

PROGETTO DATA-DRIVEN

Corso di Progettazione dei Sistemi di Controllo - **Ciavola Claudio**

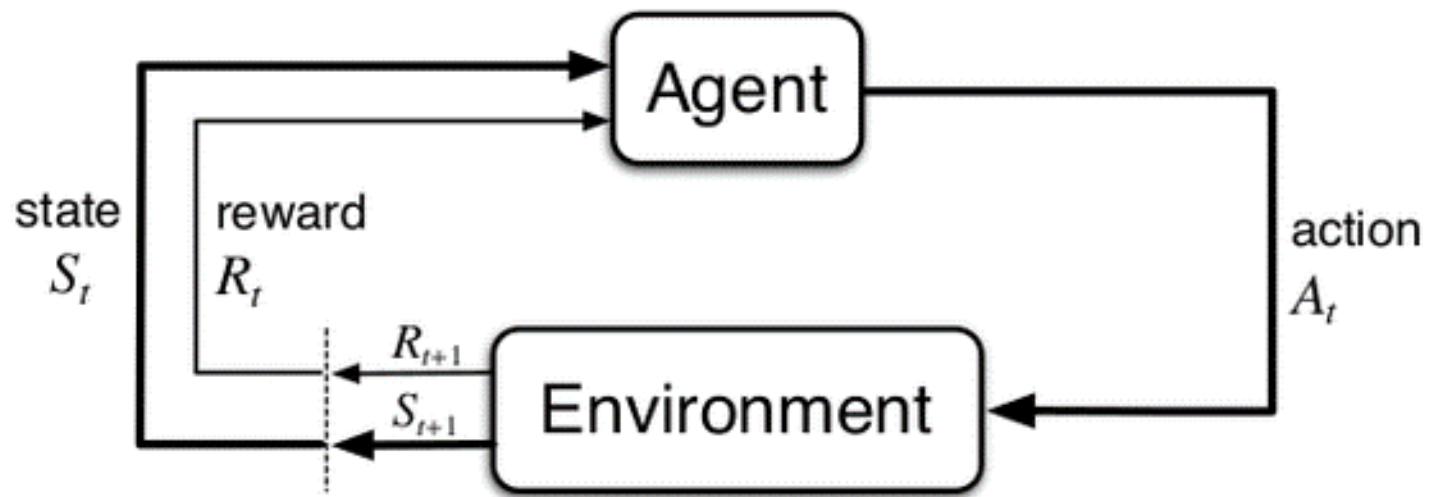
INTRODUZIONE



REINFORCEMENT LEARNING

Elementi

- Agente
- Ambiente
- Azioni
- Stati
- Ricompensa
- Politica
- Valore



PROCESSO DECISIONALE MARKOVIANO

Insieme finito di stati

$$s \in \mathcal{S}$$

Insieme finito di azioni

$$a \in \mathcal{A}$$

Dinamica di un MDP

$$p(s', r | s, a) = \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$$

Funzione di transizione di stato

$$p(s' | s, a) = \Pr\{S_t = s' | S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r | s, a)$$

Funzione di ricompensa

$$r(s, a) = \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r | s, a)$$

EQUAZIONE DI BELLMAN

Politica

$$\pi(a|s) = \Pr\{A_t = a | S_t = s\} \quad \forall a \in \mathcal{A}, s \in \mathcal{S}$$

Funzione azione-valore per la politica π

$$q_\pi(a, s) = \mathbb{E}_\pi [\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a]$$

Equazione di Bellman per funzione azione-valore

$$q_\pi(s) = \mathbb{E} [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

EQUAZIONI DI OTTIMALITÀ DI BELLMAN

Politica Ottimale

$$\pi \geq \pi' \Leftrightarrow q_\pi(s) \geq q_{\pi'}(s) \quad \forall s \in \mathcal{S}$$

Equazione di ottimalità per funzione azione-valore

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a) \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

$$q_*(s, a) = \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} q_{\pi}(S_{t+1}, a') \mid S_t = s, A_t = a \right]$$

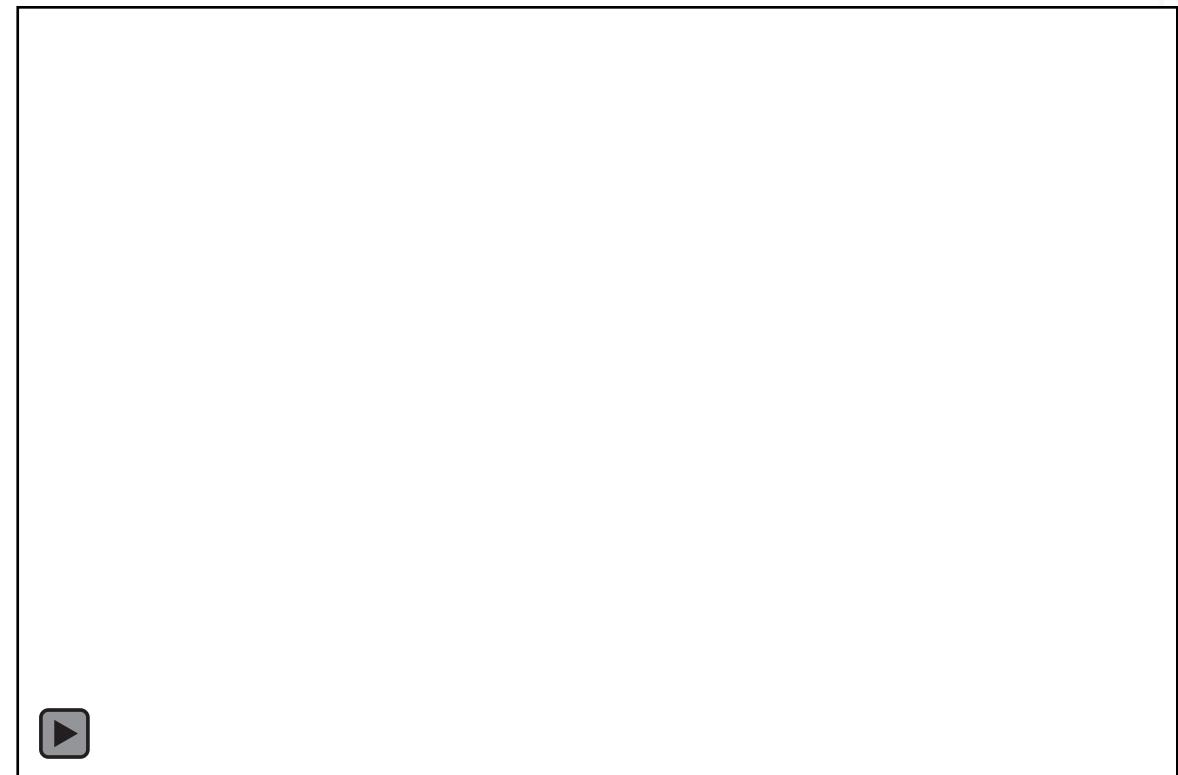
ENVIRONMENT



ENVIRONMENT – BIPEDALWALKER-V3

Caratteristiche

- Robot Bipede
- 2 gambe con 2 articolazioni
- Ricompensa massima $\cong 300$
(percorso completato)
- Ricompensa minima = -100
(robot cade)
- Spazio dello stato = 24 parametri
- Spazio delle azioni = 4 parametri
- Risoluzione problema \rightarrow Ricompensa tot media ≥ 300 per 100 episodi consecutivi



SPAZIO DI STATO E SPAZIO DELLE AZIONI

Stato

Num	Observation	Min	Max
0	Angolo scafo	0	2π
1	Velocità angolare scafo	$-\infty$	$+\infty$
2	Velocità scafo lungo x	-1	+1
3	Velocità scafo lungo x	-1	+1
4	Angolo anca gamba 1	$-\infty$	$+\infty$
5	Velocità angolare anca gamba 1	$-\infty$	$+\infty$
6	Angolo ginocchio gamba 1	$-\infty$	$+\infty$
7	Velocità angolare ginocchio gamba 1	$-\infty$	$+\infty$
8	Flag contatto gamba 1 - terreno	0	1
9	Angolo anca gamba 2	$-\infty$	$+\infty$
10	Velocità angolare anca gamba 2	$-\infty$	$+\infty$
11	Angolo ginocchio gamba 2	$-\infty$	$+\infty$
12	Velocità angolare ginocchio gamba 2	$-\infty$	$+\infty$
13	Flag contatto gamba 2 - terreno	0	1
14-23	10 parametri lidar ¹⁷	$-\infty$	$+\infty$

Azioni

Num	Action	Min	Max
0	Coppia anca gamba 1	-1	+1
1	Coppia ginocchio gamba 1	-1	+1
2	Coppia anca gamba 2	-1	+1
3	Coppia ginocchio gamba 2	-1	+1

$$state = \langle h_0, \frac{\partial h_0}{\partial t}, \frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, h_1, \frac{\partial h_1}{\partial t}, k_1, \frac{\partial k_1}{\partial t}, f_{leg_1}, h_2, \frac{\partial h_2}{\partial t}, k_2, \frac{\partial k_2}{\partial t}, f_{leg_2}, lidar \rangle$$

$$action = \langle H_1, K_1, H_2, K_2, \rangle$$



OBIETTIVO E FASI PROGETTUALI

OBIETTIVO

- Risolvere le equazioni di ottimalità di Bellman
- Trovare un algoritmo di reinforcement learning
- Raggiungere l'obiettivo di ottenere per 100 episodi consecutivi:

$$risoluzione \Rightarrow R_{100} = \frac{1}{100} \sum_{i=1}^{100} r_i \geq 300$$

FASI PROGETTUALI

Progettazione

- 1° step → Prendere confidenza con i fondamentali del reinforcement learning, sperimentando gli algoritmi basilari Q-Learning e Deep Q-Learning (DQN).
- 2° step → Prova di una prima soluzione con l'algoritmo più complesso DDPG e analisi delle problematiche.
- 3° step → Dalle problematiche individuate, ricercare un algoritmo che le risolve (TD3) e analizzare il perché le risolva.

Q-LEARNING



CARATTERISTICHE

- Model-free
- Errore temporale
- Equazione di aggiornamento

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

$\alpha \rightarrow$ learning rate

$\gamma \rightarrow$ discount rate

- Q-table che modella la Q-function
- Discretizzazione degli stati e delle azioni con tecnica a bucket

PSEUDO-CODICE Q-LEARNING

Algorithm 2 Q-Learning (off-policy TD)

```
1: inizializza i parametri:  $\alpha \in (0, 1]$ ,  $\epsilon > 0$ 
2: inizializza  $Q(s, a) \forall s \in S^+, a \in A(s)$ 
3: for all episode do
4:    $S \leftarrow$  stato iniziale
5:   for all steps in episode do
6:     scegli  $A$  da  $S$  usando la politica derivata da  $Q$  ( $\epsilon$ -greedy)
7:     seleziona  $A$ , osserva  $R, S'$ 
8:      $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$ 
9:      $S \leftarrow S'$ 
10:  end for
11: end for
```

CONFIGURAZIONI

Q-learning-1

Parametro	Valore
Dimensione Bucket Stato	(20,20,20,20,20,20,20,20,20,20,20,20,20,20,2)
Dimensione Bucket Azione	(5, 5, 5, 5)
Numero Episodi	100000
Learning Rate	0.01
Discount Rate	0.99
Exploration Rate	0.99..0

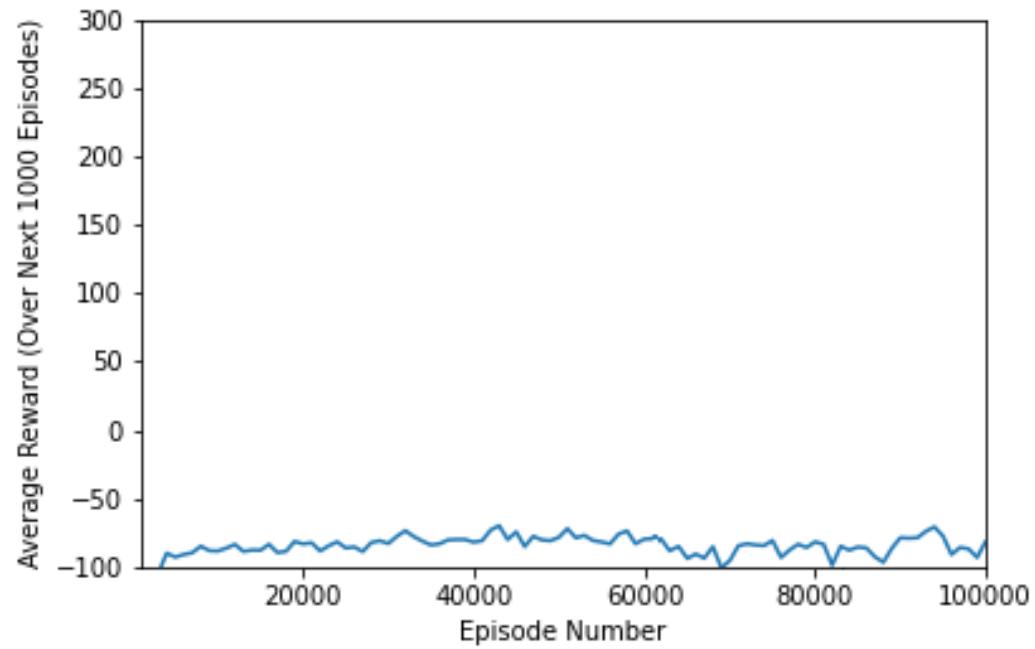
Q-learning-2

Parametro	Valore
Dimensione Bucket Stato	(4,5,5,5,4,5,4,5,2,4,5,4,5,2)
Dimensione Bucket Azione	(20,20,20,20)
Numero Episodi	100000
Learning Rate	0.01
Discount Rate	0.99
Exploration Rate	0.99..0

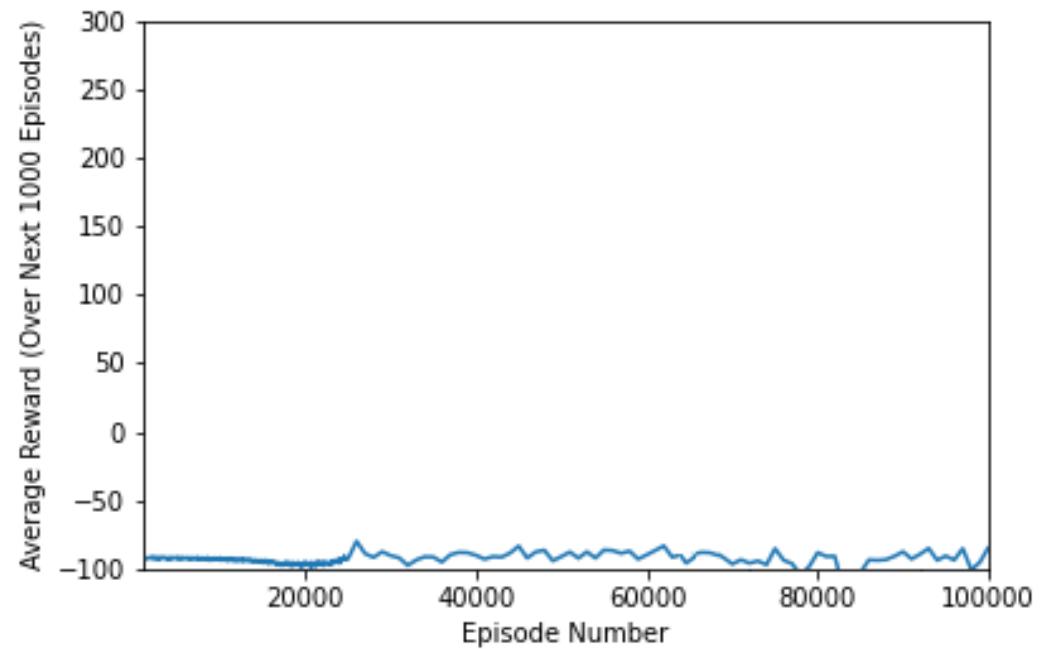
Q-LEARNING

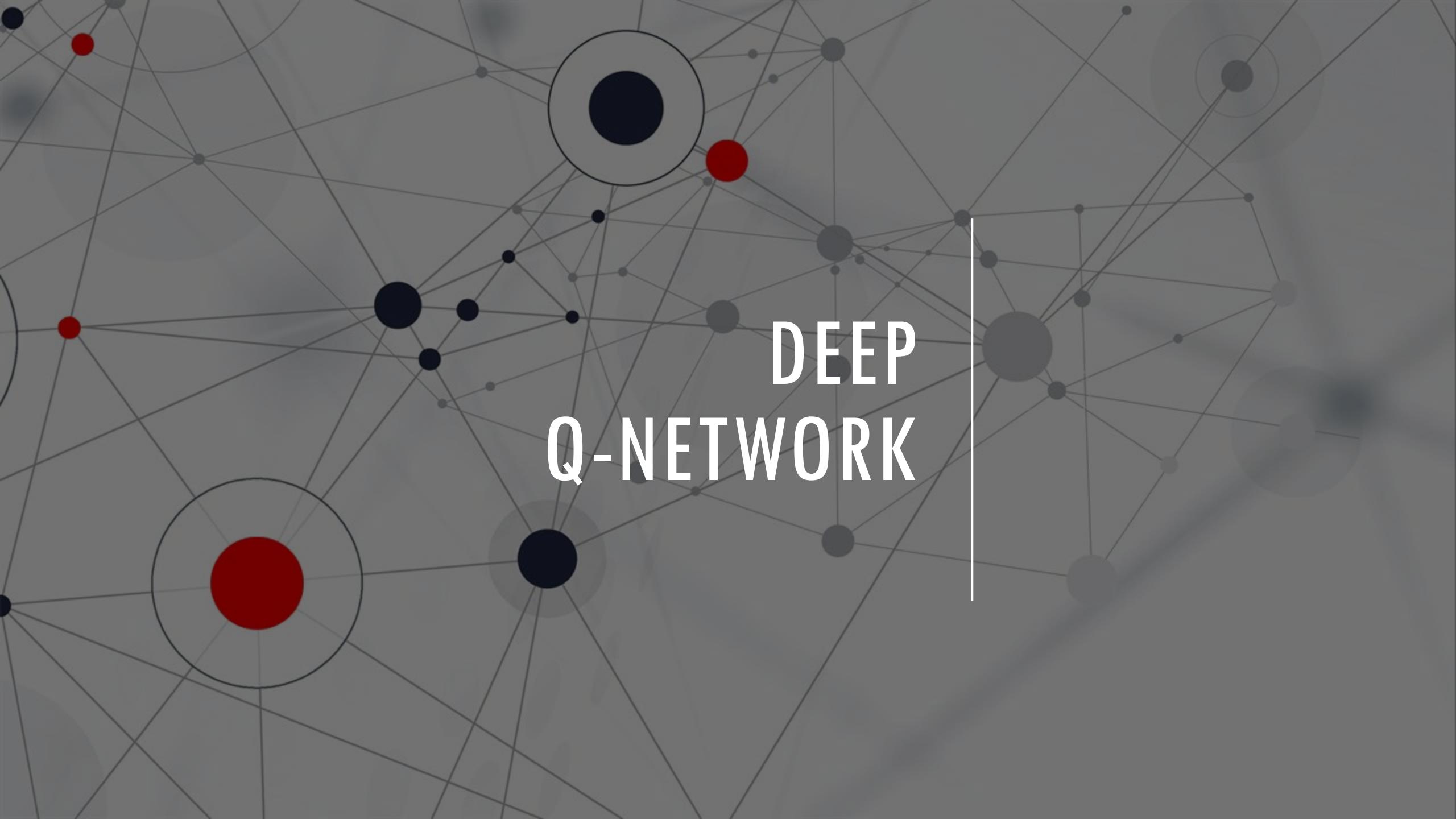
RISULTATI

Q-learning-1



Q-learning-2





DEEP Q-NETWORK

CARATTERISTICHE

- Deep Neural Network come approssimatore di Q-function
- Aggiustamento dei pesi dei neuroni attraverso backpropagation.
- Policy ϵ -greedy
- Utilizzo del Replay Buffer che memorizza esperienze (s, a, r, s')
- Utilizzo di rete target

PSEUDO-CODICE

Algorithm 3 DQN con experience replay

```

1: inizializza il buffer  $B$  di capacità  $N$ 
2: inizializza la rete azione-valore  $Q$  con i pesi casuali  $\theta$ 
3: inizializza la rete target azione-valore  $\hat{Q}$  con i pesi  $\hat{\theta} = \theta$ 
4: for  $episodio = 1, M$  do
5:   inizializza lo stato iniziale  $s_1$ 
6:   for  $t = 1, T$  do
7:     if  $\text{random}(0,1) \leq \epsilon$  then
8:       seleziona un'azione casuale  $a_t$ 
9:     else
10:      seleziona  $a_t = \arg \max_a Q(s_t, a; \theta)$ 
11:    end if
12:    esegui l'azione  $a_t$  e osserva la ricompensa  $r_t$  e lo stato  $s_{t+1}$ 
13:    memorizza l'esperienza  $(s_t, a_t, r_t, s_{t+1})$  in  $B$ 
14:    estrai un minibatch di esperienze  $(s_j, a_j, r_j, s_{j+1})$  da  $B$ 
15:     $y_t = r_t + (1 - \text{done}) \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \hat{\theta})$ 
16:    esegui il gradiente discendente della MSE  $(y_j - Q(s_j, a_j; \theta))^2$  rispetto a  $\theta$ 
17:    if  $t \bmod C = 0$  then
18:      reset  $\hat{Q} = \tau Q + (1 - \tau)\hat{Q}$ 
19:    end if
20:     $s_t = s_{t+1}$ 
21:    if  $\text{done}$  then
22:      print("Episode finished")
23:      break
24:    end if
25:  end for
26: end for

```

FUNZIONE DI LOSS

- Definita come MSE tra i valori target e i valori stimati.

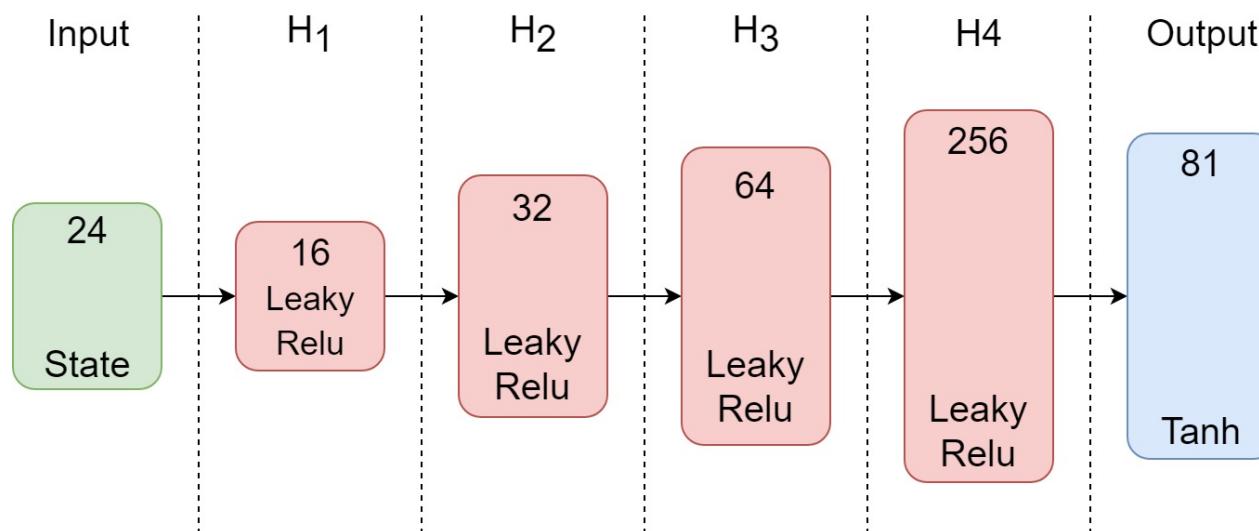
$$L_t(\theta) = \frac{1}{n} \sum_{i=1}^n \left[r + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \hat{\theta}) - Q(s_t, a; \theta) \right]^2$$

Dove:

$r + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \hat{\theta})$ rappresenta l'expected return ottimo (rete target)

$Q(s_t, a; \theta)$ rappresenta il valore stimato dalla rete

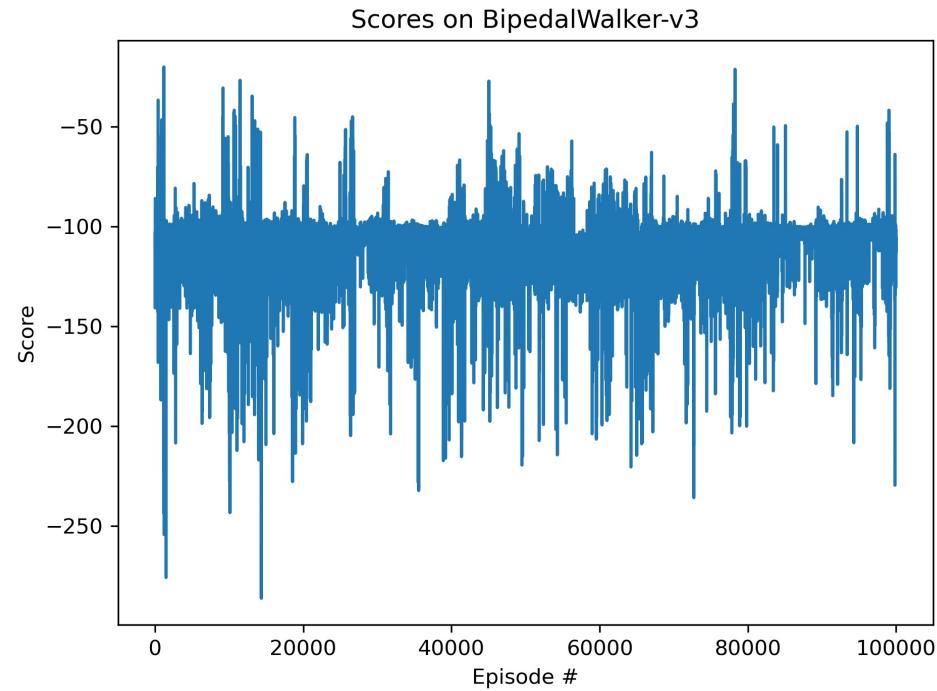
MODELLO RETE E CONFIGURAZIONE



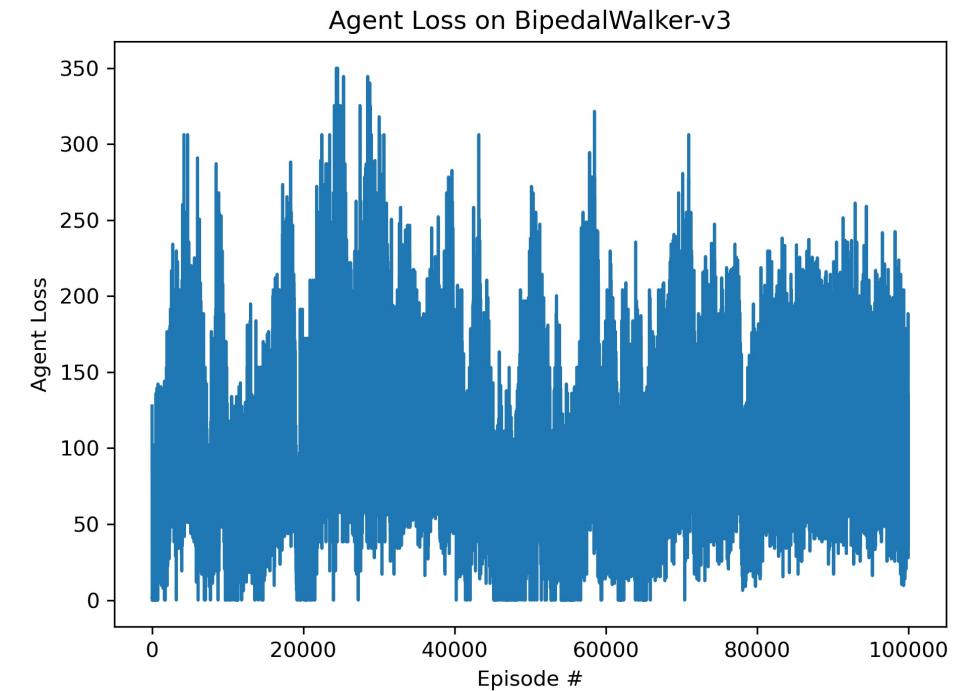
Parametro	Valore
Dimensione Replay Buffer	100000
Dimensione Mini Batch	64
Discount Rate γ	0.99
Interpolation Parameter τ	0.01
Learning Rate	0.05
Exploration Rate ϵ	0.99..0.01
ϵ Decay	0.01
Numero Step per Aggiornamento C	4
Numero Azioni	81

RISULTATI

Ricompensa Totale

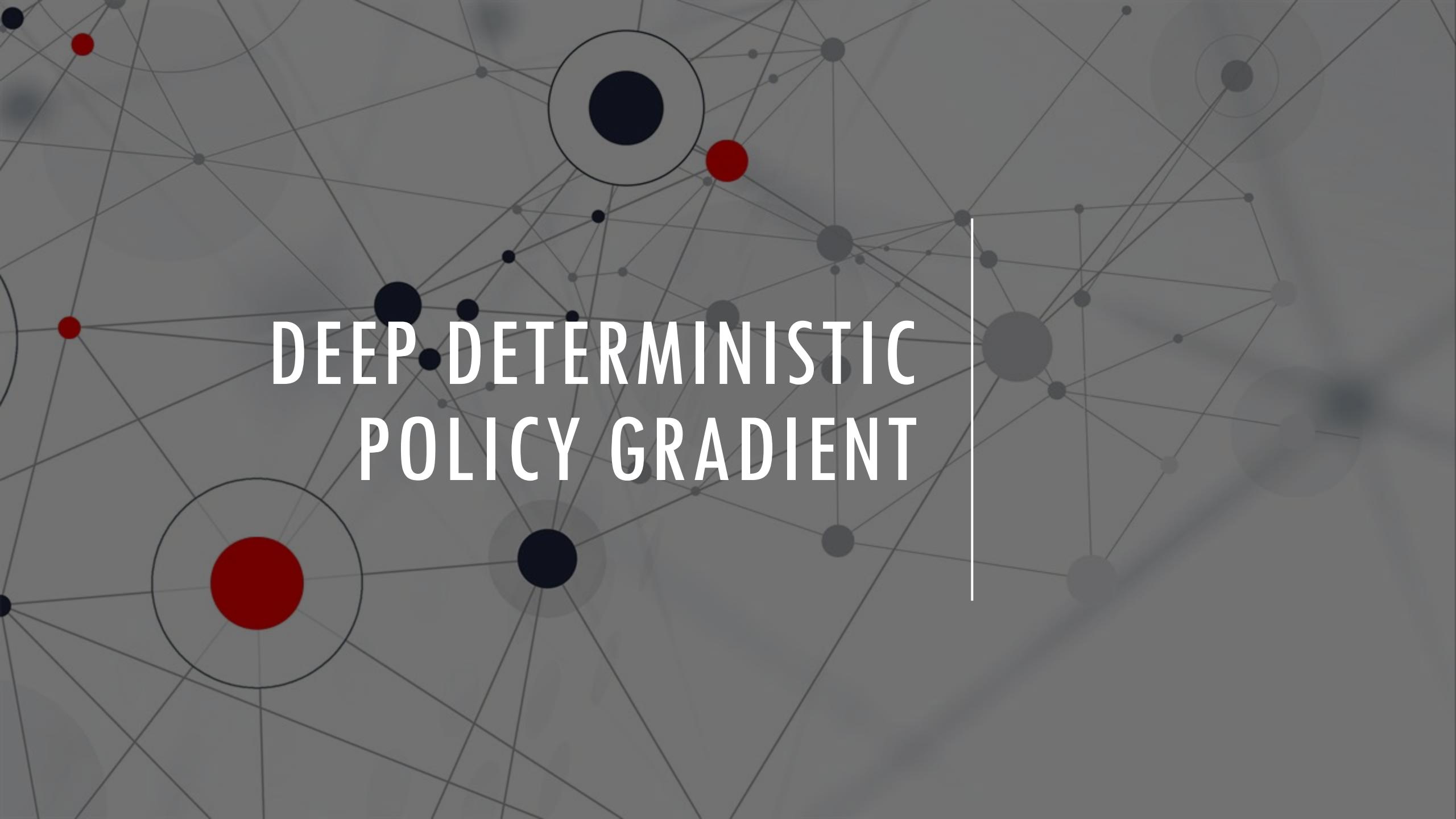


Loss



PROBLEATICHE

- Q-function complessa
- Apprendimento complicato
- Non si raggiunge la convergenza
- Discretizzazione delle azioni



DEEP DETERMINISTIC POLICY GRADIENT

CARATTERISTICHE

- Combinazione tra l'algoritmo DQN e la tecnica Policy Gradient

$$\nabla \mathbb{E}_\pi[r(\tau)] = \mathbb{E}_\pi \left[G_t \cdot \left(\sum_{t=1}^T \nabla \log \pi(a_t | s_t) \right) \right]$$

- Utilizza l'architettura Actor-Critic

$$\nabla \mathbb{E}_\pi[r(\tau)] = \mathbb{E}_\pi \left[\sum_{t=1}^T \nabla \log \pi(a_t | s_t) Q(s_t, a_t) \right]$$

ricordando che $\mathbb{E}_\pi[G_t] = Q(s_t, a_t)$

- No discretizzazione delle azioni
- Rumore gaussiano per le azioni

FUNZIONE DI LOSS

- La funzione di loss per la rete critic- Q Definita come MSE tra i valori target e i valori stimati.

$$J_Q = \frac{1}{N} \sum_i (r_i + \gamma(1-d)Q'(s_{i+1}, \mu'(s_{i+1})) - Q(s_i, a_i))^2$$

- La funzione di loss per la rete actor- μ è definita come:

$$J_\mu = \frac{1}{N} \sum_{i=1}^N Q(s_i, \mu(s_i))$$

- Gradiente per aggiornamento della rete actor

$$\nabla_{\theta^\mu} J_\mu \approx \frac{1}{N} \sum_i (\nabla_\mu Q(s_i, \mu(s_i)) \nabla_{\theta^\mu} \mu(s_i))$$

PSEUDO-CODICE

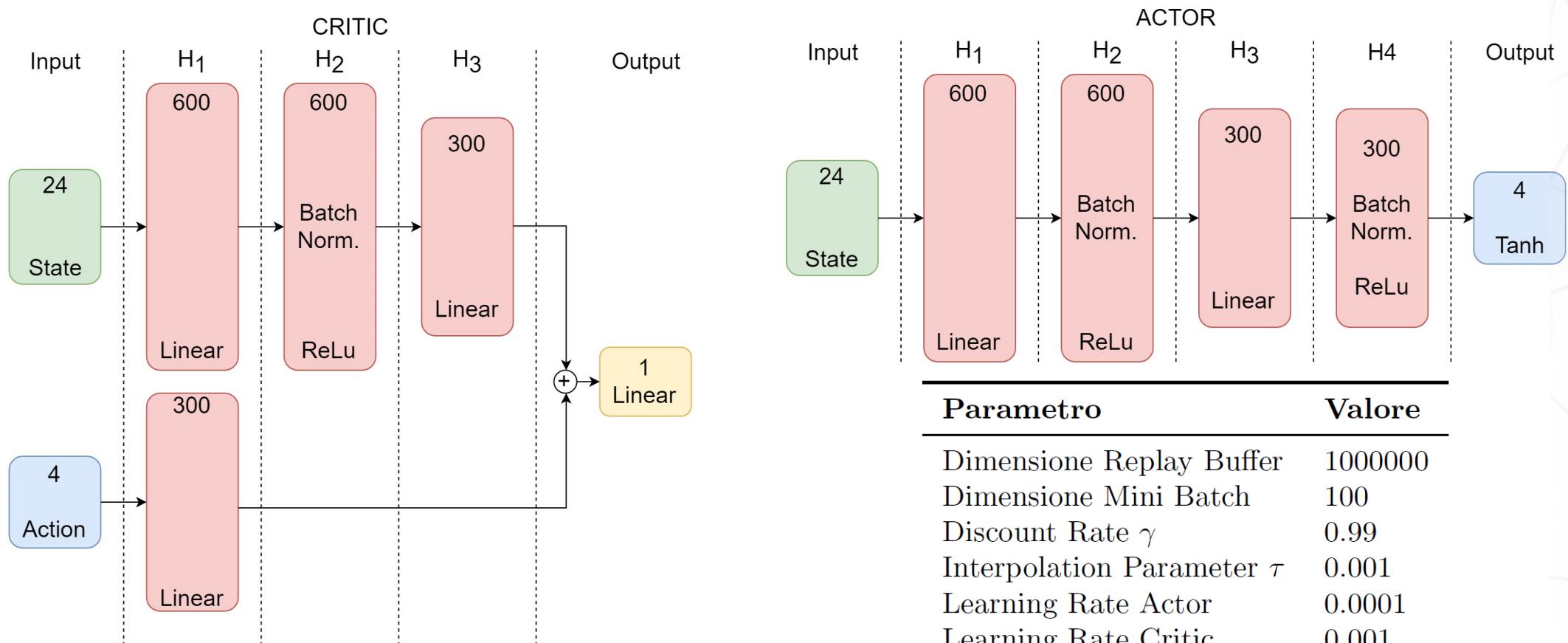
Algorithm 4 DDPG

```

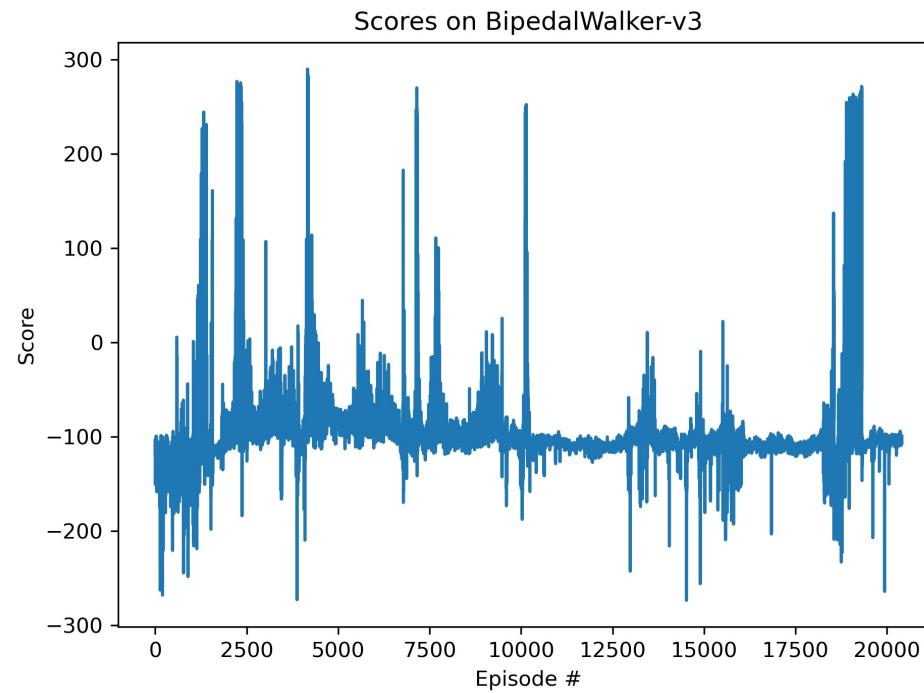
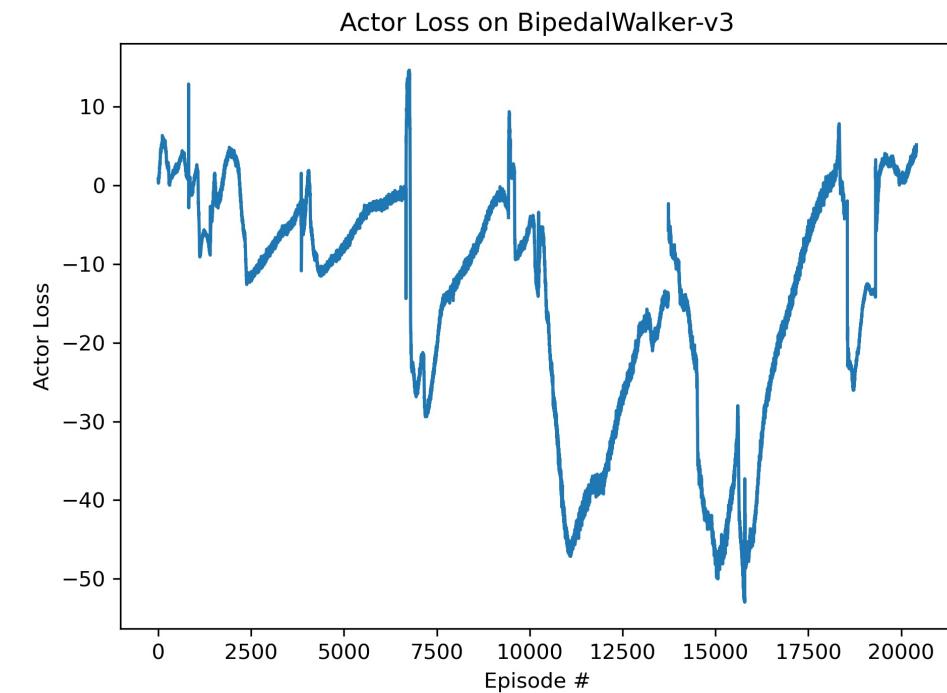
1: inizializza il buffer  $R$  di capacità  $M$ 
2: inizializza la rete critic  $Q(s, a|\theta^Q)$  con pesi casuali  $\theta^Q$ 
3: inizializza la rete actor  $\mu(s|\theta^\mu)$  con pesi casuali  $\theta^\mu$ 
4: inizializza le reti target  $Q'$  e  $\mu'$  con i pesi  $\theta^{Q'} \leftarrow \theta^Q$ ,  $\theta^{\mu'} \leftarrow \theta^\mu$ 
5: for  $episodio = 1, M$  do
6:   inizializza un processo random  $\mathcal{N}$  per l'esplorazione delle azioni
7:   ricevi lo stato iniziale  $s_1$ 
8:   for  $t = 1, T$  do
9:     seleziona l'azione  $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ 
10:    esegui l'azione  $a_t$  e osserva la ricompensa  $r_t$  e il nuovo stato  $s_{t+1}$ 
11:    memorizza l'esperienza  $(s_t, a_t, r_t, s_{t+1})$  in  $R$ 
12:    estrai un minibatch di  $N$  esperienze  $(s_i, a_i, r_i, s_{i+1})$  da  $R$ 
13:     $y_i = r_i + \gamma(1 - d)Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$ 
14:    aggiorna la rete critic- $Q$  minimizzando la loss:  $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ 
15:    aggiorna la rete actor- $\mu$  usando il gradiente della politica deterministica  $\nabla_{\theta^\mu} \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i}$ 
16:    aggiorna le reti target:  $\theta^{Q'} \leftarrow \tau\theta^Q + (1 - \tau)\theta^{Q'}$   $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1 - \tau)\theta^{\mu'}$ 
17:     $s_t = s_{t+1}$ 
18:    if  $d$  then
19:      print("Episode finished")
20:      break
21:    end if
22:  end for
23: end for

```

MODELLO RETE E CONFIGURAZIONE

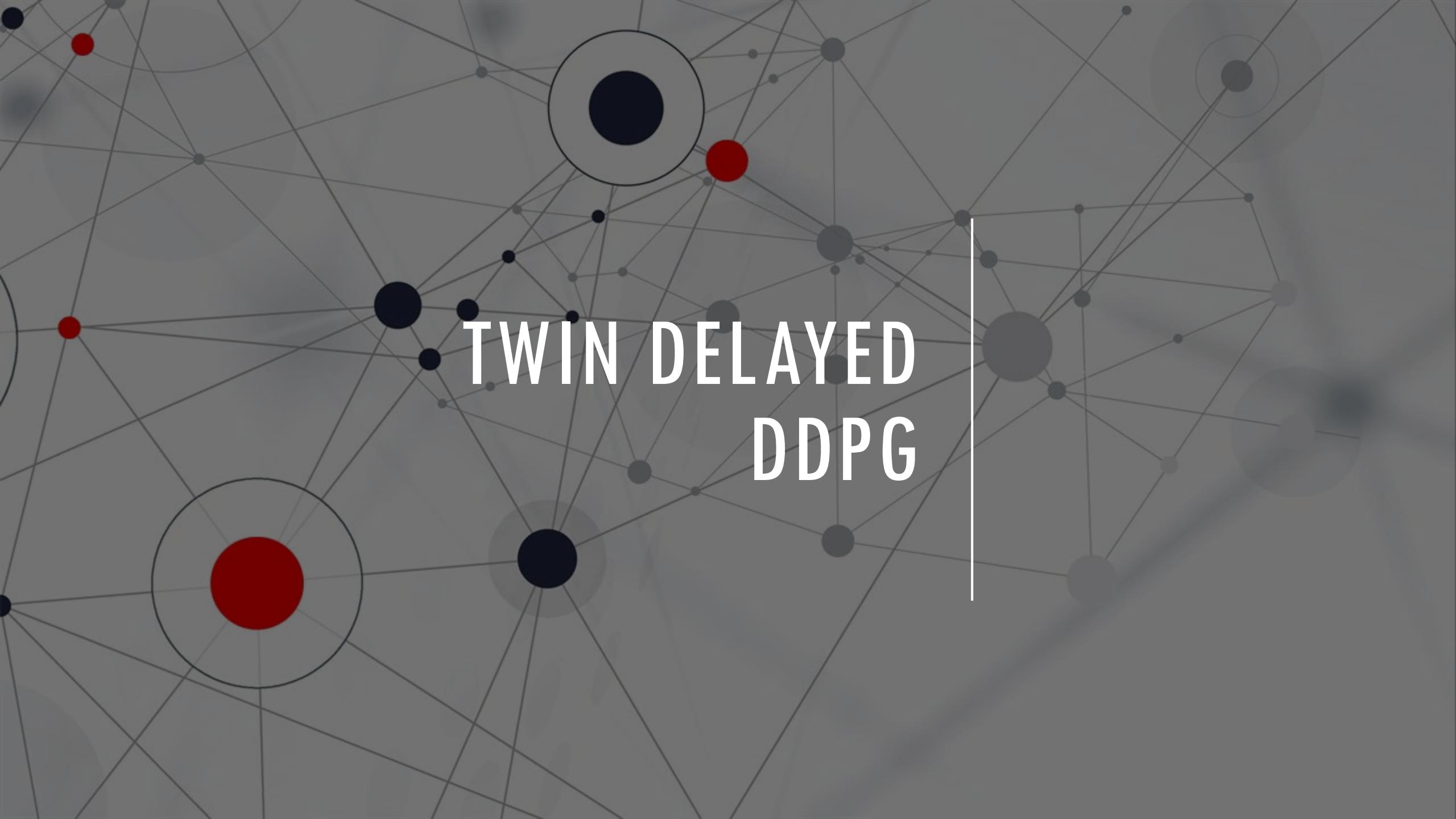


RISULTATI - 1

Ricompensa Totale**Actor Loss**

PROBLEMATICHE

- Non si raggiunge la convergenza
- Tempo di training elevato
- Dipendenza dai parametri ipertestuali
- Accumulo degli errori di stima
- Correlazione tra rete actor e critic
- Elevata varianza tra i valori target



TWIN DELAYED DDPG



TD3

CARATTERISTICHE

- Uso di due reti critiche gemelle
- Aggiornamenti ritardati della rete actor-policy
- Tecnica di smoothing

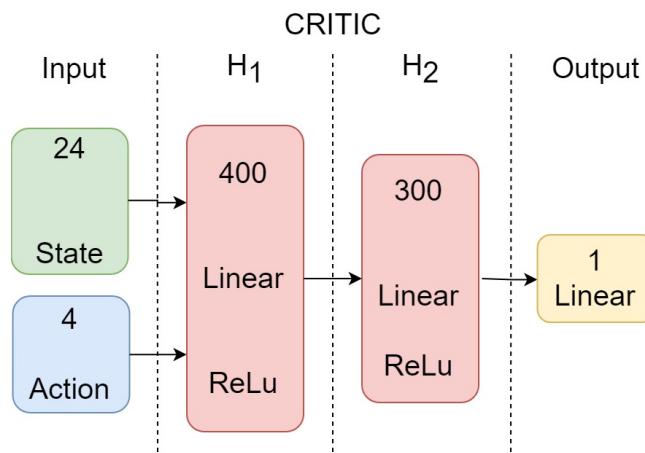
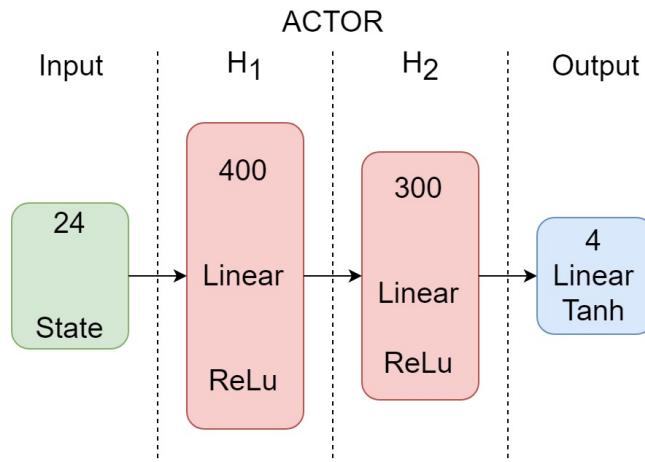
PSEUDO-CODICE

Algorithm 5 TD3

```

1: inizializza il buffer  $R$  di capacità  $M$ 
2: inizializza le reti critic  $Q_{\theta_1}, Q_{\theta_2}$  con pesi casuali  $\theta_1, \theta_2$ 
3: inizializza la rete actor  $\mu_\phi$  con pesi casuali  $\phi$ 
4: inizializza le reti target  $Q'_{\theta'_1}, Q'_{\theta'_2}$  e  $\mu'_{\phi'}$  con i pesi  $\theta'_1 \leftarrow \theta_1$ ,  $\theta'_2 \leftarrow \theta_2$  e  $\phi' \leftarrow \phi$ 
5: for  $episodio = 1, M$  do
6:   inizializza un processo random  $\epsilon \sim \mathcal{N}(0, \sigma)$  per l'esplorazione delle azioni
7:   ricevi lo stato iniziale  $s_1$ 
8:   for  $t = 1, T$  do
9:     seleziona l'azione  $a_t = \mu_\phi(s_t) + \epsilon$ 
10:    esegui l'azione  $a_t$  e osserva la ricompensa  $r_t$  e il nuovo stato  $s_{t+1}$ 
11:    memorizza l'esperienza  $(s_t, a_t, r_t, s_{t+1})$  in  $R$ 
12:     $s_t = s_{t+1}$ 
13:    if  $d$  then
14:      print("Episode finished")
15:      break
16:    end if
17:   end for
18:   for  $n = 1, t$  do
19:     estrai un minibatch di  $N$  esperienze  $(s_i, a_i, r_i, s_{i+1})$  da  $R$ 
20:     aggiungi rumore all'azione in input alle reti critiche  $\tilde{a} = \mu'_{\phi'}(s_{t+1}) + \epsilon$ 
21:      $y_i = r_i + \gamma(1 - d) \min_{j=1,2} Q'_{\theta'_j}(s_{i+1}, \tilde{a})$ 
22:     aggiorna la rete critic  $Q_{\theta_j} \forall j = 1, 2$  minimizzando le loss:  $L_j = \frac{1}{N} \sum_i (y_i - Q_{\theta_j}(s_i, a_i))^2$ 
23:     if  $n \bmod policydelay = 0$  then
24:       aggiorna la rete actor- $\mu$  usando il gradiente della politica deterministica  $\nabla_\phi \approx \frac{1}{N} \sum_i \nabla_a Q_{\theta_1}(s, a)|_{s=s_i, a=\mu_\phi(s_i)} \nabla_\phi \mu_\phi(s)|_{s=s_i}$ 
25:       aggiorna le reti target:  $\theta'_j \leftarrow \tau \theta_j + (1 - \tau) \theta'_j \forall j = 1, 2$   $\phi' \leftarrow \tau \phi + (1 - \tau) \phi'$ 
26:     end if
27:   end for
28: end for
  
```

MODELLO RETE E CONFIGURAZIONE

**Conf 1**

Livello	Nodi
Actor H1	400
Actor H2	300
Critic H1	400
Critic H2	300

Conf 2

Livello	Nodi
Actor H1	480
Actor H2	240
Critic H1	560
Critic H2	280

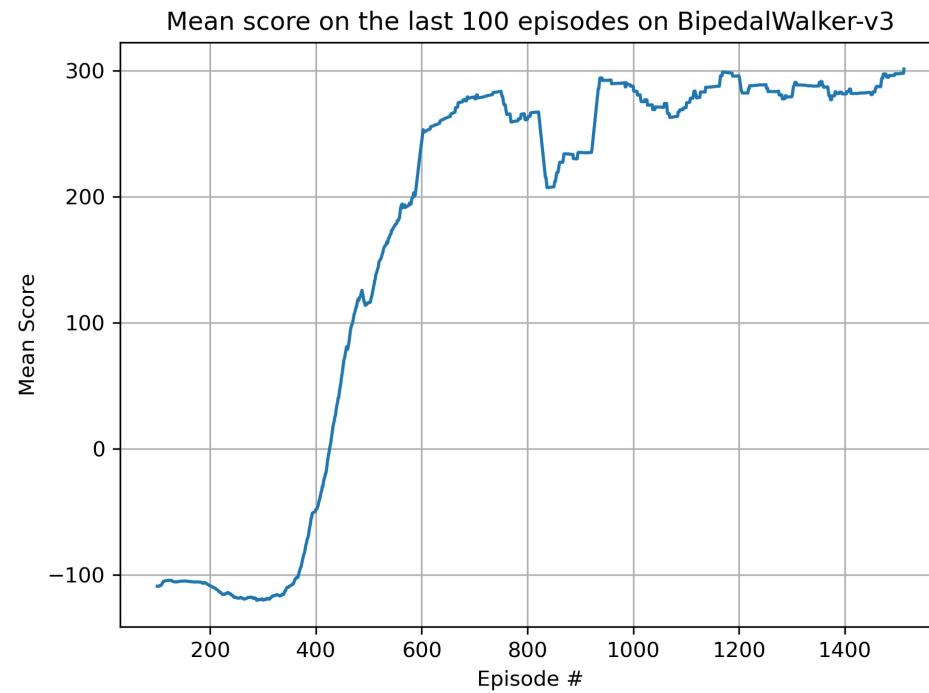
Parametro	Valore
-----------	--------

Dimensione Replay Buffer	500000
Dimensione Mini Batch	100
Discount Rate γ	0.99
Interpolation Parameter τ	0.005
Learning Rate Actor	0.001
Learning Rate Critic	0.001
Policy Delay	2
Policy Noise	(-0.5,0.5)

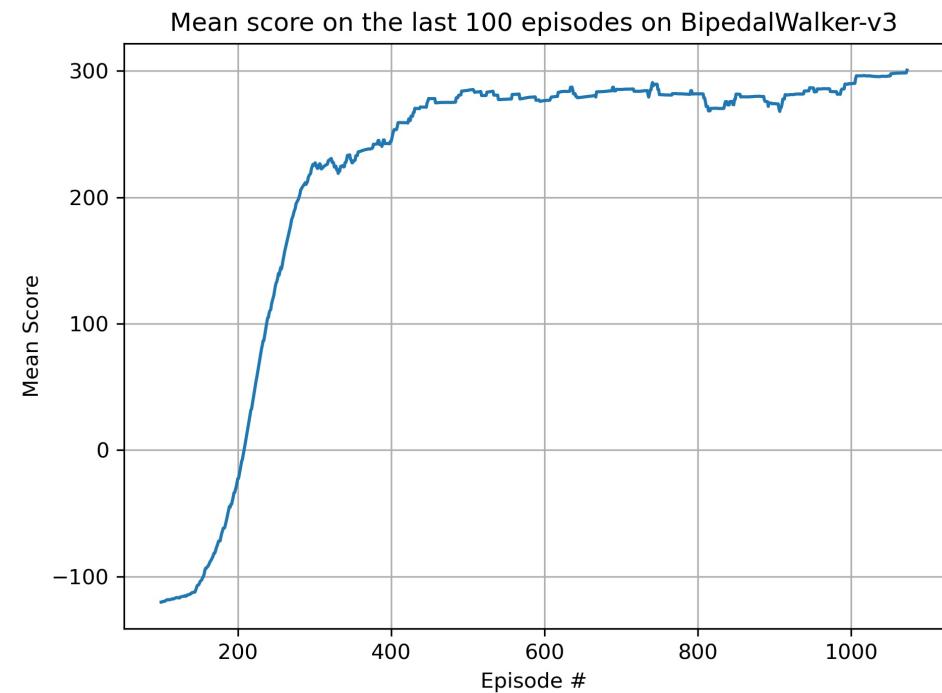
RISULTATI - 1

Ricompensa media sugli ultimi 100 episodi

Conf 1



Conf 2



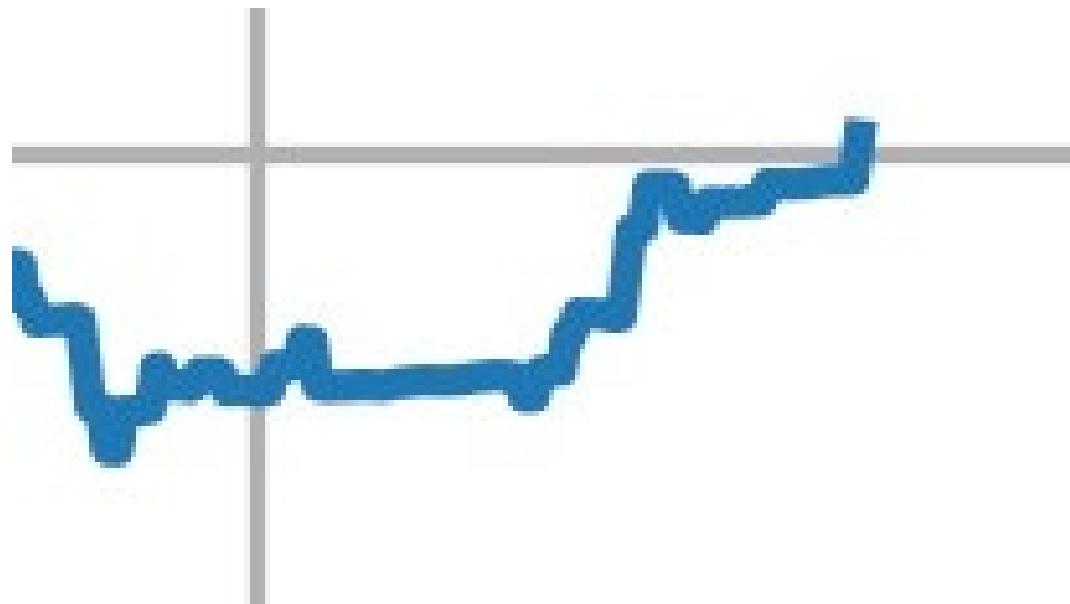


TD3

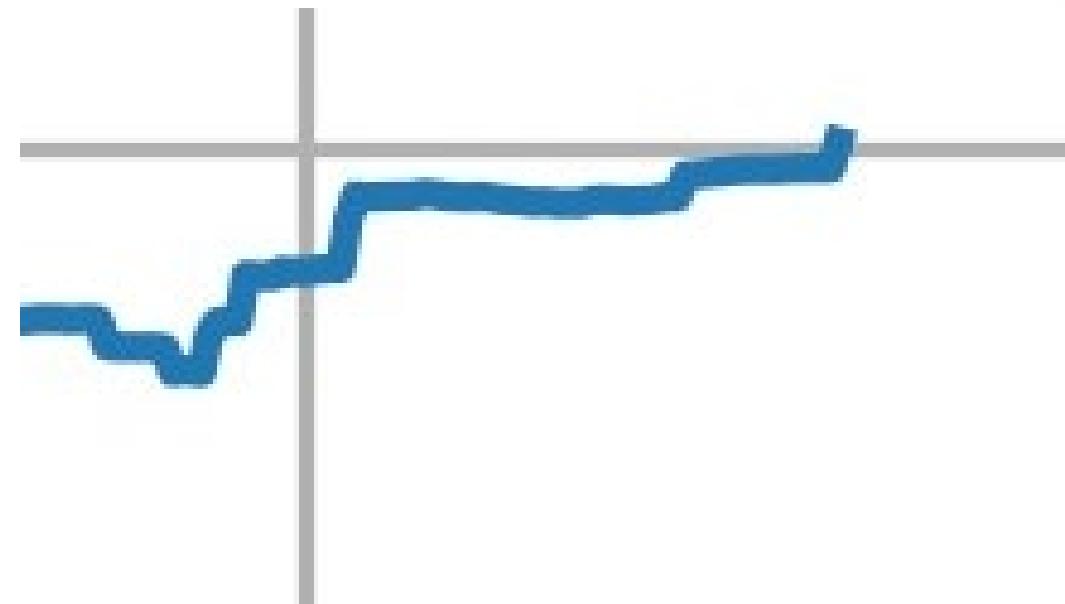
RISULTATI - 1

Ricompensa totale media sugli ultimi 100 episodi

Conf 1



Conf 2

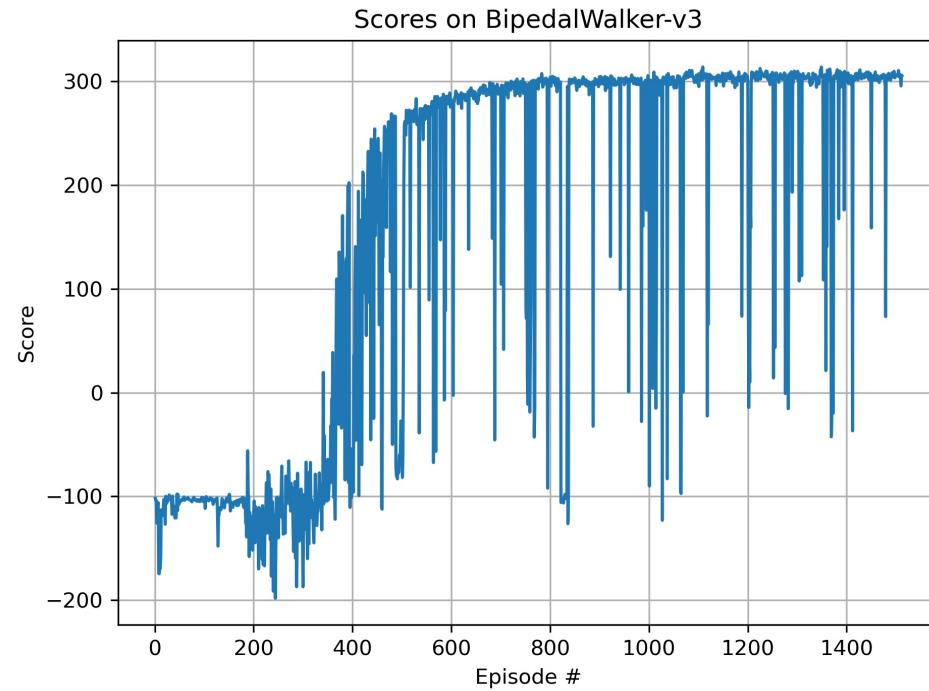


TD3

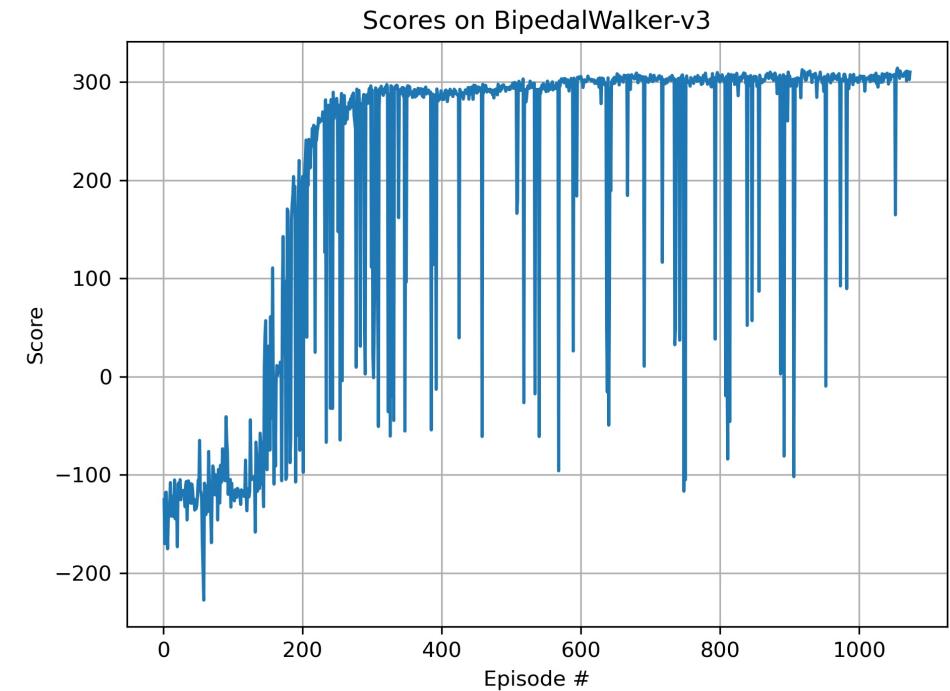
RISULTATI - 1

Ricompensa totale

Conf 1



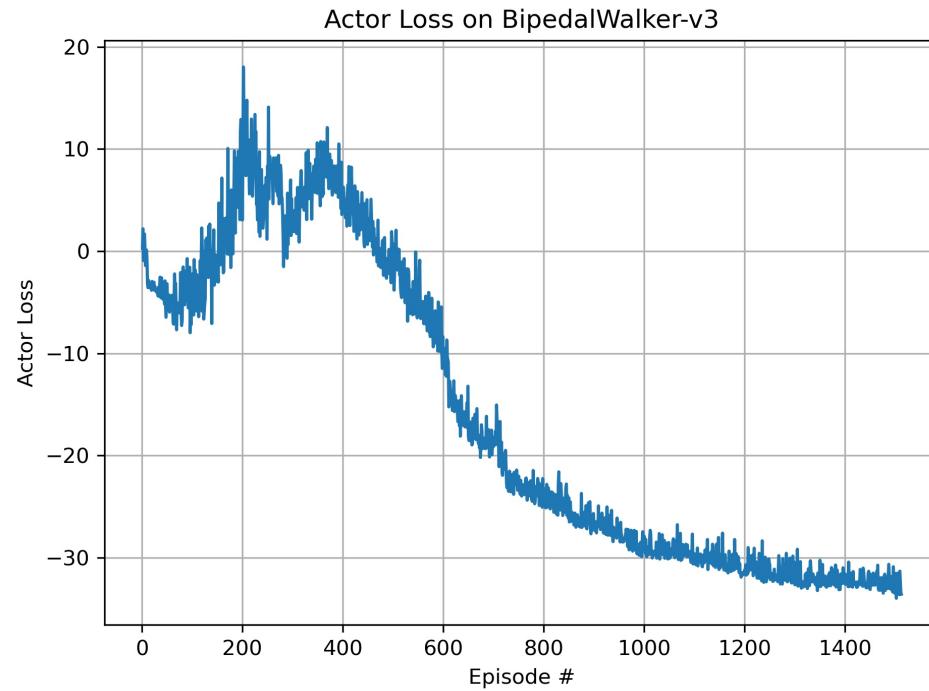
Conf 2



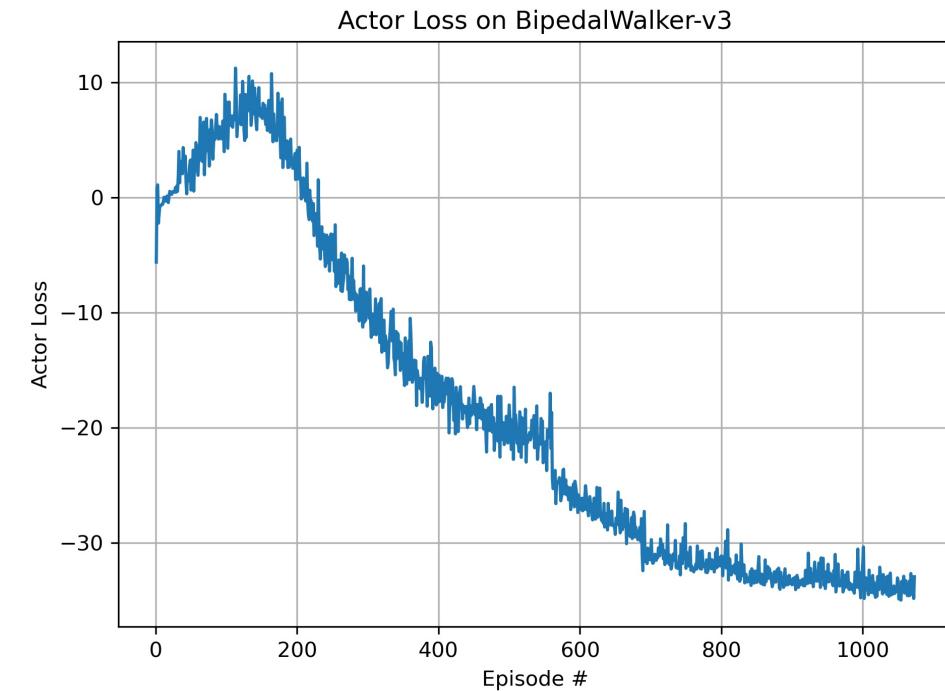
TD3

RISULTATI - 2

Actor Loss – Conf 1



Actor Loss – Conf 2



CONCLUSIONI

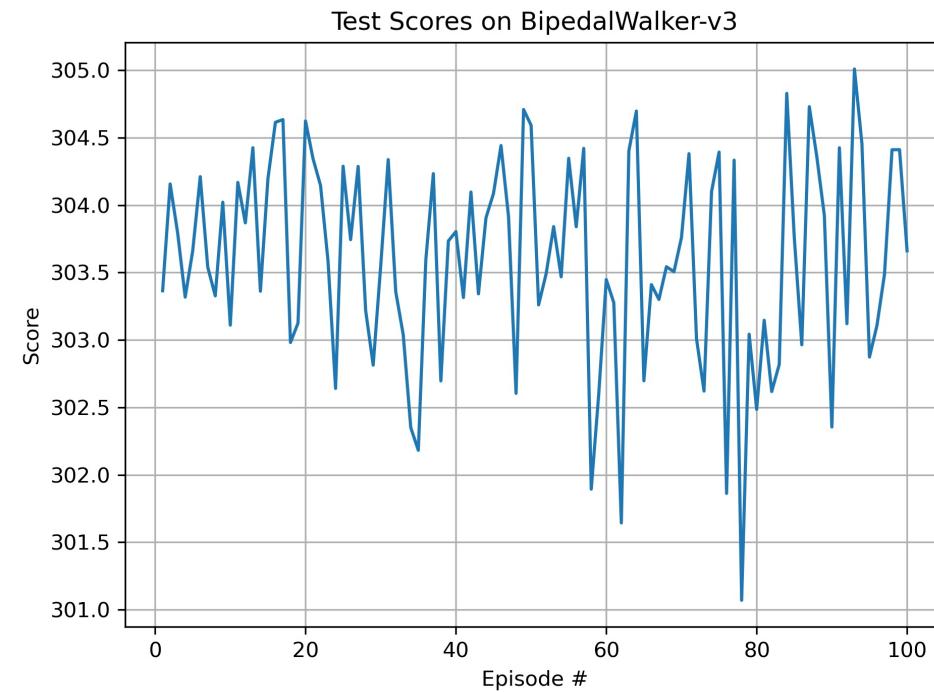


TEST

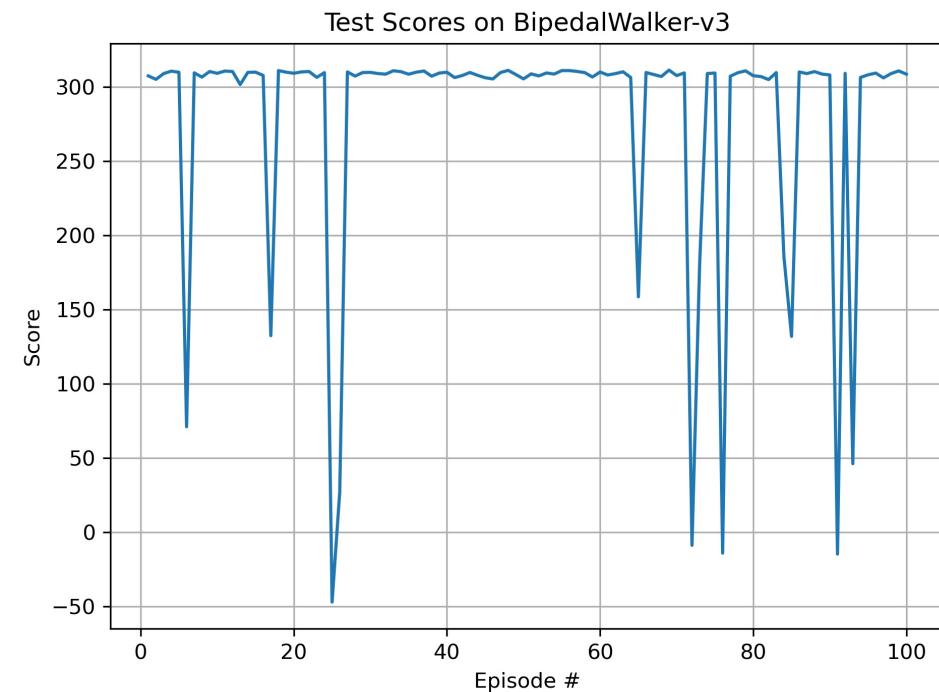
TD3

Ricompensa su 100 episodi consecutivi

Conf 1 – Media = 303.71



Conf 2 - Media = 280.28



CONCLUSIONI

Algoritmo	Score Medio Max	Score Massima	Episodi	Tempo in ore
Q-Learning	-76	-50	10000	12
DQN	-100	-20	10000	8
DDPG	43	280	2000	100
TD3	303	305	1512	12



GRAZIE PER L'ATTENZIONE |