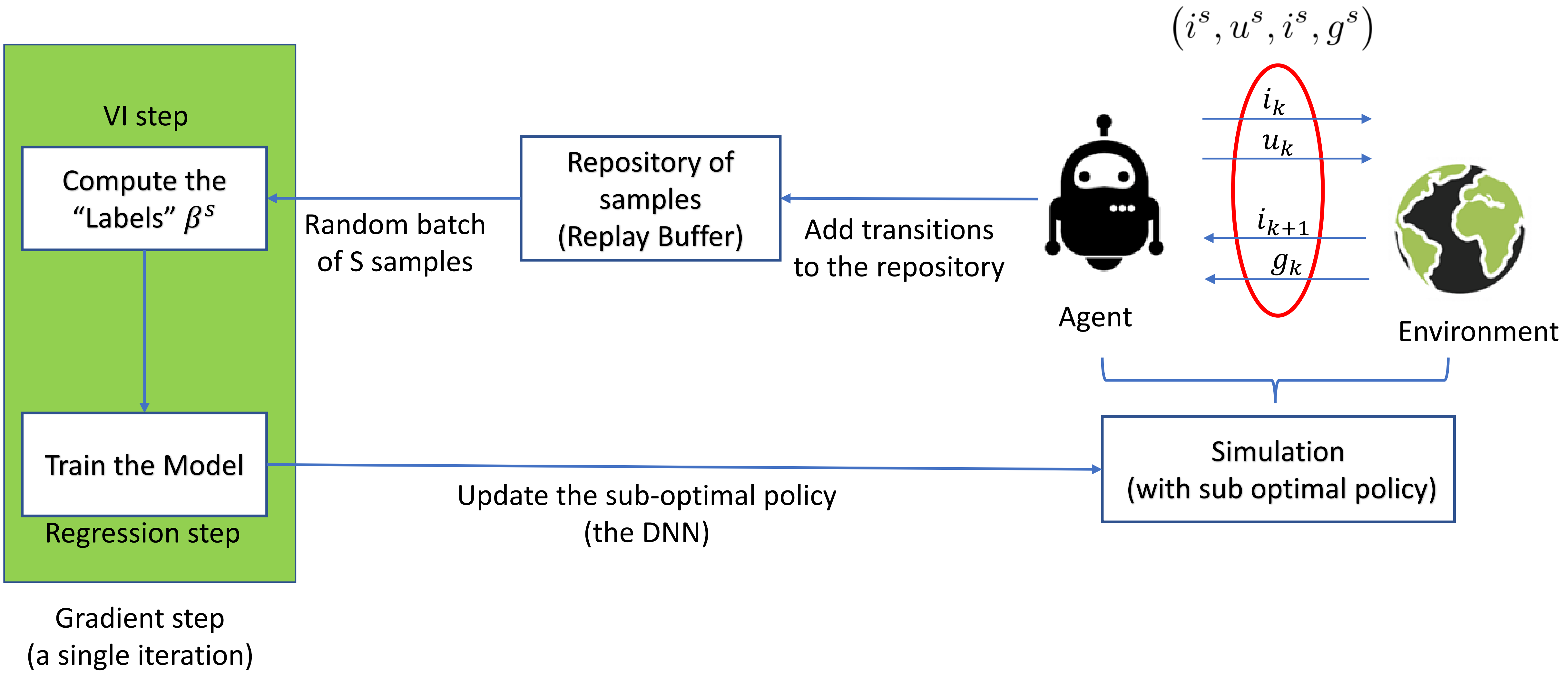
Deep Q-Learning Algorithm

- First let's recall the following diagram:



Deep Q Networks Algorithm (DQN)

Algorithm 1 DQN Algorithm (Minh et al, 2015)

Input: Initial DNN parameters θ^0 and architecture $Q_{\theta^0}(i,u)$. Replay Buffer $\mathcal{B}$.

for $p = 1, ..., P$: **do** (updates on the target network θ')

  Save the network parameters $\theta' \leftarrow \theta$.

  for $k = 0, ..., K$ **do** (obtaining new samples)

    Collect M sample transitions $\left\{(i^m, u^m, i^{m+1}, g^m)\right\}_{m=1}^{M}$ using the sub-optimal policy:

$$\tilde{\mu}^{(t+1)}(i) = (1 - \epsilon^{(t+1)}) \arg \min_{u \in U(i)} \left\{ \tilde{Q}_{\theta^{(t+1)}}(i,u) \right\} + \epsilon^{(t+1)} \mathrm{A}(U(i))$$

and add those sample transitions to the Replay Buffer $\mathcal{B}$.

    for $t = 1, ..., T$: **do** (gradient step)

      Randomly sample a batch of size S from the Replay Buffer $\mathcal{B}$.

      Perform one gradient step:

$$\theta^{(t+1)} = \theta^{(t)} - \gamma^{(t)} \left\{ \sum_{i=1}^{S} \nabla_\theta \tilde{Q}_{\theta^{(t)}}(i^s, u^s) \big(\tilde{Q}_{\theta^{(t)}}(i^s, u^s) - g(i^s, u^s) - \alpha \min_{u \in U(i^s)} \{ \tilde{Q}_{\theta'}(i^{s+1}, u) \} \big) \right\}$$

    end for

  end for

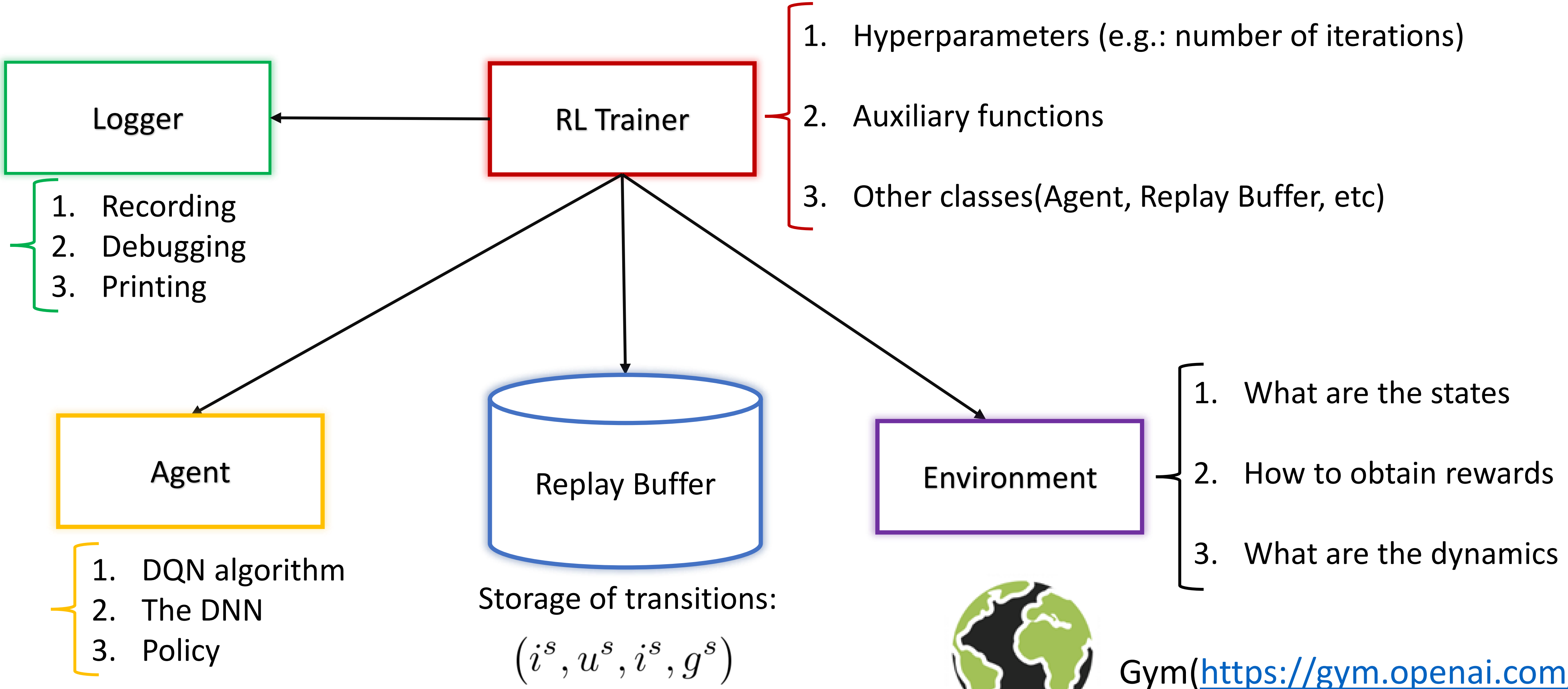
end for

Output: The last DNN configuration $\bar{\theta}$. A suboptimal policy:

$$\tilde{\mu}(i) = \arg \min_{u \in U(i)} \left\{ \tilde{Q}_{\bar{\theta}}(i,u) \right\}$$

Implementing DQN Algorithm

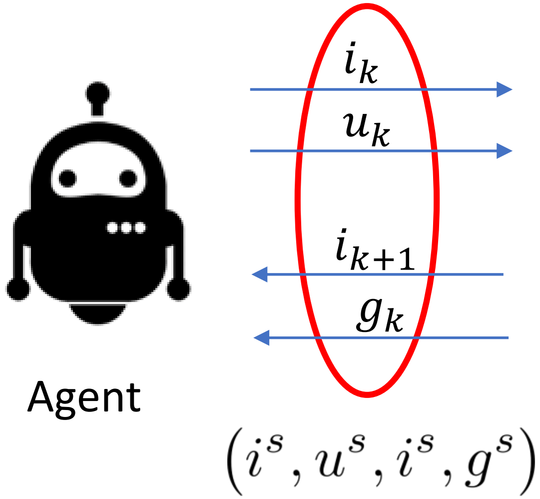
- We can divide the algorithm into classes:

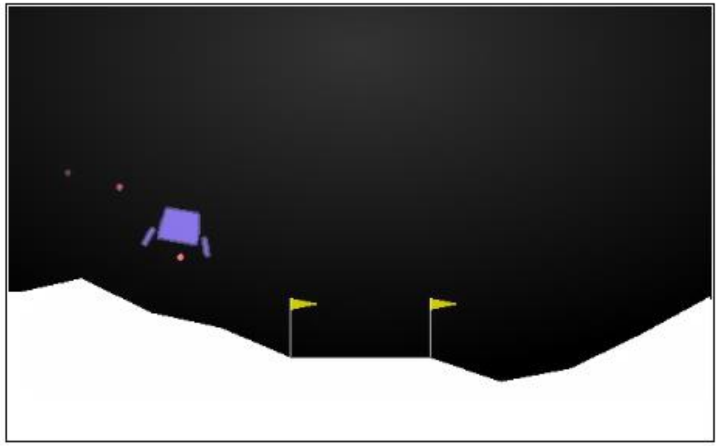


Gym(https://gym.openai.com/)

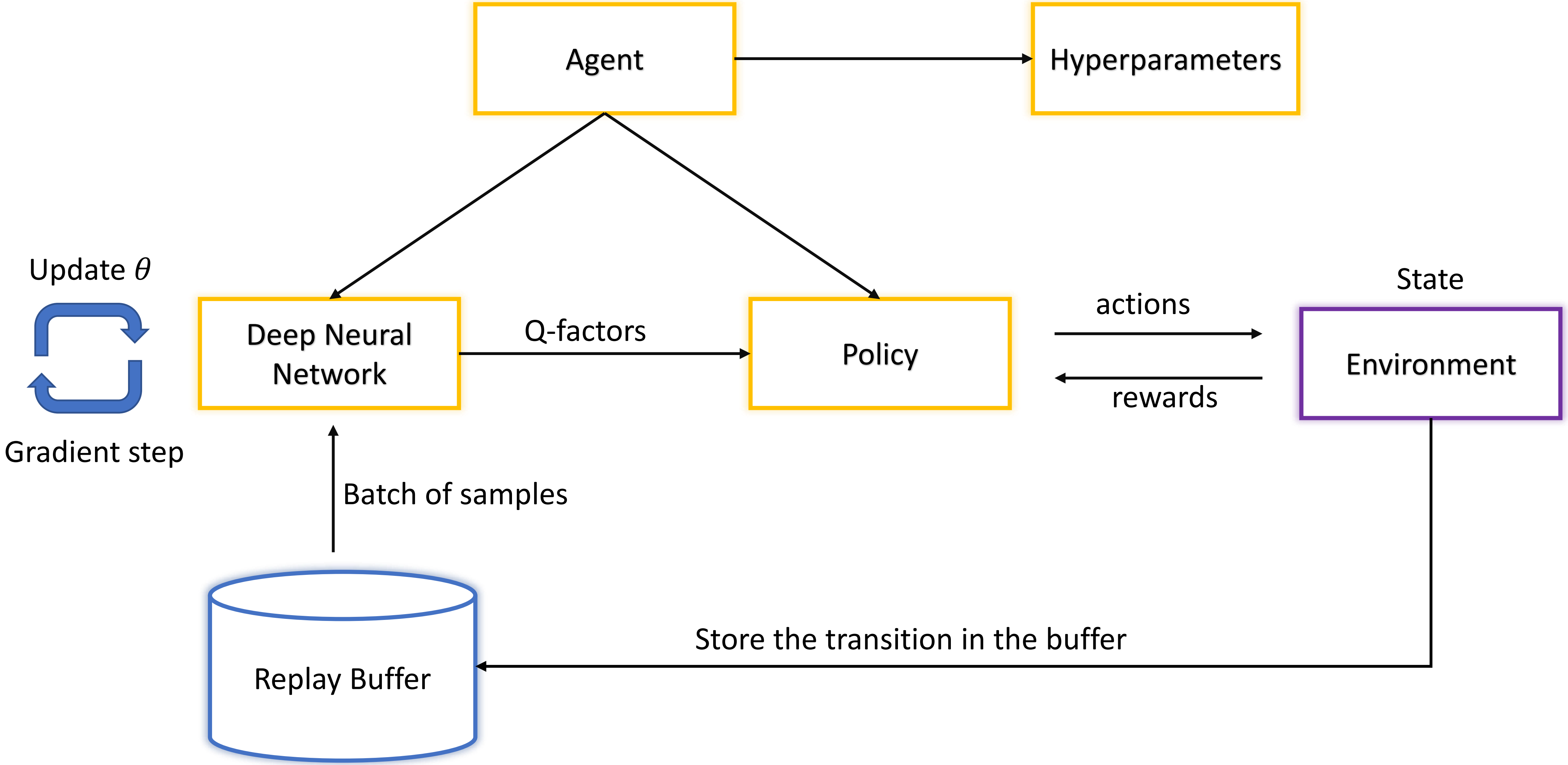
Environment

- We will try to design an AI to play the “Lunar Lander” game:
  - The goal is to land a space shuttle on the upright possible with zero vertical speed between the flags at coordinate (0,0).
  - Reward for moving from the top of the screen to landing pad and zero speed is about 100..140 points.
  - Landing outside the landing pad loses reward.
  - Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points.
  - Each leg ground contact is +10.
  - Firing main engine is -0.3 points each frame.
  - Max score is 200 points. Landing outside landing pad is possible.
  - Four discrete actions: do nothing, fire left orientation engine, fire main engine, fire right orientation engine.

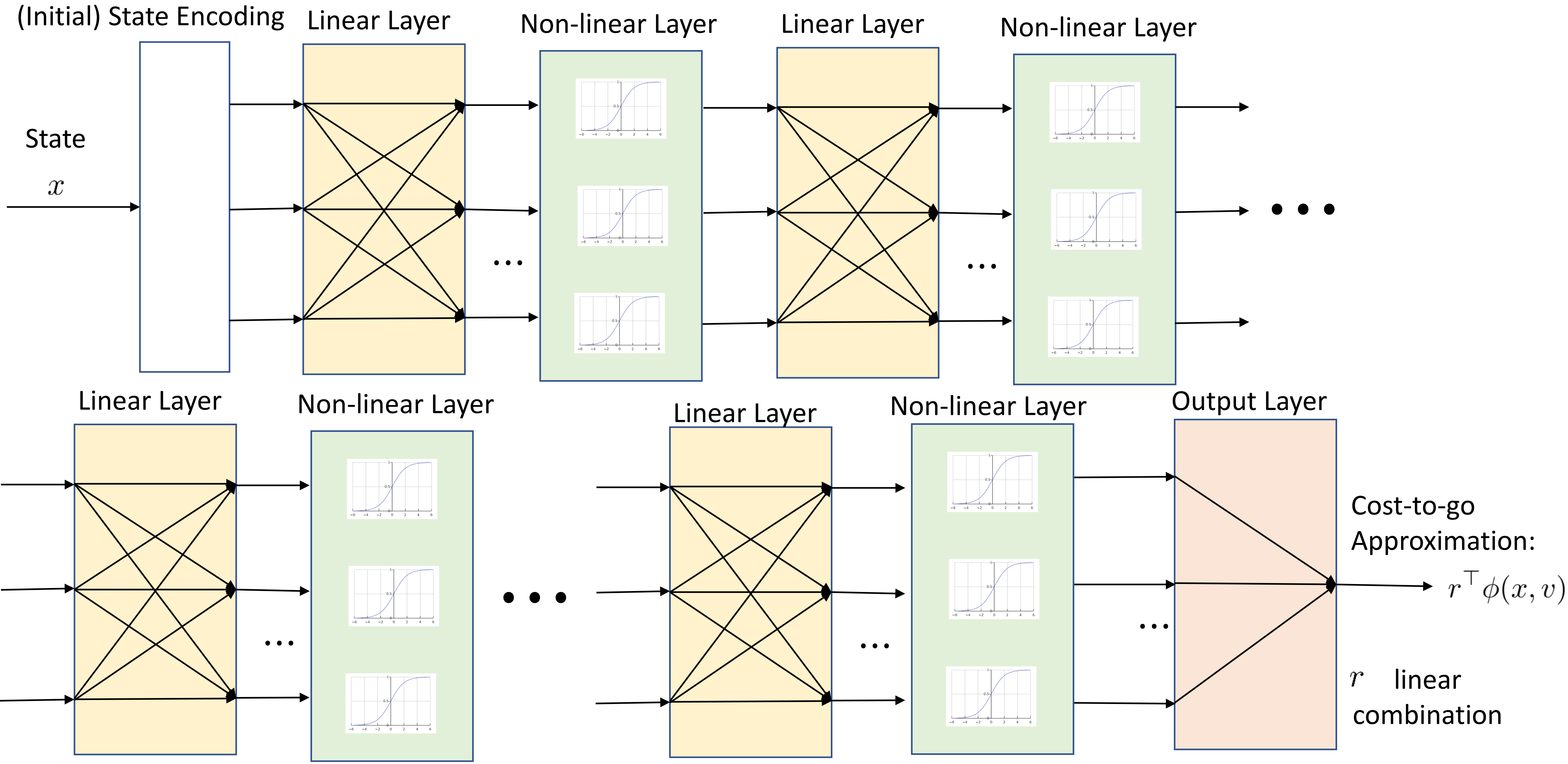




Agent configuration

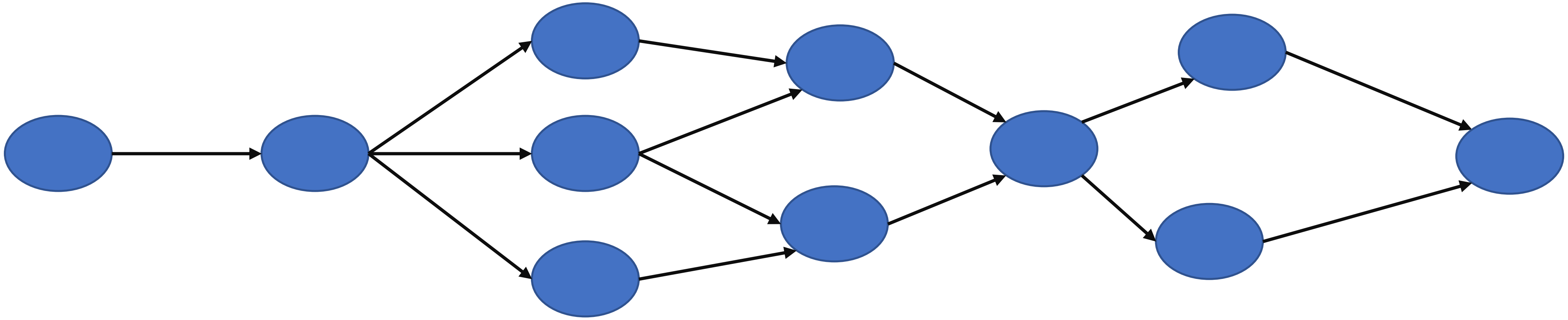


Implementing a DNN: Tensorflow



Tensorflow: a general overview

- Tensorflow is a platform that allows us flexible construction of **computational graphs**
  - A neural network is a computational graph
- The key idea is that Tensorflow allow us to first build the graph (with no data):



- And then, on runtime, it let’s us “flow” the data through the graph.
- It also allow us to differentiate the operations on each node: **Backpropagation**

DQN Implementation

Let's jump to the code!

Code is available on Bcourses.

Zoom meeting link with the lecture:
https://berkeley.zoom.us/rec/share/1O1Yd5b-9EZOXM_0zGHEQrQCLqv7aaa80XAX8vFfxRmWoG3gdGTUEvswzMyoig-q?startTime=1583877350000