Naresh Venkataramanan
Katherine Carson

Self Driving Car - Reinforcement Learning

What is PPO? DQN?

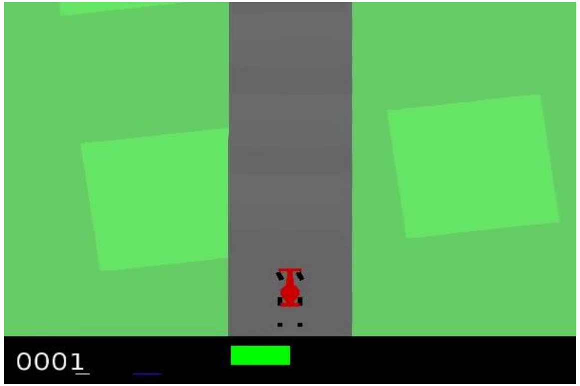
- PPO (Proximal Policy Optimization)
  - Easily implement cost function
  - Run gradient descent
  - Policy based

- DQN (Deep Q- Neural Network)
  - Stable training environment
    - Target Network
    - Skipping Frames
    - Rewards
  - Value based

Environment

- OpenGym AI
  - Compatible with Python
  - Access to standardized environments

https://gym.openai.com/envs/CarRacing-v0/



- env = gym.make('CarRacing-v0')
- env.render()
- observation = env.reset()
- observation, reward, done, info = env.step(action)

Tools

TensorFlow

- Open-source library
- Training and inference of deep neural networks

Keras

- API developed on top of TensorFlow
- Helps quickly build NN's

Pytorch

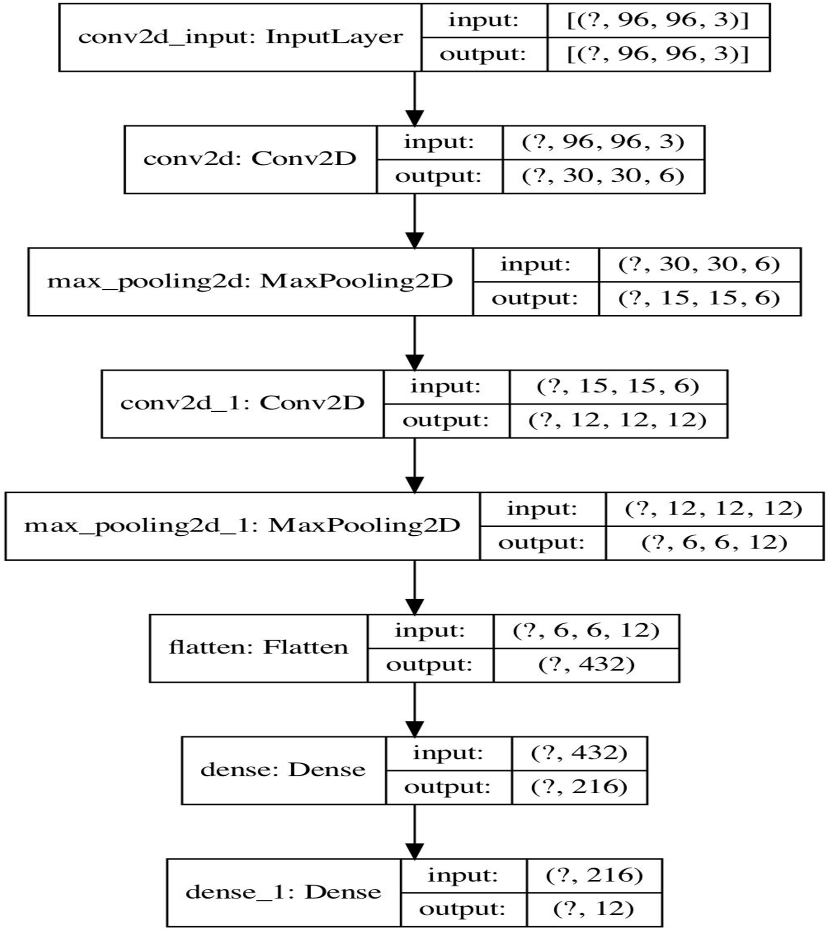
- Ease of use | debugging | dynamic graphs
- Conv2D
- Maxpool2d, Dense

GPU (graphics processing unit)

- Co-processor for faster computing

Model

- **class Conv2D**: 2D convolution layer (e.g. spatial convolution over images).
- **class MaxPooling2D**: Max pooling operation for 2D spatial data.
- **class Flatten**: Flattens the input. Does not affect the batch size.
- **class Dense**: the fully connected layer output = activation(dot(input, weight) + bias)



Pseudo Code for 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 $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ 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

Pseudo Code for PPO

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, ...$ **do**

  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

  Compute rewards-to-go $\hat{R}_t$.

  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.

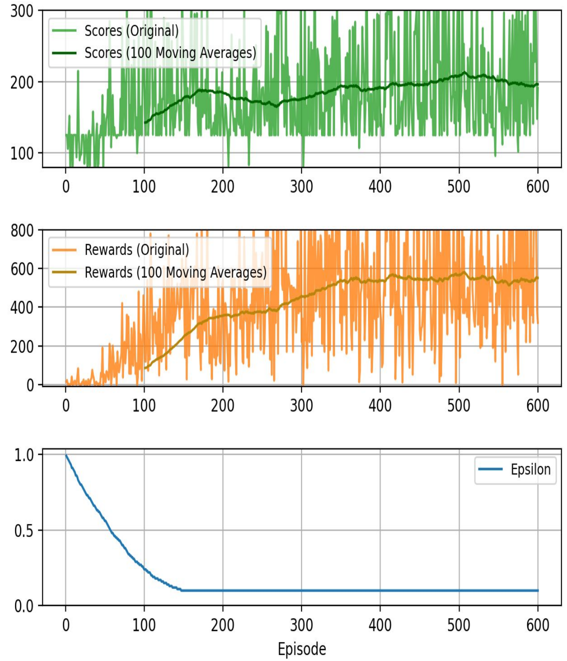
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.

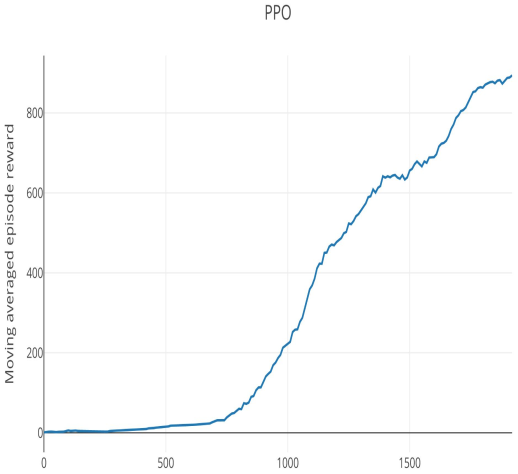
end for

DQN

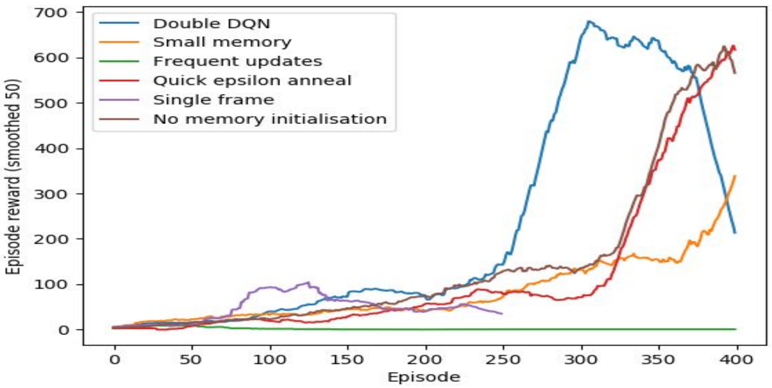


Scores (Original)
Scores (100 Moving Averages)
Rewards (Original)
Rewards (100 Moving Averages)
Epsilon
Episode

PPO



PPO
Moving averaged episode reward



Double DQN
Small memory
Frequent updates
Quick epsilon anneal
Single frame
No memory initialisation
Episode reward (smoothed 50)
Episode

Conclusion

- PPO results seem better than DQN
  - Average episode reward is higher after 2000 episodes
- DQN is more sample efficient
  - Reaches high average episode reward in around 400/500 episodes