Reinforcement Learning (CS4122)

Lecture 41 (15/05/2025)

Lecture 42 (16/05/2025)

- This is the beginning of Module 5.
- The topic of this lecture is:
  - Deep Deterministic Policy Gradient (DDPG)
- Here is a YouTube video of this lecture:

https://youtu.be/5WeeYQenQjU

- DDPG is the de-facto RL algorithm for continuous action space.
  - Other Deep-RL algos for continuous action space: TD3, PPO.
- DDPG is the extension of DQN for continuous action space.
  - Then why is has "Policy gradient" in the name.

    Answered later in these slides.

Which DQN Architecture for Continous Action Space?

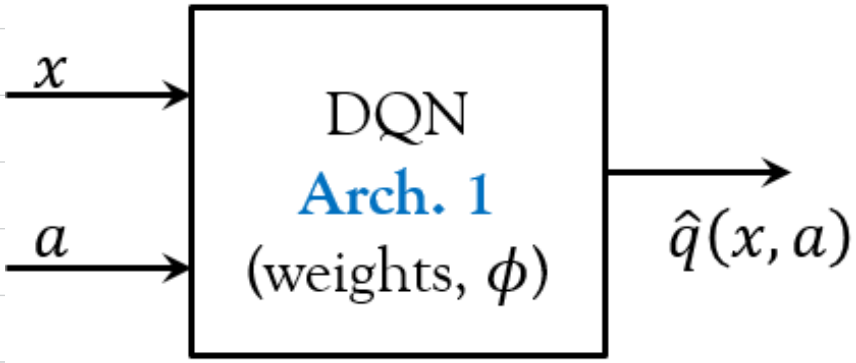
Algorithm 3: Psuedocode for Deep Q-Learning with Replay Buffer and Target Network

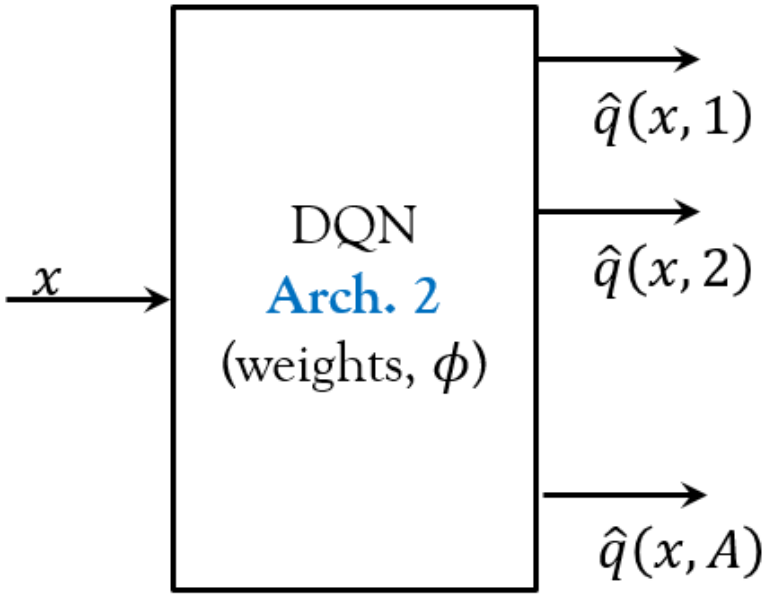
1 Initialize a predict DQN $\hat{q}(\cdot;\phi_P)$ and target DQN $\hat{q}(\cdot;\phi_T)$. Both the DQN should have the same architecture just different parameters.
2 Initialize an empty replay buffer, B, of a certain size.
3 Set an integer N_u (predict DQN update frequency), N_T (target DQN update frequency), N_b (training batch size), and $counter = 0$.
4 **for** *every episode until convergence* **do**
5 Reset the environment to get the current state x.
6 Choose learning rate, α, and exploration probability, ε, for this episode.
7 **for** *every time slot of the episode until convergence* **do**
8 Pick action, a, for current state, x, using a policy (like ε-greedy policy for the Q-function given by the current predict DQN $\hat{q}(\cdot;\phi_P)$).
9 Take action, a, and get reward, r, and next state, x'.
10 Append $\langle x, a, r, x' \rangle$ to replay buffer B.
11 **if** $counter \% N_u == 0$ **then**
12 Randomly sample a batch of size N_b from replay buffer B.
13 Computer input, X, and target, y, to train predict DQN using the sampled batch and the **current** target DQN $\hat{q}(\cdot;\phi_T)$.
14 Use X and y to update ϕ_P of the predict DQN by taking **one** gradient **descent** step based on the current learning rate, α.
15 **if** $counter \% N_T == 0$ **then**
16 Set $\phi_T \leftarrow \phi_P$.
17 Set $x \leftarrow x'$ and $counter \leftarrow counter + 1$.

DQN Arch.1 only.

Why in next page.

Which DQN Architecture for Continous Action Space?





O/P layer needs infinite no. of o/ps for continous action space.

Problem with DQN Arch. 1 for Continuous Action Space

Algorithm 3: Psuedocode for Deep Q-Learning with Replay Buffer and Target Network

1 Initialize a predict DQN $\hat{q}(\cdot;\phi_P)$ and target DQN $\hat{q}(\cdot;\phi_T)$. Both the DQN should have the same architecture just different parameters.
2 Initialize an empty replay buffer, B, of a certain size.
3 Set an integer N_u (predict DQN update frequency), N_T (target DQN update frequency), N_b (training batch size), and $counter = 0$.
4 **for** *every episode until convergence* **do**
5 Reset the environment to get the current state x.
6 Choose learning rate, α, and exploration probability, ε, for this episode.
7 **for** *every time slot of the episode until convergence* **do**
8 Pick action, a, for current state, x, using a policy (like ε-greedy policy for the Q-function given by the current predict DQN $\hat{q}(\cdot;\phi_P)$).
9 Take action, a, and get reward, r, and next state, x'.
10 Append $\langle x, a, r, x' \rangle$ to replay buffer B.
11 **if** $counter \% N_u == 0$ **then**
12 Randomly sample a batch of size N_b from replay buffer B.
13 Computer input, X, and target, y, to train predict DQN using the sampled batch and the **current** target DQN $\hat{q}(\cdot;\phi_T)$.
14 Use X and y to update ϕ_P of the predict DQN by taking **one** gradient **descent** step based on the current learning rate, α.
15 **if** $counter \% N_T == 0$ **then**
16 Set $\phi_T \leftarrow \phi_P$.
17 Set $x \leftarrow x'$ and $counter \leftarrow counter + 1$.

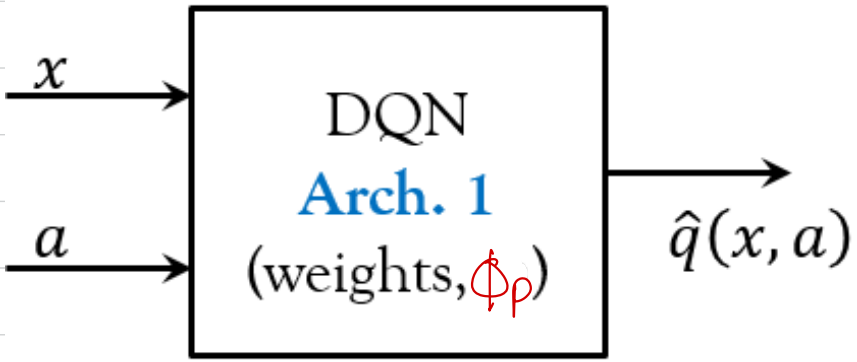
DQN Arch. 1 only.

Pick actions
Uses predict network.

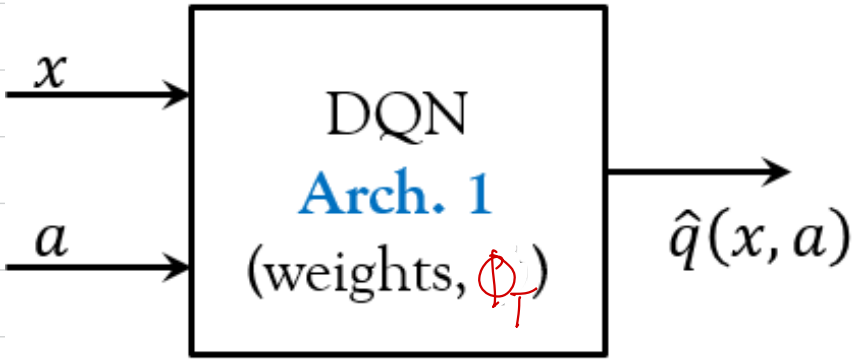
Generating target.
Uses target network.

Problem with DQN Arch. 1 for Continuous Action Space

Predict Network



Target Network



Picking Actions (line 8)

w.p. ε:

$a \leftarrow$ Pick actions uniformly at random.

w.p. $1-\varepsilon$:

$$a \leftarrow \boxed{\underset{\tilde{a} \in A(x)}{\operatorname{argmax}}} \hat{q}(x, \tilde{a}; \phi_P)$$

Generating target (line 13)

$$y = r + B \boxed{\max_{\tilde{a} \in A(x')}} \hat{q}(x', \tilde{a}; \phi_T)$$

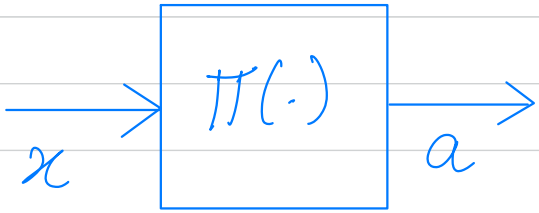
Maximization operation can't be brute-forced in continuous action space.

How to "smartly" do maximization in continuous action space?

- Generate samples from action space at random. Evaluate Q-function only for these samples. Find the maximum Q-value among these samples.

- Use special NN arch. where maximization over action space is trivial (look into previous year's end-sem paper).

- DDPG approach: Train a seperate NN that outputs the "best" action for a given state.



Neural Networks in DDPG

Algorithm 3: Psuedocode for Deep Q-Learning with Replay Buffer and Target Network

1 Initialize a predict DQN $\hat{q}(\cdot;\phi_P)$ and target DQN $\hat{q}(\cdot;\phi_T)$. Both the DQN should have the same architecture just different parameters.
2 Initialize an empty replay buffer, B, of a certain size.
3 Set an integer N_u (predict DQN update frequency), N_T (target DQN update frequency), N_b (training batch size), and $counter = 0$.
4 **for** *every episode until convergence* **do**
5 Reset the environment to get the current state x.
6 Choose learning rate, α, and exploration probability, ε, for this episode.
7 **for** *every time slot of the episode until convergence* **do**
8 Pick action, a, for current state, x, using a policy (like ε-greedy policy for the Q-function given by the current predict DQN $\hat{q}(\cdot;\phi_P)$).
9 Take action, a, and get reward, r, and next state, x'.
10 Append $\langle x, a, r, x' \rangle$ to replay buffer B.
11 **if** $counter \% N_u == 0$ **then**
12 Randomly sample a batch of size N_b from replay buffer B.
13 Computer input, X, and target, y, to train predict DQN using the sampled batch and the **current** target DQN $\hat{q}(\cdot;\phi_T)$.
14 Use X and y to update ϕ_P of the predict DQN by taking **one** gradient **descent** step based on the current learning rate, α.
15 **if** $counter \% N_T == 0$ **then**
16 Set $\phi_T \leftarrow \phi_P$.
17 Set $x \leftarrow x'$ and $counter \leftarrow counter + 1$.

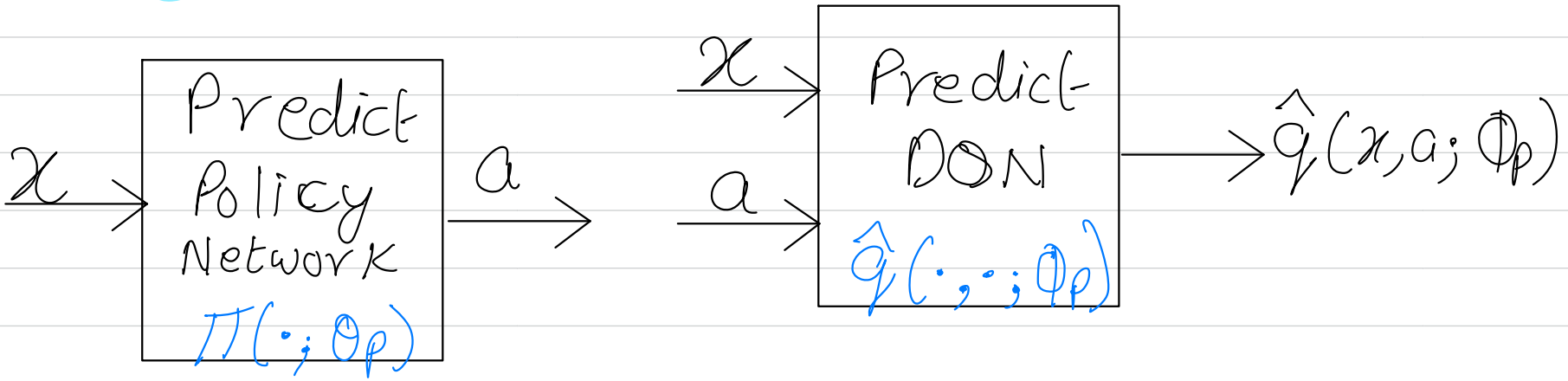
Pick actions
Uses predict network.

Generating target.
Uses target network.

Neural Networks in DDPG

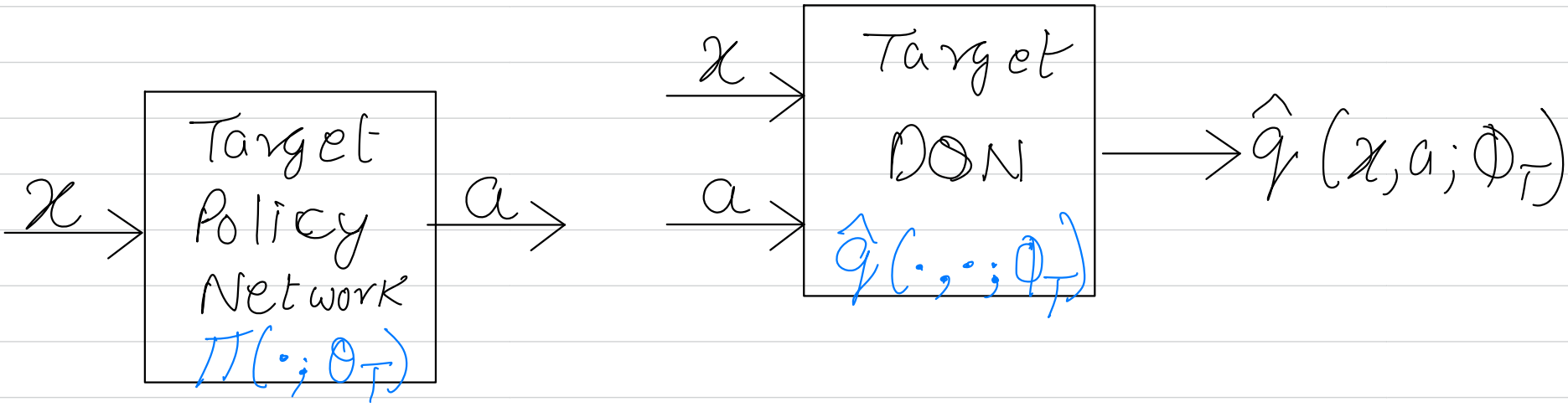
Line 8

Picking Actions



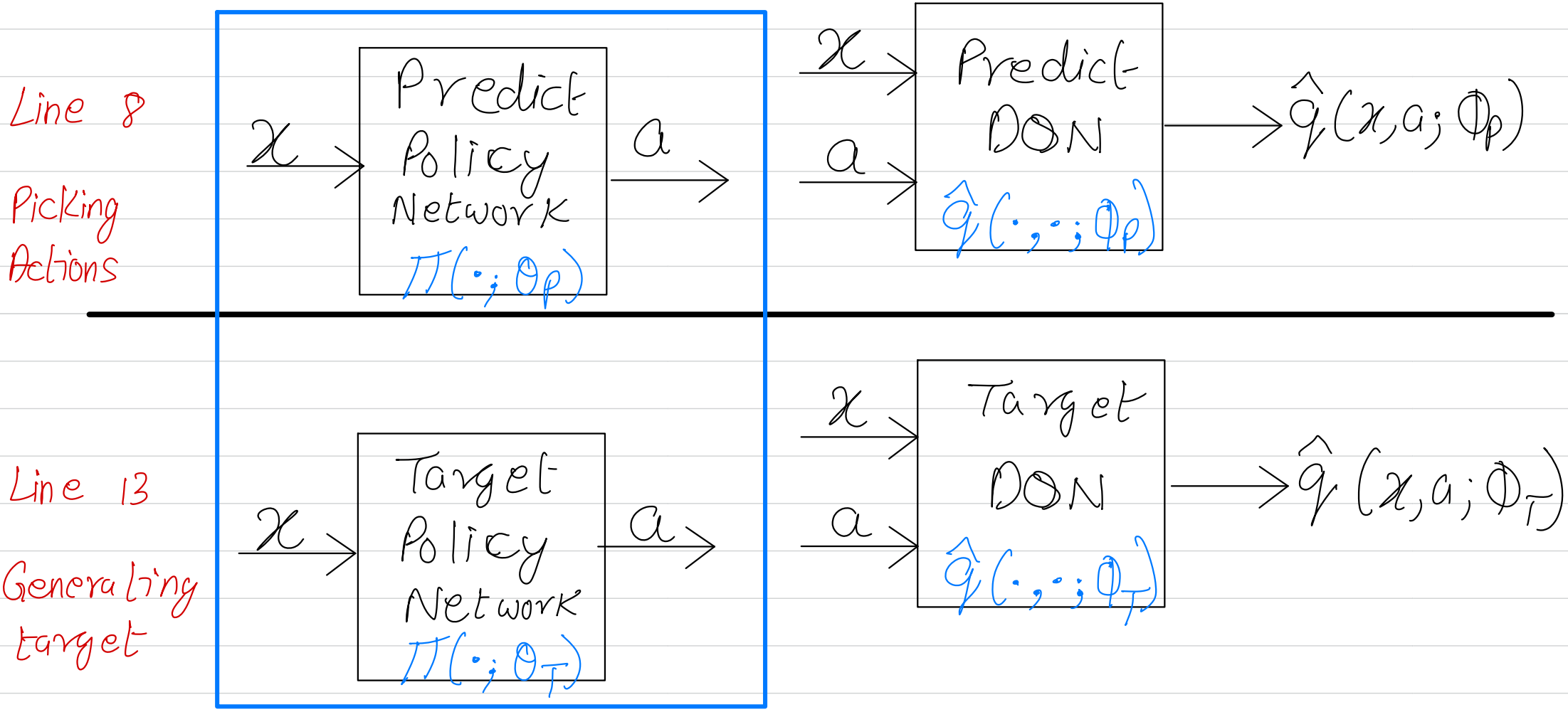
Line 13

Generating target



> Predict policy network is the final deliverable after training.

Neural Networks in DDPG



> The last layer of policy networks have hyperbolic tan, $\tanh(x)$, activation. Why? To bound the range of actions.

- Linear activation can also be used.

Inferencing in DDPG

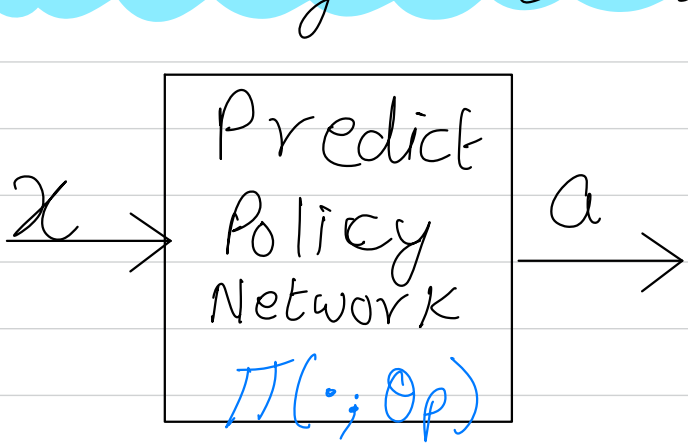
Algorithm 3: Psuedocode for Deep Q-Learning with Replay Buffer and Target Network

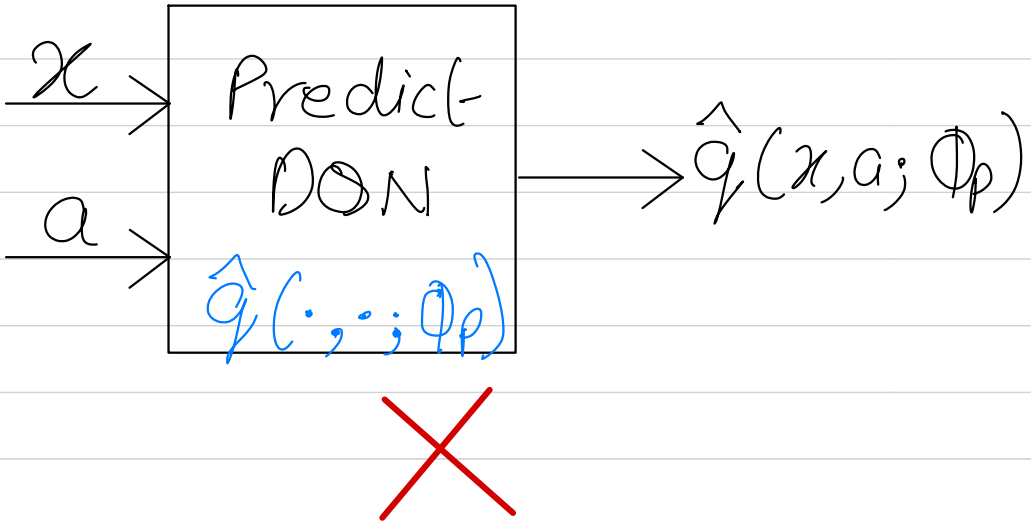
1 Initialize a predict DQN $\hat{q}(\cdot;\phi_P)$ and target DQN $\hat{q}(\cdot;\phi_T)$. Both the DQN should have the same architecture just different parameters.
2 Initialize an empty replay buffer, B, of a certain size.
3 Set an integer N_u (predict DQN update frequency), N_T (target DQN update frequency), N_b (training batch size), and $counter = 0$.
4 **for** *every episode until convergence* **do**
5 Reset the environment to get the current state x.
6 Choose learning rate, α, and exploration probability, ε, for this episode.
7 **for** *every time slot of the episode until convergence* **do**
8 Pick action, a, for current state, x, using a policy (like ε-greedy policy for the Q-function given by the current predict DQN $\hat{q}(\cdot;\phi_P)$).
9 Take action, a, and get reward, r, and next state, x'.
10 Append $\langle x, a, r, x' \rangle$ to replay buffer B.
11 **if** $counter \% N_u == 0$ **then**
12 Randomly sample a batch of size N_b from replay buffer B.
13 Computer input, X, and target, y, to train predict DQN using the sampled batch and the **current** target DQN $\hat{q}(\cdot;\phi_T)$.
14 Use X and y to update ϕ_P of the predict DQN by taking **one** gradient **descent** step based on the current learning rate, α.
15 **if** $counter \% N_T == 0$ **then**
16 Set $\phi_T \leftarrow \phi_P$.
17 Set $x \leftarrow x'$ and $counter \leftarrow counter + 1$.

Pick actions
Uses predict network.

Generating target.
Uses target network.

Inferencing in DDPG

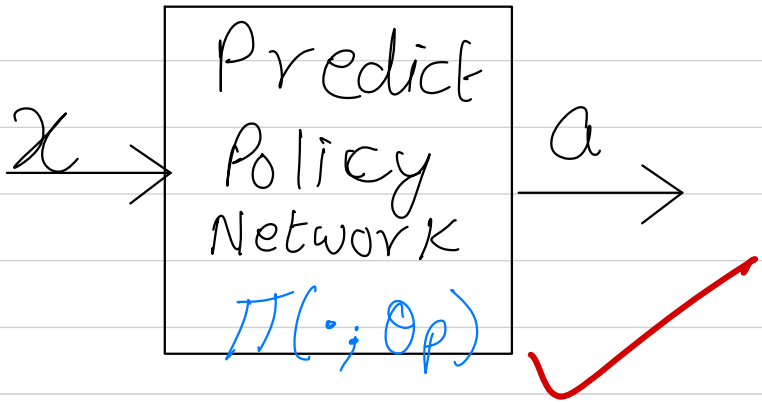


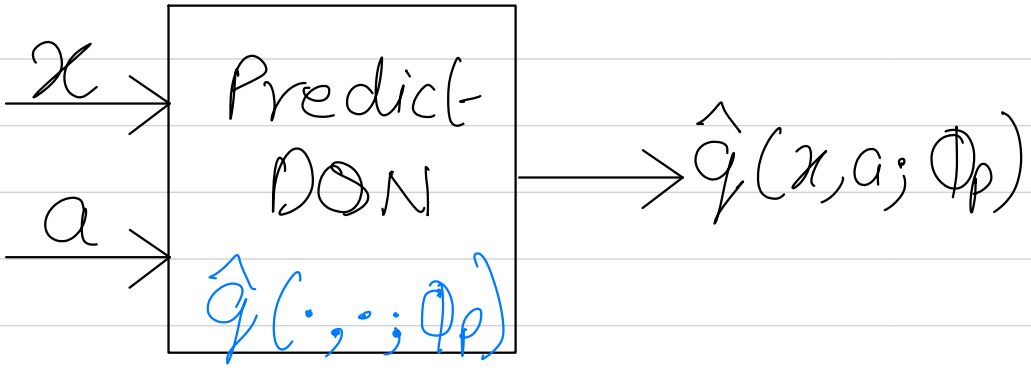


Picking actions (line 8):

- Predict DQN is NOT relevant for picking actions during inferencing; only required during training.
- Why predict DQN is NOT relevant for picking actions?
  - Because for continuous action space, we can't find the optimal action using $\underset{a \in A(x)}{\operatorname{argmax}}\ \hat{q}(x,a;\phi_p)$.

Inferencing in DDPG

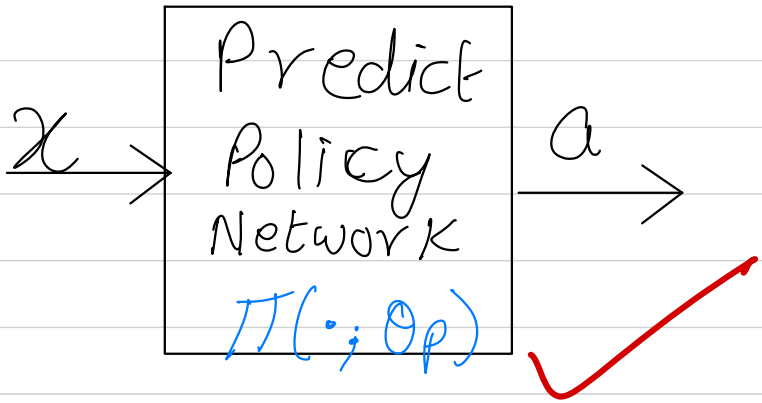


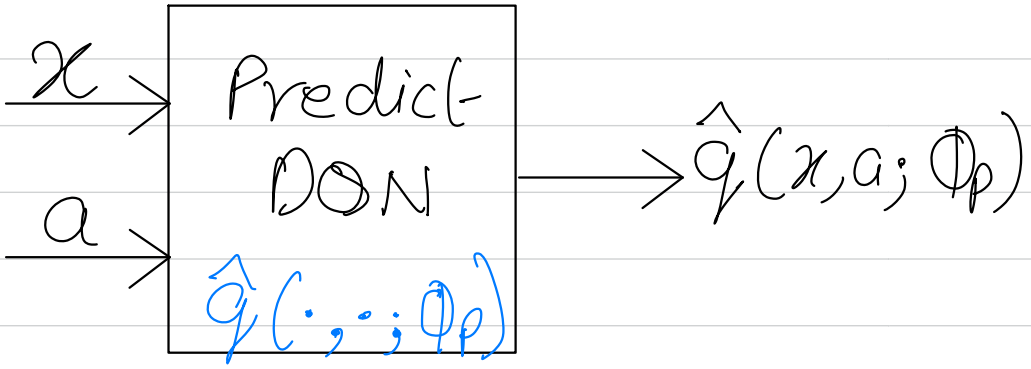


Picking actions (line 8):

- Predict policy network is used for picking actions during inferencing. The formula used for picking actions differs during training and testing phase.
  - During testing, we don't have to do exploration.

Inferencing in DDPG





Picking actions (line 8):

Training (DDPG approch)

$a = \pi(x;\theta_p) + \varepsilon$

where $\varepsilon \sim N(0,\sigma)$

$\pi(x;\theta_p)$: exploitation term

ε: exploration term. σ decreases as training progresses.

Training (Alternate approach)

w.p. ε:

a = select uniformly at random from action space.

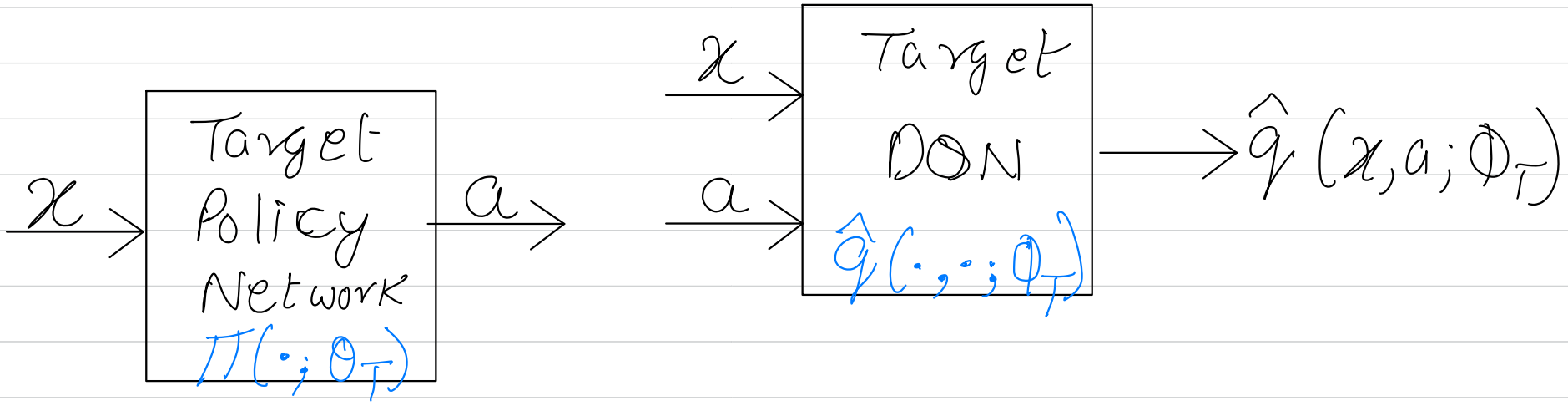
w.p. $1-\varepsilon$:

$a = \pi(x;\theta_p)$

Testing

$a = \pi(x;\theta_p)$

Inferencing in DDPG



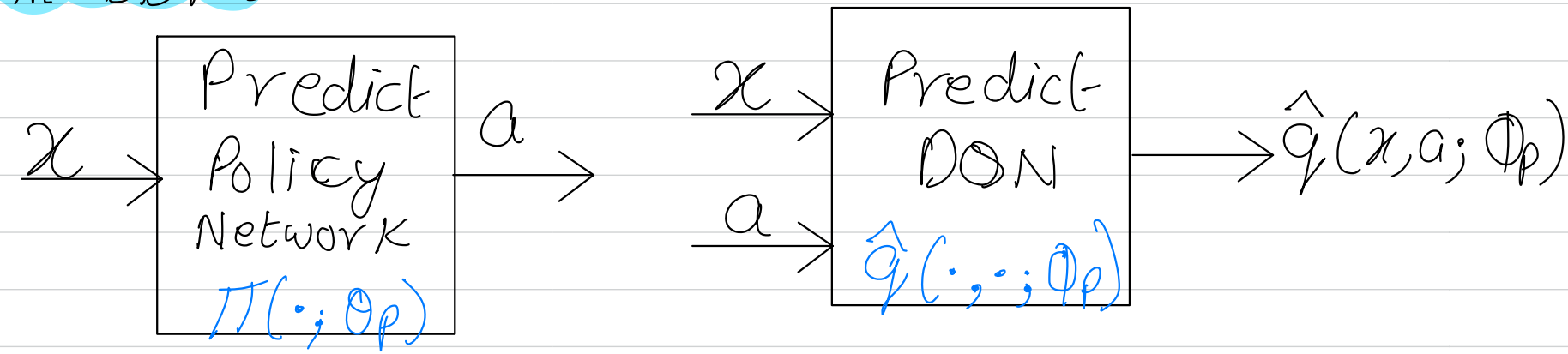
Generating target (line 13):

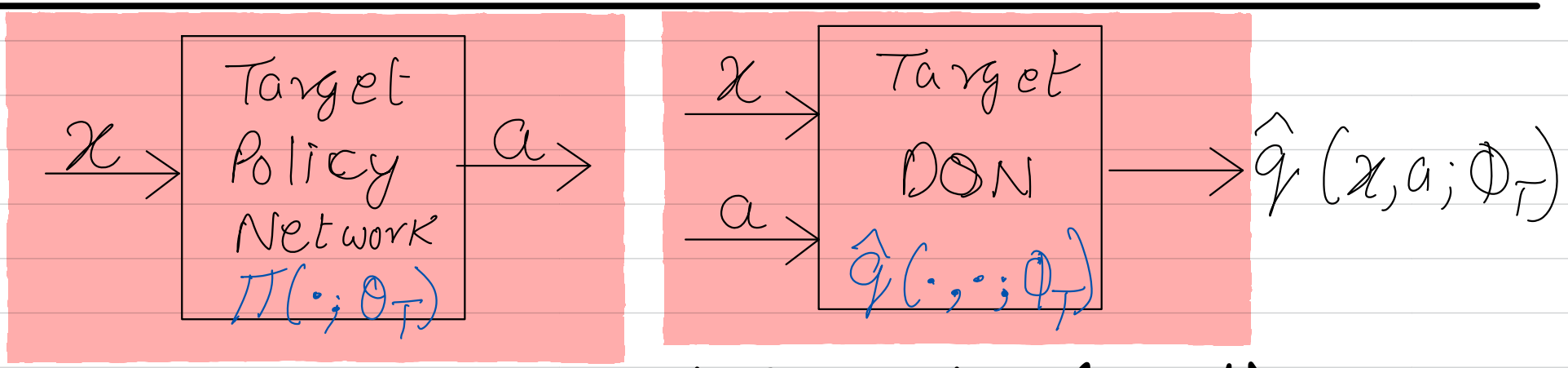
$$y = r + \beta \max_{\tilde{a} \in A(x')} \hat{q}(x', \tilde{a}; \phi_T)$$

$$= r + \beta \hat{q}\left(x', \operatorname*{argmax}_{\tilde{a} \in A(x')} \hat{q}(x', \tilde{a}; \phi_T); \phi_T\right)$$

$$= r + \beta \hat{q}(x', \pi(x'; \theta_T); \phi_T)$$

Training in DDPG





Approach 1:

anything periodic; episodes, time-slots

Every N_T steps:

$\theta_T \leftarrow \theta_p$

$\phi_T \leftarrow \phi_p$

Approach 2 (smooth)

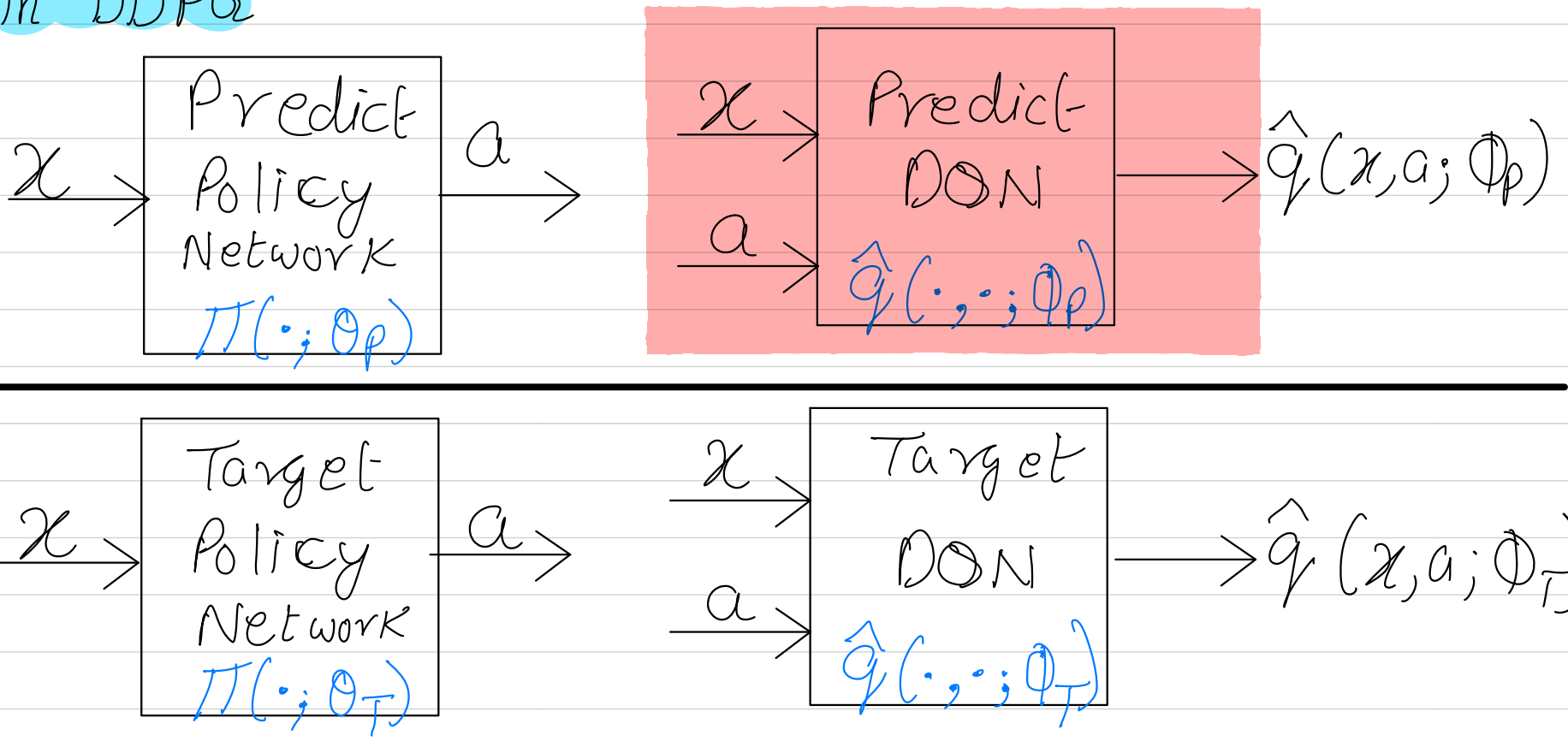
$\theta_T = (1-\alpha)\,\theta_T + \alpha\,\theta_p$

$\phi_T = (1-\tilde{\alpha})\,\phi_T + \tilde{\alpha}\,\phi_p$

} DDPG uses this.

$\alpha, \tilde{\alpha}$ are quite small. Like 0.001.

Training in DDPG



- Similar to DQN. The difference is that while generating target we can't use $y = r + \beta \max_{a' \in A(x')} \hat{q}(x', a'; \phi_T)$. Why? This is NOT possible for continuous action space. We instead have to do $y = r + \beta \hat{q}(x', \pi(x'; \theta_T); \phi_T)$.

Training in DDPG (Predict DQN)

- Sample a batch B from replay buffer of size N_B.

- Calculate mean square loss between the predicted Q-value and target Q-value across all samples in B,

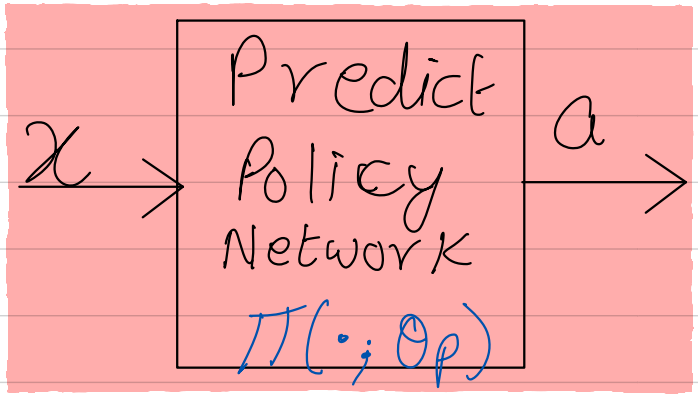
$$L = \frac{1}{N_B} \sum_{K=1}^{N_B} \left(\underbrace{r_K + \beta \hat{q}(x_K', \pi(x_K'; \theta_T); \phi_T)}_{\text{Target Q-Value}} - \underbrace{\hat{q}(x_K, a_K; \phi_p)}_{\text{Predicted Q-value}} \right)^2$$

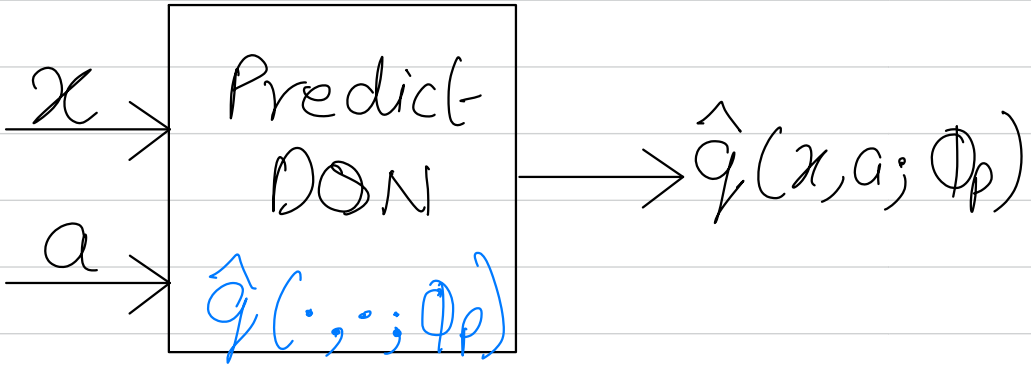
where $\langle x_K, a_K, r_K, x_K' \rangle$ is the K^{th} sample of B.

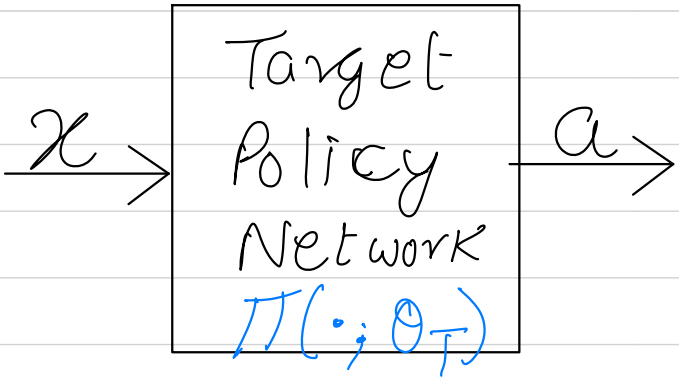
- Calculate $\nabla_{\phi_p} L$ using Autograd. $\nabla_{\phi_p} L$ is the gradient of L with respect to ϕ_p. ϕ_p is predict DQN's parameters.

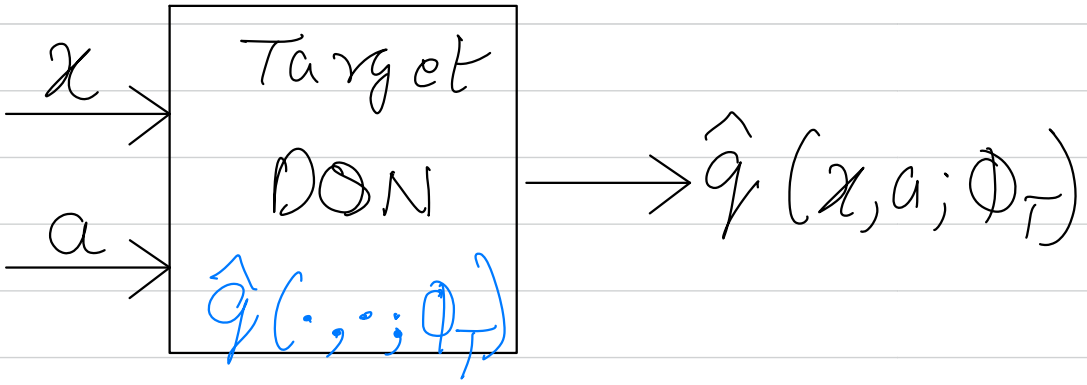
- Do gradient descent: $\phi_p \leftarrow \phi_p - \gamma \nabla_{\phi_p} L$ (γ: learning rate.)

Training in DDPG









PTO

Training in DDPG (Predict Policy Network)

- Sample a batch B from replay buffer of size N_B.
- Calculate mean Q-values using actions suggested by predict policy network across all samples in B,

$$J = \frac{1}{N_B} \sum_{K=1}^{N_B} \hat{q}(x_K, \pi(x_K; \theta_p); \phi_p)$$

where $\langle x_K, a_K, r_K, x_K' \rangle$ is the K^{th} sample of B.

Not needed here.

- Calculate $\nabla_{\theta_p} J$ using Autograd. $\nabla_{\theta_p} J$ is the gradient of J with respect to θ_p. θ_p is predict policy network parameters.
- Do gradient ascent : $\theta_p \leftarrow \theta_p + \gamma_2 \nabla_{\theta_p} J$

Learning rate.