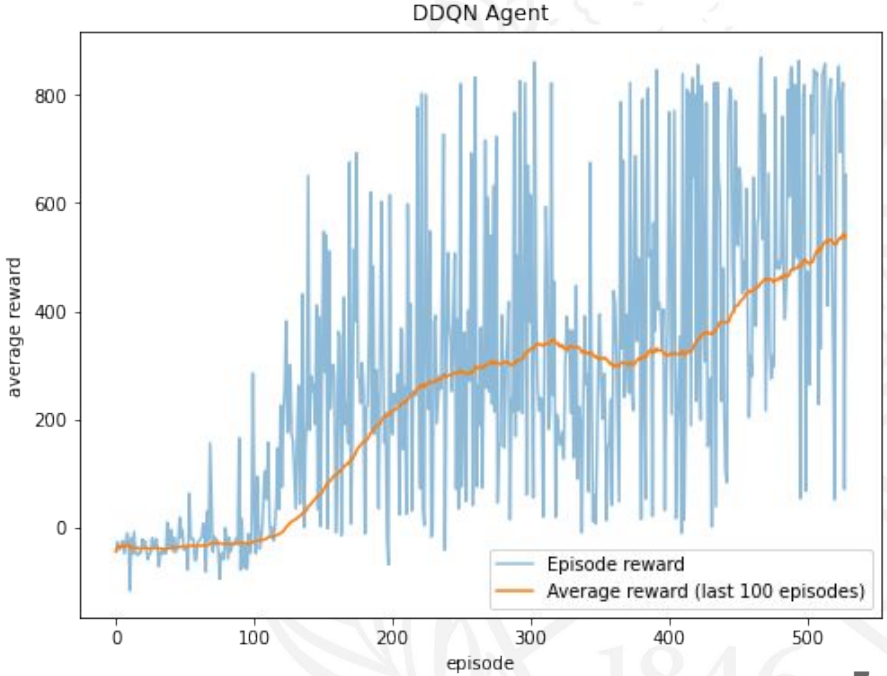
    Perform a gradient descent step on $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$ with respect to the primary network parameters θ
    Every C steps update target network parameters:

$$\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$$

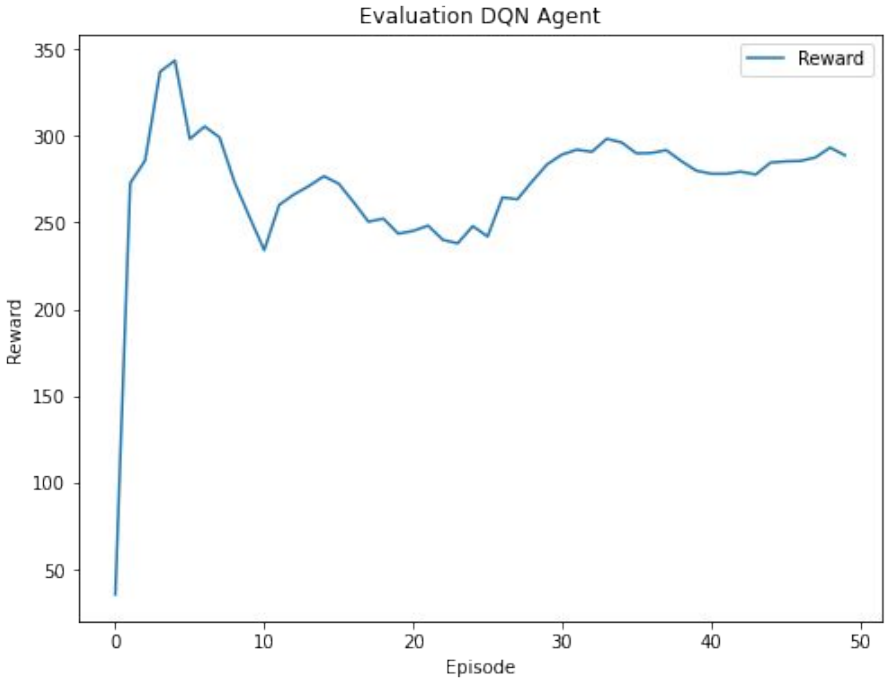
  end
end

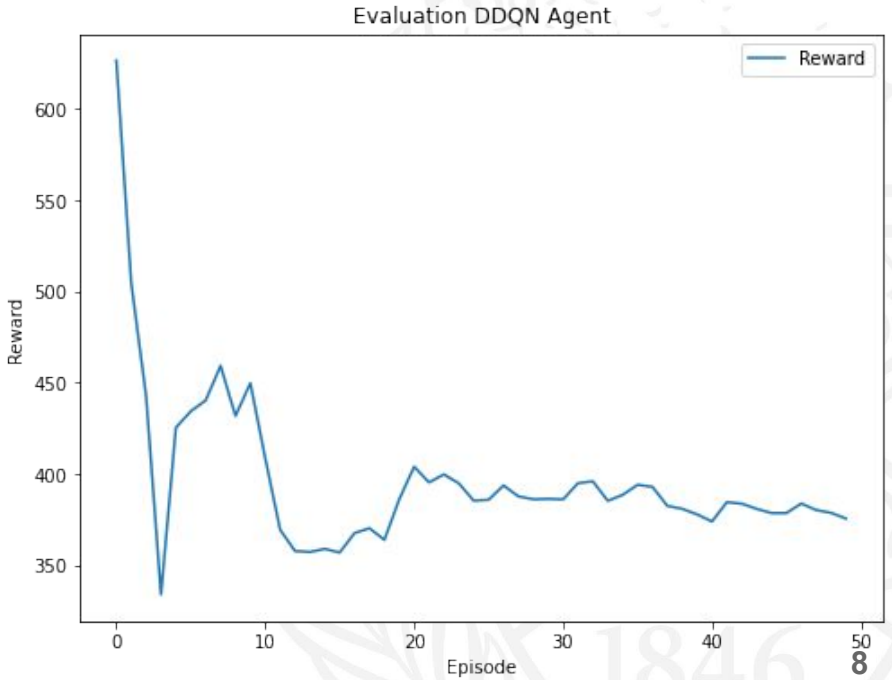
Training: DQN vs DDQN





Evaluation: DQN vs DDQN





3 Basic Approaches of RL

- Value Based
- Policy Based
- Model Based

  - Optimizing the policy Pi from the goal.
  - Best in Continuous Space and High Dimensional Space
  - It Follows Policy gradient trying to find the best parameters and smooth the update each step.
  - It does not need explore exploit strategies, as it uses stochastic policies.
  - Here we don’t need a value function that tells us the expected sum of rewards given a state and action. Here we try to find the policy function that will tell us the action directly as it maps actions to state.

Policy Optimization Method

- This method does not need any replay buffer to store the experience.
- We update the policy with one batch of experience and discard the batch.
- We need to define the policy gradient laws as this.

Loss Function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Algorithm

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s), \text{ where } a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t), \text{ for } t \geq 0$$

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ

for $k = 0, 1, 2, \ldots$ **do**

Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Compute policy update

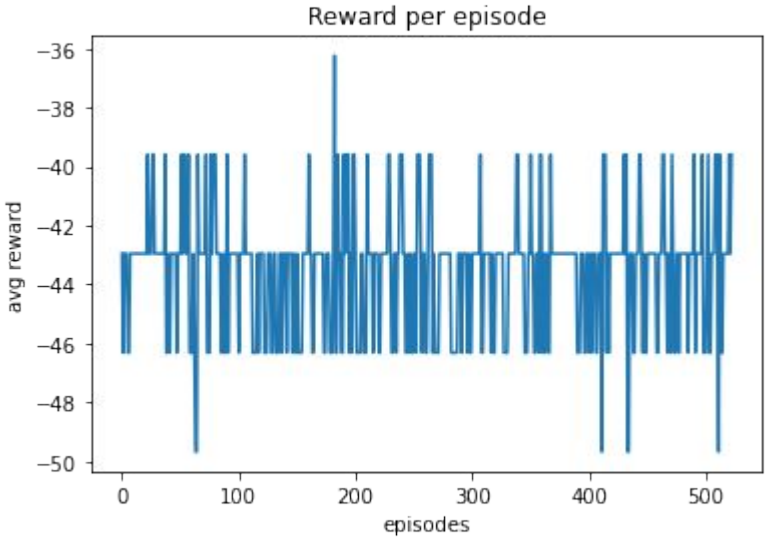
$$\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

by taking K steps of minibatch SGD (via Adam), where

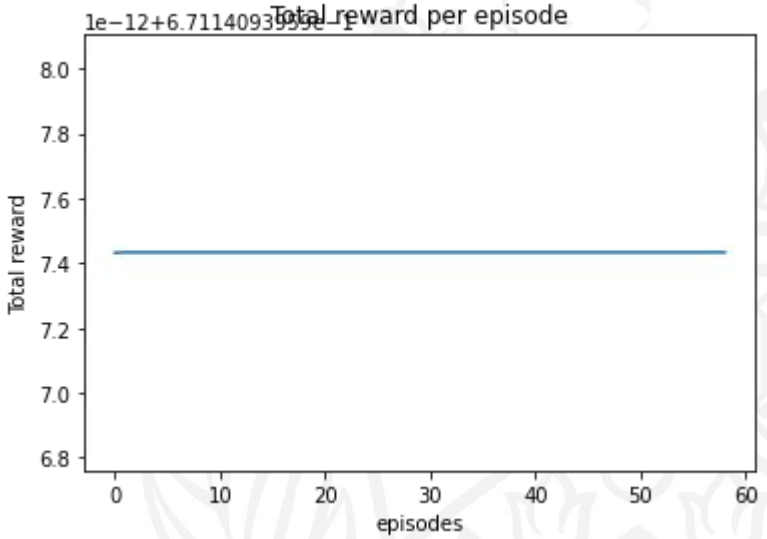
$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k})\right]\right]$$

end for

Training the agent for 500 Episodes



Evaluating the agent for 60 Episodes



Contribution

- DQN and DDQN - Arjun Ramesh Kaushik (kaushik3)
- PPO - Ritesh Manchikanti (vmanchik)

References

- https://arxiv.org/abs/1509.06461
- https://www.gymlibrary.dev/
- https://www.youtube.com/watch?v=vQ_ifavFBkI
- https://arxiv.org/pdf/1707.06347.pdf
- Lecture Slides

THANK YOU