



UNIVERSITÀ
DEGLI STUDI
FIRENZE

DeepCrawl

Deep Reinforcement Learning per lo Sviluppo di
Videogiochi Strategici Turn-Based

Relatori:

Prof. Andrew D. Bagdanov

Prof. Marco Bertini

Correlatori:

Dott. Alexander Kuhnle

Candidato:

Alessandro Sestini

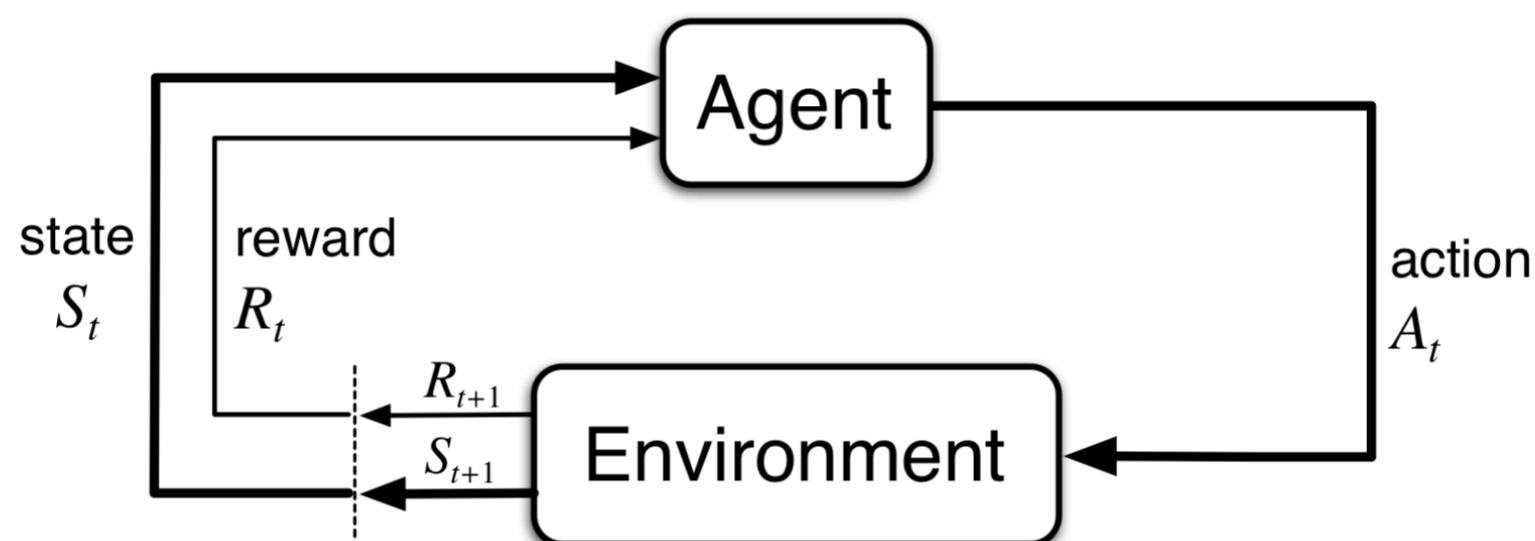
A.A. 2017-2018

Introduzione

- Uno dei problemi principali nell'industria videoludica è la **creazione di AI** per personaggi non giocanti.
- Lo sviluppo del **Deep Reinforcement Learning (DRL)** offre un nuovo modo per definire degli agenti nel dominio dei videogiochi.
- **Obiettivo:** creare un prototipo di videogioco in cui i personaggi non giocanti interagiscono con l'utente attraverso tecniche di DRL.

Reinforcement Learning

- Il RL è un metodo di allenamento di agenti attraverso delle **ricompense** senza bisogno di specificare come deve essere raggiunto l'obiettivo.
- Il compito dell'agente è trovare una **policy** che massimizzi il **future discounted reward**.



Reinforcement Learning

- Si definisce **future discounted reward** come:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

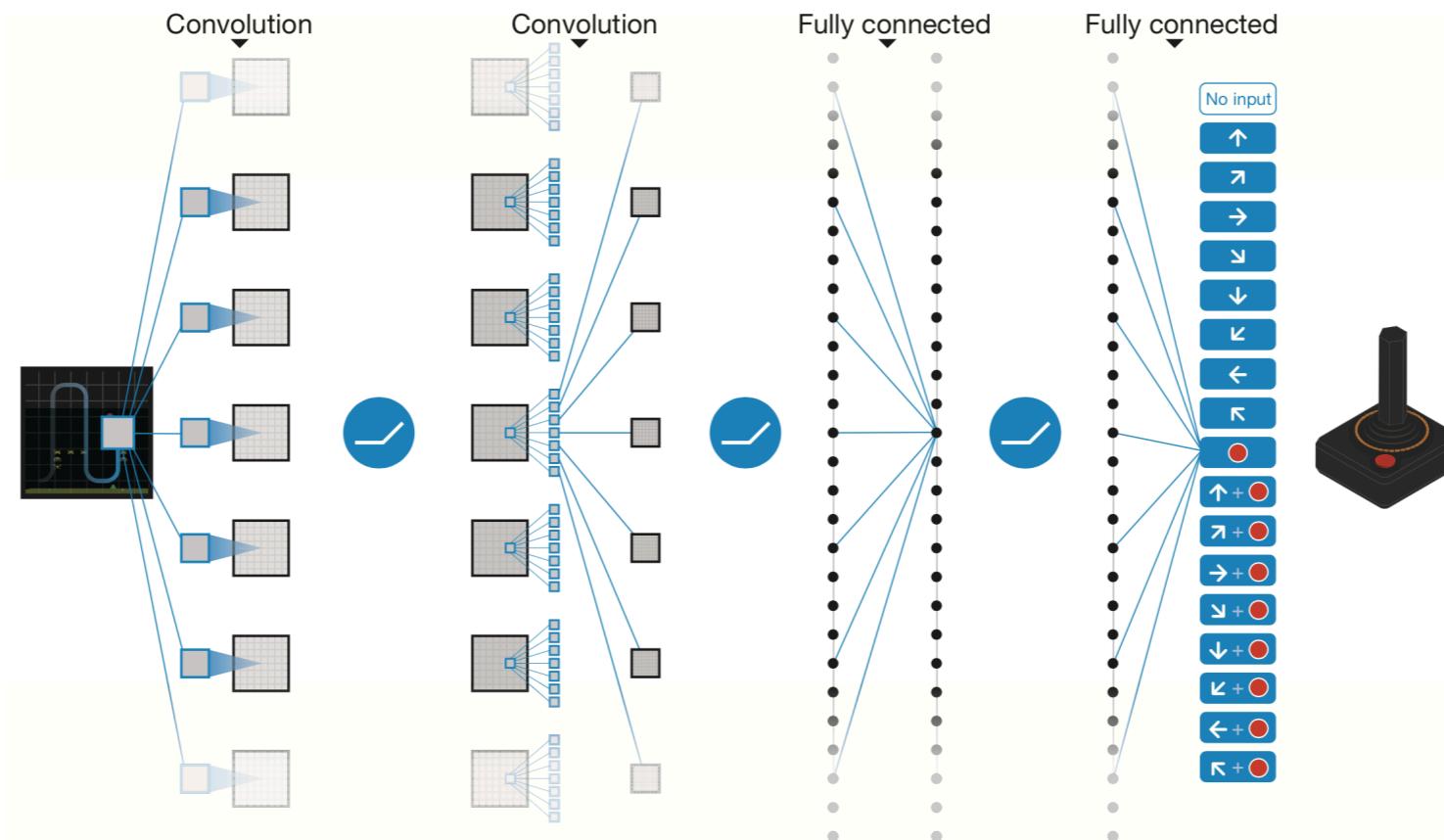
- La **policy** rappresenta la mappatura dagli stati alla distribuzione di probabilità sulle azioni:

$$\pi : S \rightarrow p(A = a | S)$$

$$\pi^* = \operatorname*{argmax}_{\pi} \mathbb{E}[G_t | \pi]$$

Deep Reinforcement Learning

- Il DRL utilizza le proprietà delle **reti neurali** per approssimare la policy per problemi ad alta dimensionalità.



Rete neurale per l'algoritmo DQ-L¹, sviluppato da DeepMind

¹ Mnih, 2015

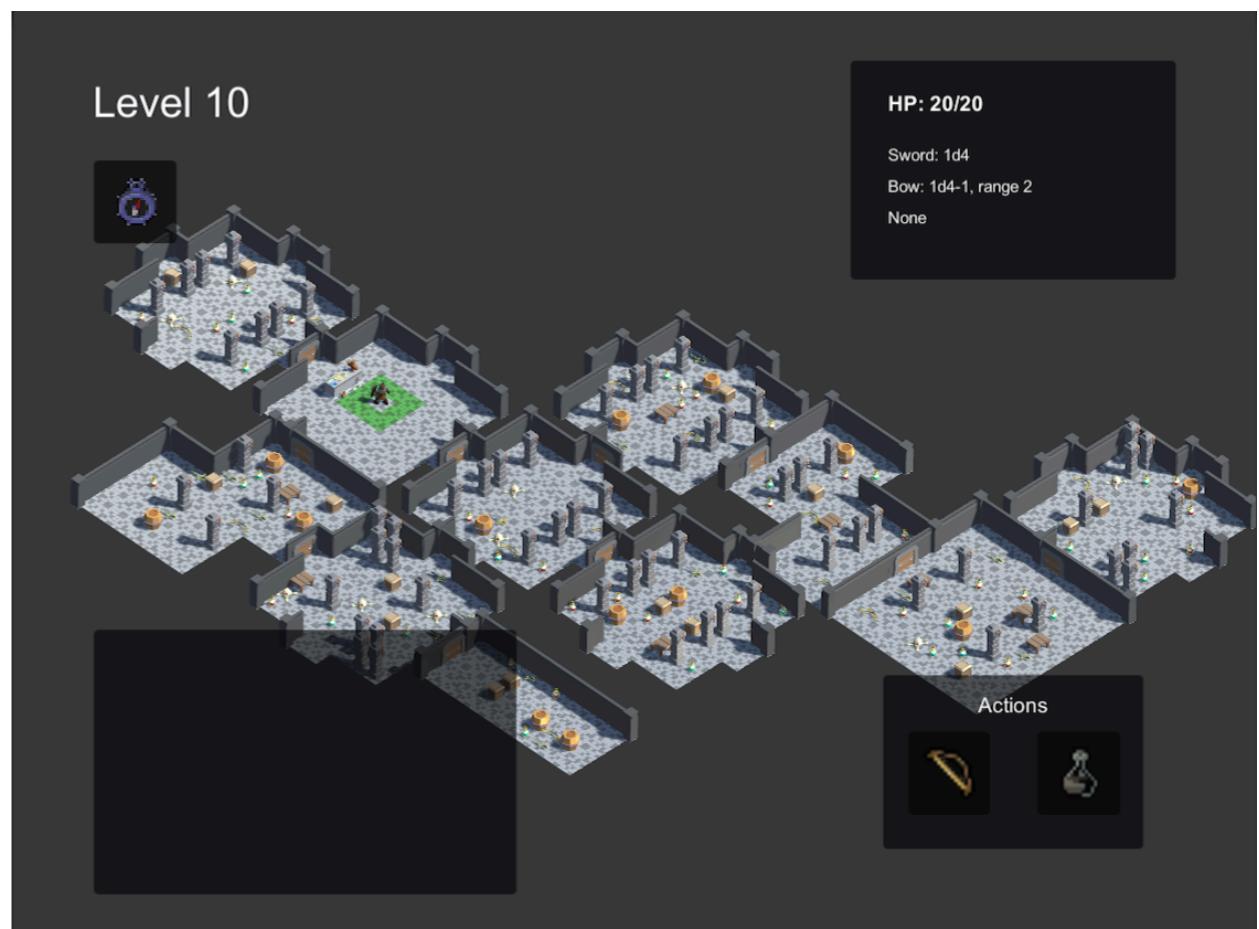
Game Design

- È stato sviluppato un *RogueLike*, un videogioco **strategico a turni** in cui il giocatore deve sfidare dei nemici all'interno di mappe procedurali.



Game Design

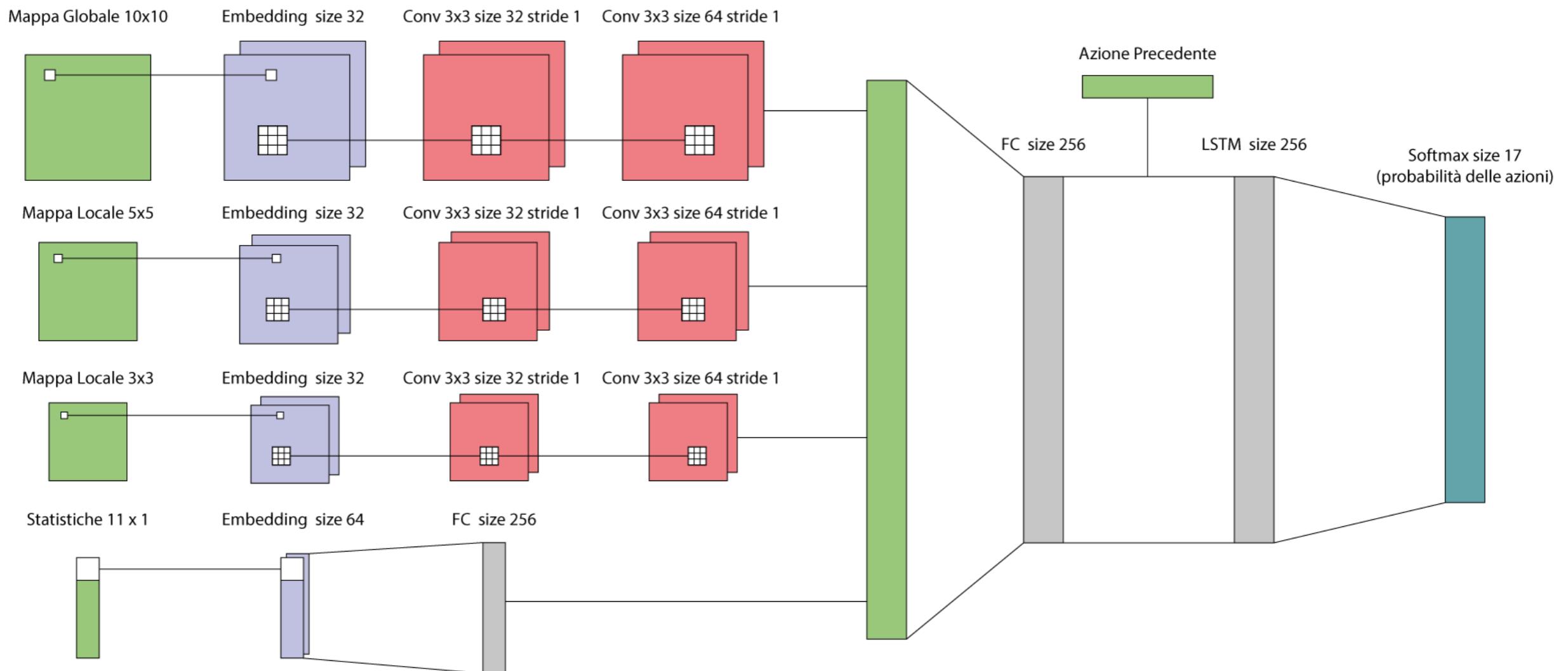
- **DeepCrawl** presenta una struttura a livelli, caratterizzati da labirinti procedurali composti da un numero di stanze crescente.
- Il raggio di azione degli agenti è limitato alla stanza di generazione.
- Il prototipo è stato implementato in **Unity** e C# per dispositivi iOS e Android.



Desiderata

- Gli agenti devono essere **credibili**, devono essere percepiti come intelligenti e devono rappresentare una sfida per l'utente.
- Gli agenti devono essere **imperfetti**, la sfida deve essere correttamente proporzionata per permettere all'utente di poter vincere in tutte le situazioni.
- Gli agenti devono essere **model-free**, devono ricavare autonomamente la giusta strategia per poter risolvere l'ambiente in ogni condizione.
- Il sistema deve offrire **varietà**, deve avere dei parametri che una volta modificati permettano agli agenti di imparare diverse strategie.

Rete Neurale



Reward Function

- È stata scelta una funzione che fosse più **sparsa** possibile per permettere agli agenti di estrapolare autonomamente la strategia adatta.
- Il termine ***hp*** rappresenta la quantità normalizzata dei punti vita rimasti: vincere con il maggior numero di hp è l'obiettivo implicito dei *RogueLike*.

$$R(t) = -0.01 + \begin{cases} -0.1 \text{ per un movimento impossibile} \\ +10.0 * hp \text{ per la vittoria} \end{cases}$$

Proximal Policy Optimization

- Sviluppato da **OpenAI**² rappresenta lo stato dell'arte nell'ambito dei videogiochi.
- È un metodo Actor-Critic che utilizza due funzioni che devono essere imparate: la **policy** (l'attore) e la **baseline** (il critico).
- La sua particolarità consiste nel **limitare gli aggiornamenti** per non modificare troppo la policy rispetto alla strategia precedente.
- Per l'implementazione è stato usato **TensorForce**³: un framework Python per algoritmi DRL che permette la libera configurazione dei parametri.

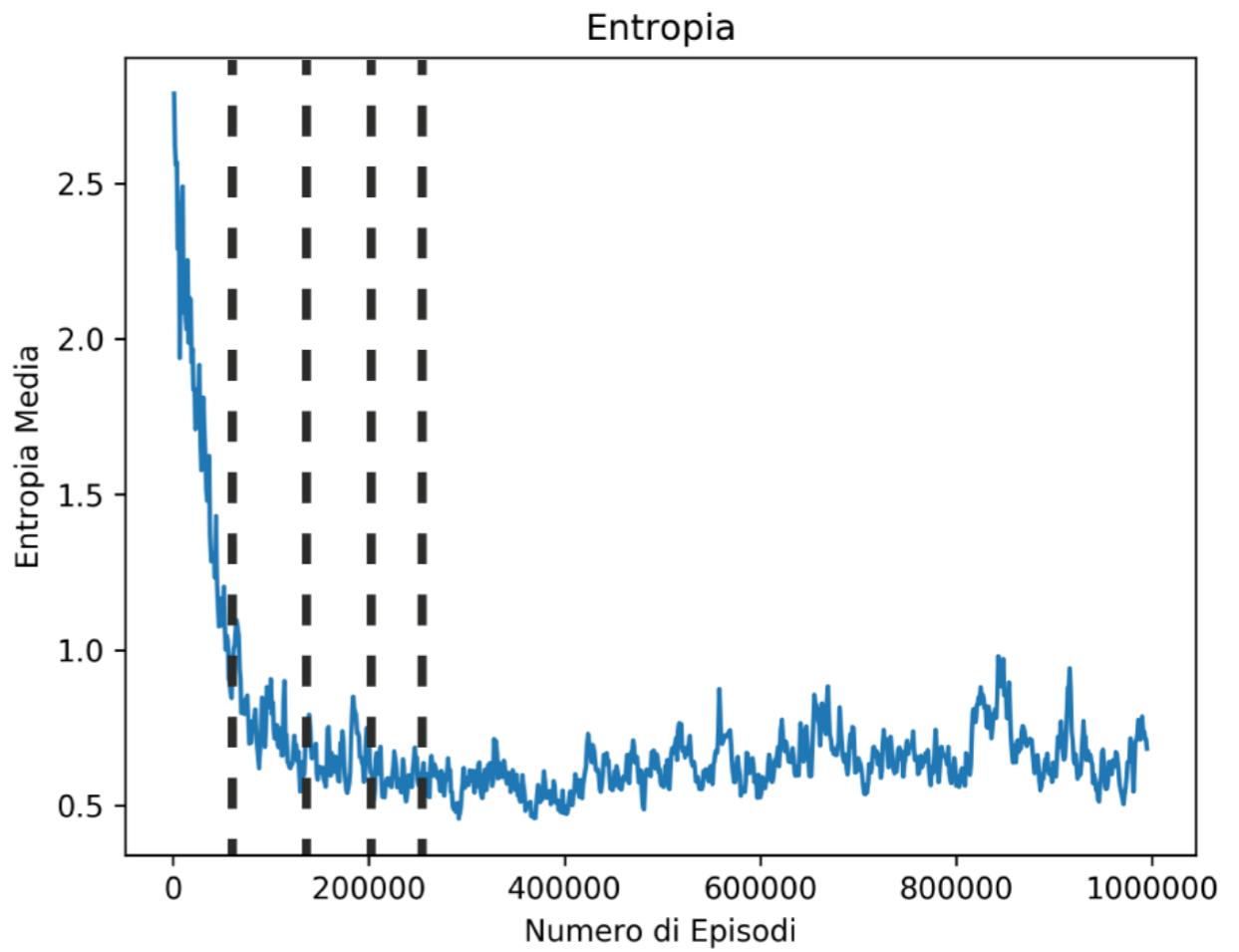
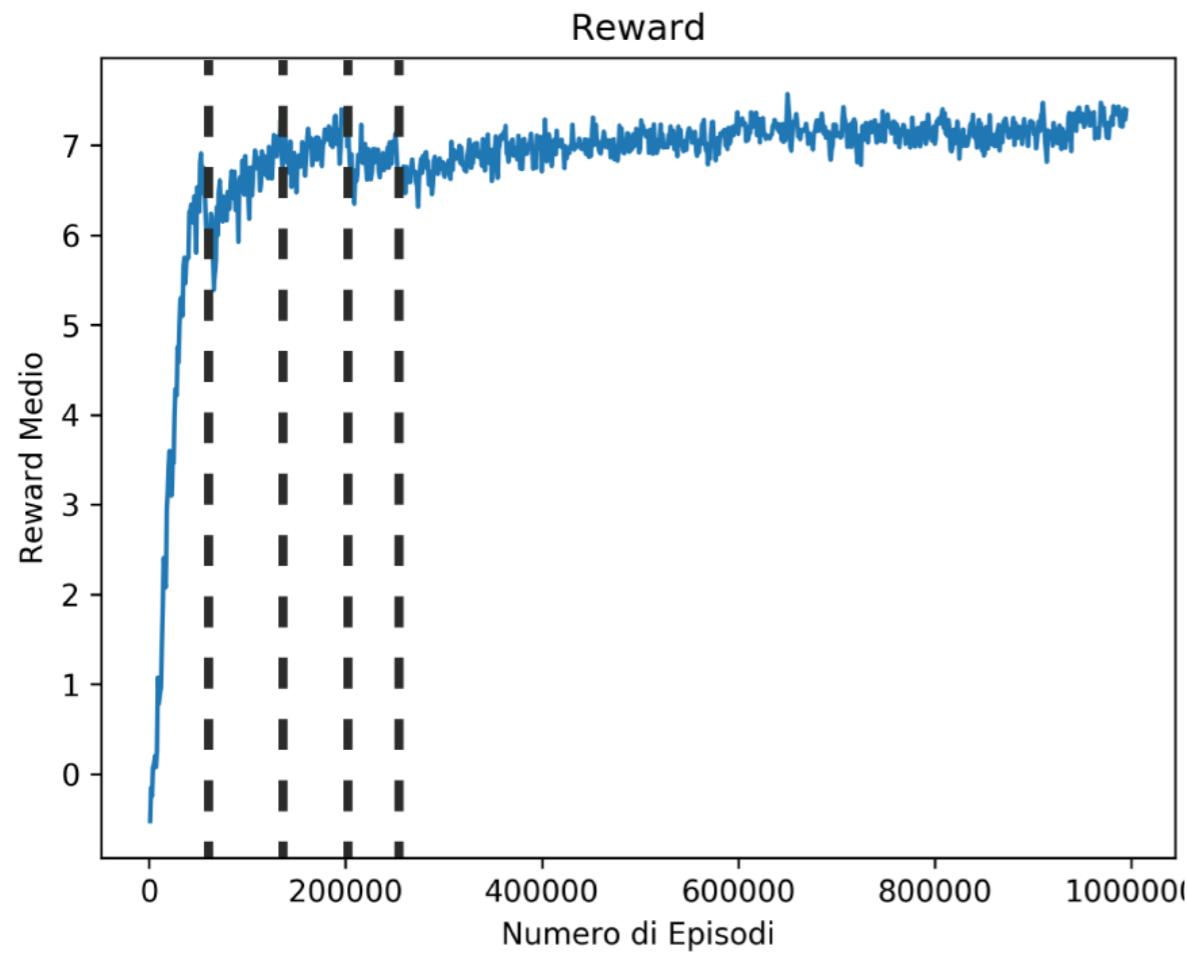
² Schulman, 2017; ³ Kuhnle, 2017

Set-Up di Training

- Per avere degli agenti che riuscissero a gestire tutte le possibili situazioni è stato necessario un certo **grado di casualità** negli elementi dell'ambiente; inoltre, gli agenti hanno giocato contro un nemico che esegue solamente **mosse casuali**.
- Per facilitare l'addestramento è stata usata la tecnica del **Curriculum Learning**, che prevede l'aggiunta di complessità graduale per velocizzare e migliorare l'addestramento.
- I valori di alcune **caratteristiche** degli agenti possono essere modificati prima dell'allenamento, per creare diversi comportamenti.

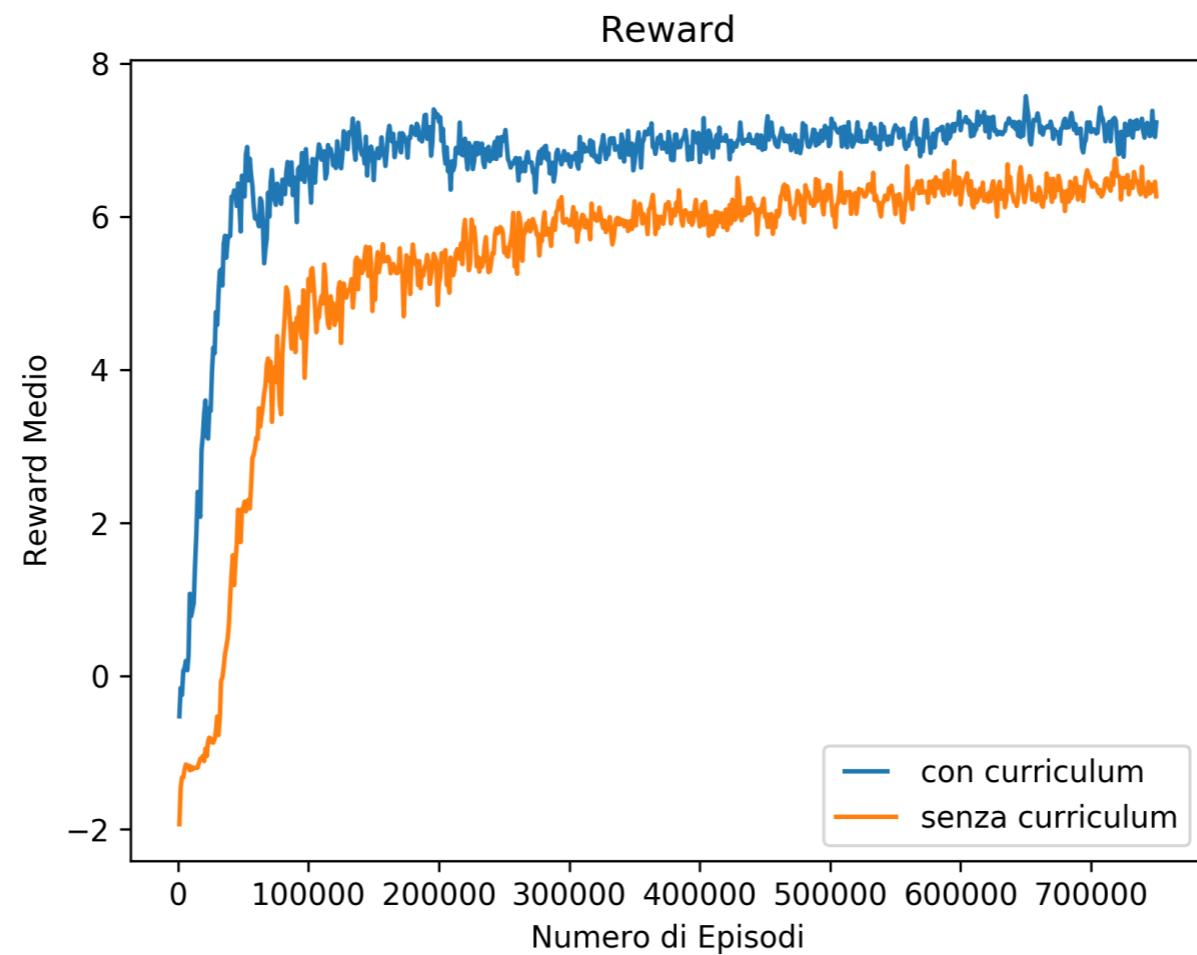
Risultati

- Per valutare la corretta esecuzione dell'addestramento è stato utile eseguire delle analisi **quantitative** osservando l'evoluzione della policy.



Risultati

- Per valutare la corretta esecuzione dell'addestramento è stato utile eseguire delle analisi **quantitative** osservando l'evoluzione della policy.



Risultati

- Per valutare il soddisfacimento dei requisiti è stata necessaria un'analisi **qualitativa**: per questa sono stati condotti dei **test di “giocabilità”**.
- La ricerca è stata effettuata su 10 candidati a cui, alla fine della sessione, è stato richiesto di rispondere ad un **questionario SEQ**.
- Il questionario ha avuto lo scopo di misurare la qualità sia dell'intelligenza artificiale che delle **caratteristiche generali** del prototipo.

Risultati

N°	Domande	Media	σ
1	Ti sentiresti in grado di arrivare fino al livello 10?	5.54	1.03
2	All'aumentare del livello, i nemici sono sembrati troppo forti?	4.63	0.67
3	Ti sembra che i nemici siano intelligenti?	5.72	0.78
4	Ti sembra che i nemici seguano una strategia?	6.18	0.40
5	Ti sembra che i nemici facciano delle mosse controintuitive?	2.00	0.63
6	Le tipologie di nemico hanno lo stesso comportamento?	1.27	0.46
7	Il significato delle icone e delle scritte è facilmente comprensibile?	5.72	1.67
8	Le informazioni offerte dall'interfaccia sono chiare e sufficienti?	5.54	1.21
9	I livelli sono grandi e dispersivi?	2.00	1.34
10	Gli oggetti posti sulla mappa sono tutti ben visibili o hai avuto difficoltà a riconoscerli?	5.81	1.66
11	È utile leggere le caratteristiche di un nemico e/o oggetto (longpress)?	6.90	0.30
12	Quanto è importante avere una strategia?	6.90	0.30
13	Valuta in generale le capacità dei nemici rispetto ad altri giochi RogueLike	6.00	0.77
14	Il gioco è piacevole e divertente?	5.80	0.87
15	L'applicazione presenta molti bug?	1.09	0.30

Conclusioni

- Gli agenti definiti offrono una sfida impegnativa, senza essere proibitivi e presentando un comportamento vario tra le diverse tipologie: tutti i requisiti sono stati soddisfatti.
- Il giusto modello il DRL può essere quindi utilizzato per la creazione di personaggi non giocanti che interagiscono con l'utente in un nuovo prodotto con obiettivi ludici.
- Il modello definito può essere eventualmente utilizzato per altri giochi *RogueLike* che mantengono le caratteristiche generali di *DeepCrawl*.

Lavori Futuri

- Aumentare il grado di generalità del sistema tramite:
 - **Meta Learning,**
 - **Fine-Tuning,**
 - **Controllo Gerarchico.**
- Approfondire e studiare, se esistono, delle tecniche di **testing** più efficaci per valutare la condotta dell'agente già allenato.

Bibliografia

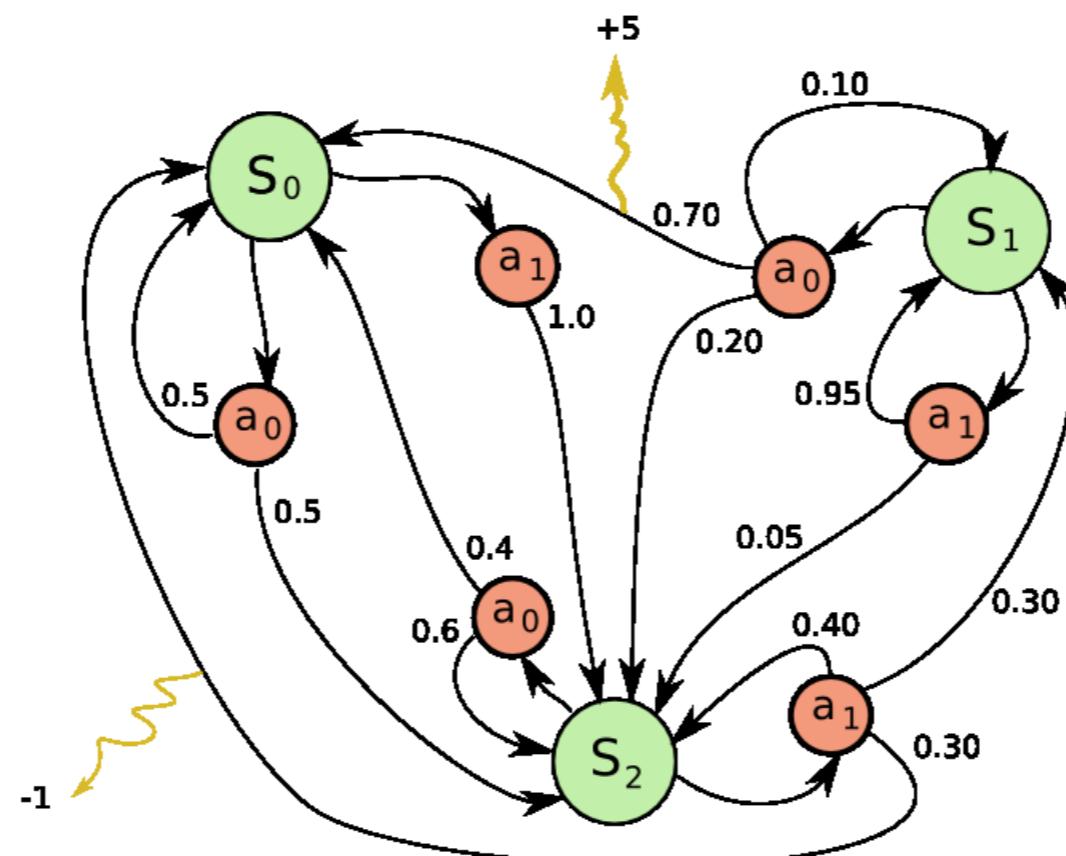
1. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. *Human-level control through deep reinforcement learning*. *Nature*, 518(7540):529–533, 2015.
2. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal policy optimization algorithms*. CoRR, abs/1707.06347, 2017.
3. Alexander Kuhnle, Michael Schaarschmidt, and Kai Fricke. *Tensorforce: a tensorflow library for applied reinforcement learning*. Web page, 2017



Riserva

Reinforcement Learning

- Formalmente il RL può essere modellato come un **Processo Decisionale di Markov** (MDP) con un ambiente in cui si assume che valga la **Proprietà di Markov**.
- Il problema può essere descritto da una tupla $\langle S, A, T, R, \gamma \rangle$.



Metodi Value Function

- **Metodi Value Function:** tecniche basate sulla stima della *state-value function*, che approssima il valore del reward atteso di trovarsi in un determinato stato; la policy è rappresentata da un sistema differente che effettua una scelta a partire dai valori calcolati.
- **Metodi Policy Based:** tecniche basate sulla stima diretta della policy; il risultato è una distribuzione di probabilità sulle azioni.
- **Metodi Actor-Critic:** tecniche ibride che uniscono concetti delle precedenti: viene usata una value function, chiamata baseline, per migliorare la stima della policy.

PPO

- Si definisce:

$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}$$

- La loss function è:

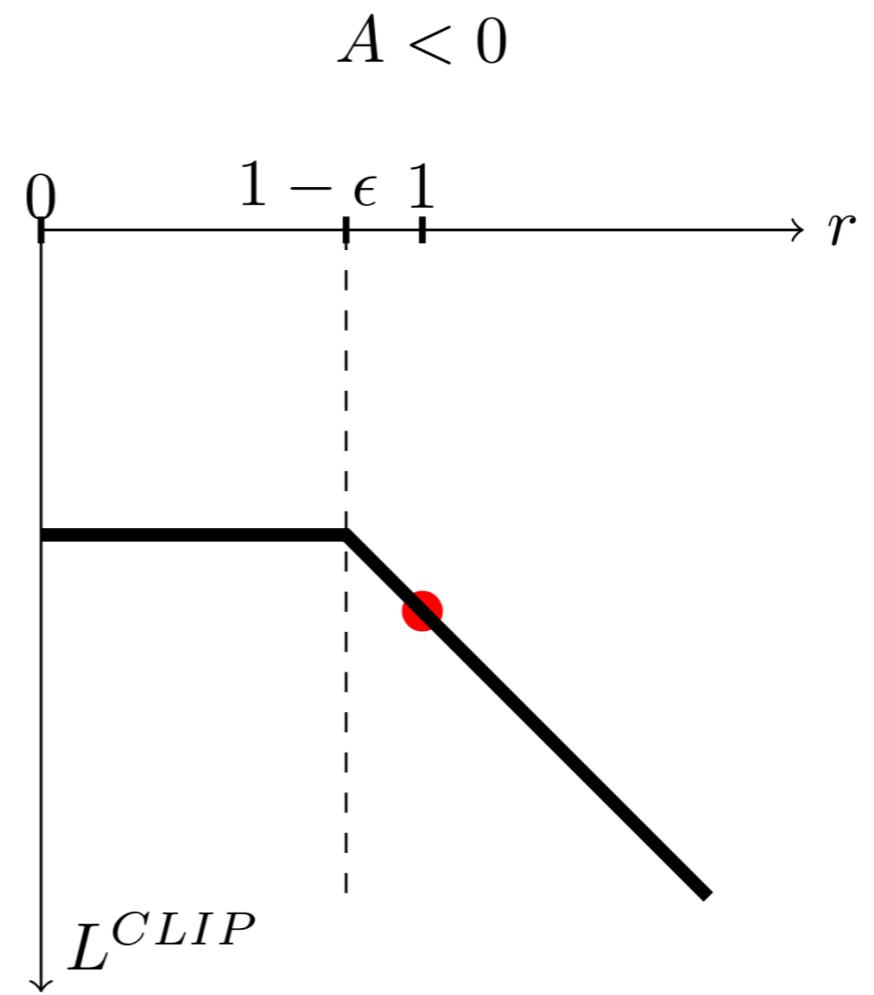
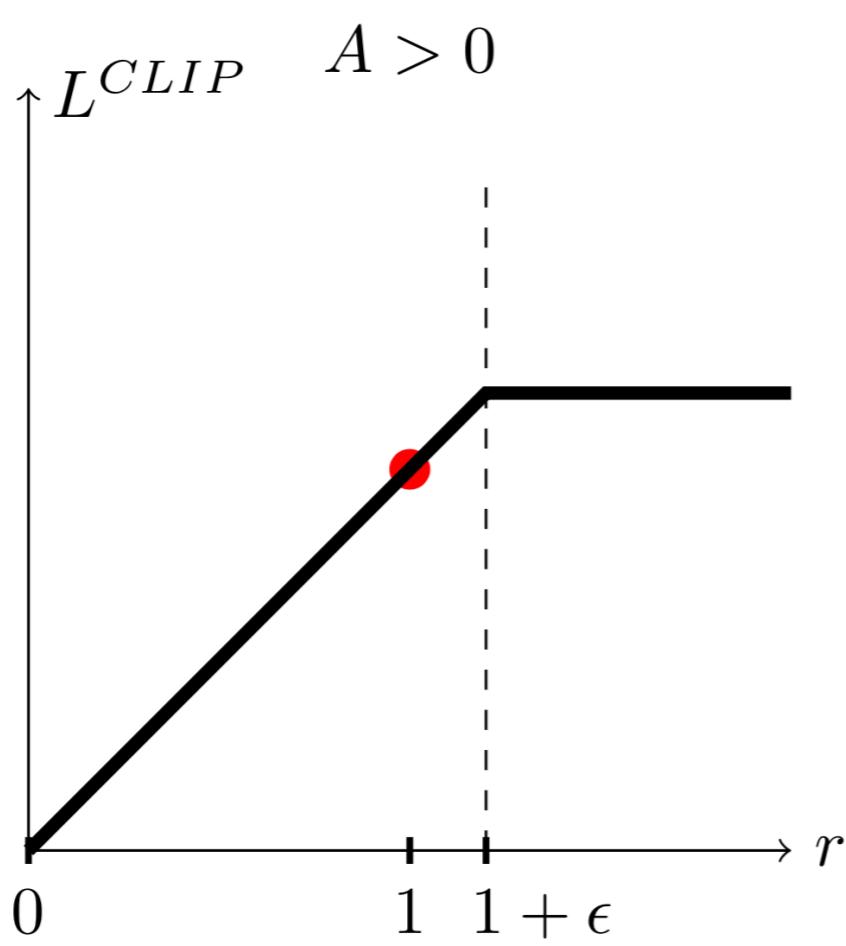
$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)]$$

- Dove:

$$A_t = G_t - b$$

PPO

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



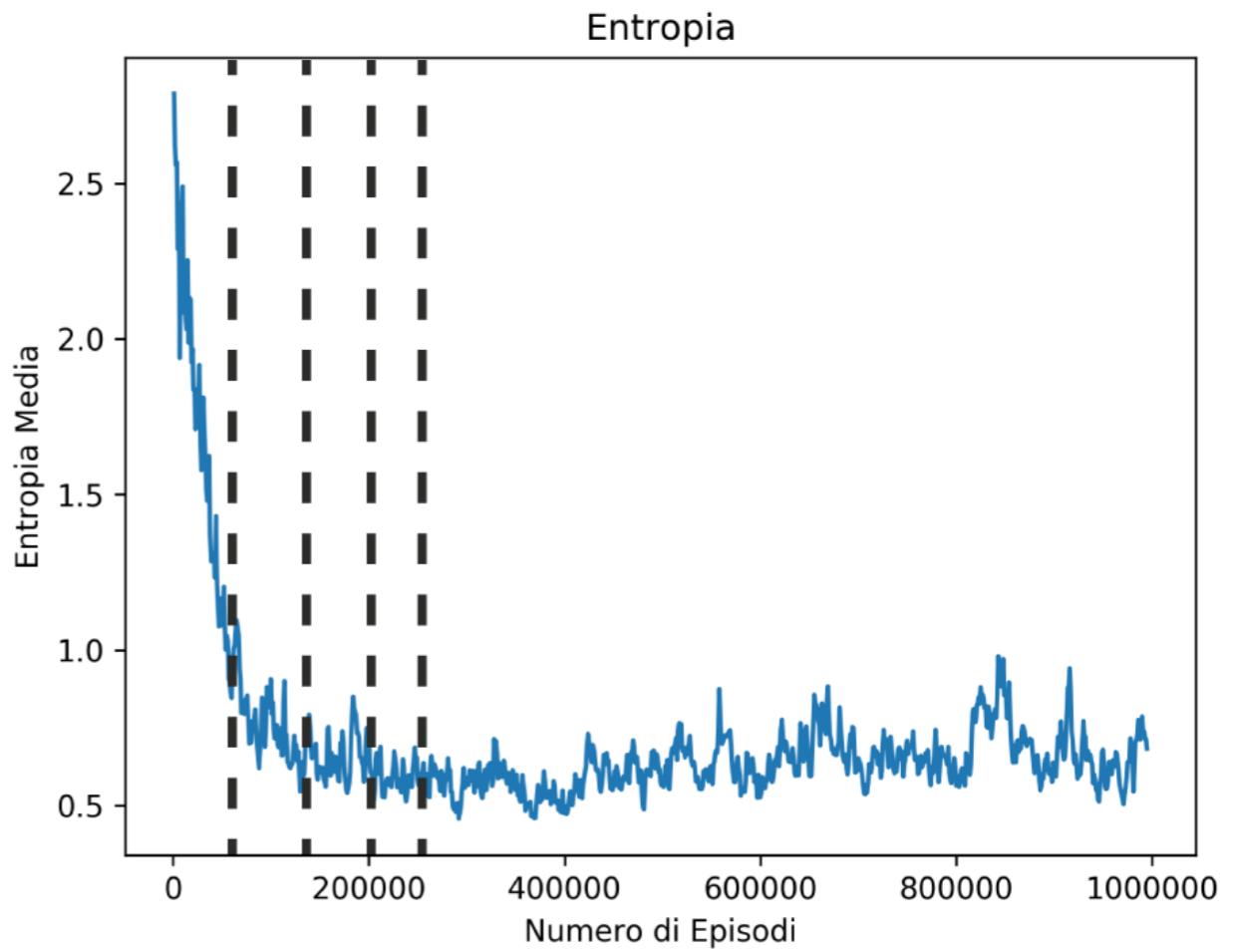
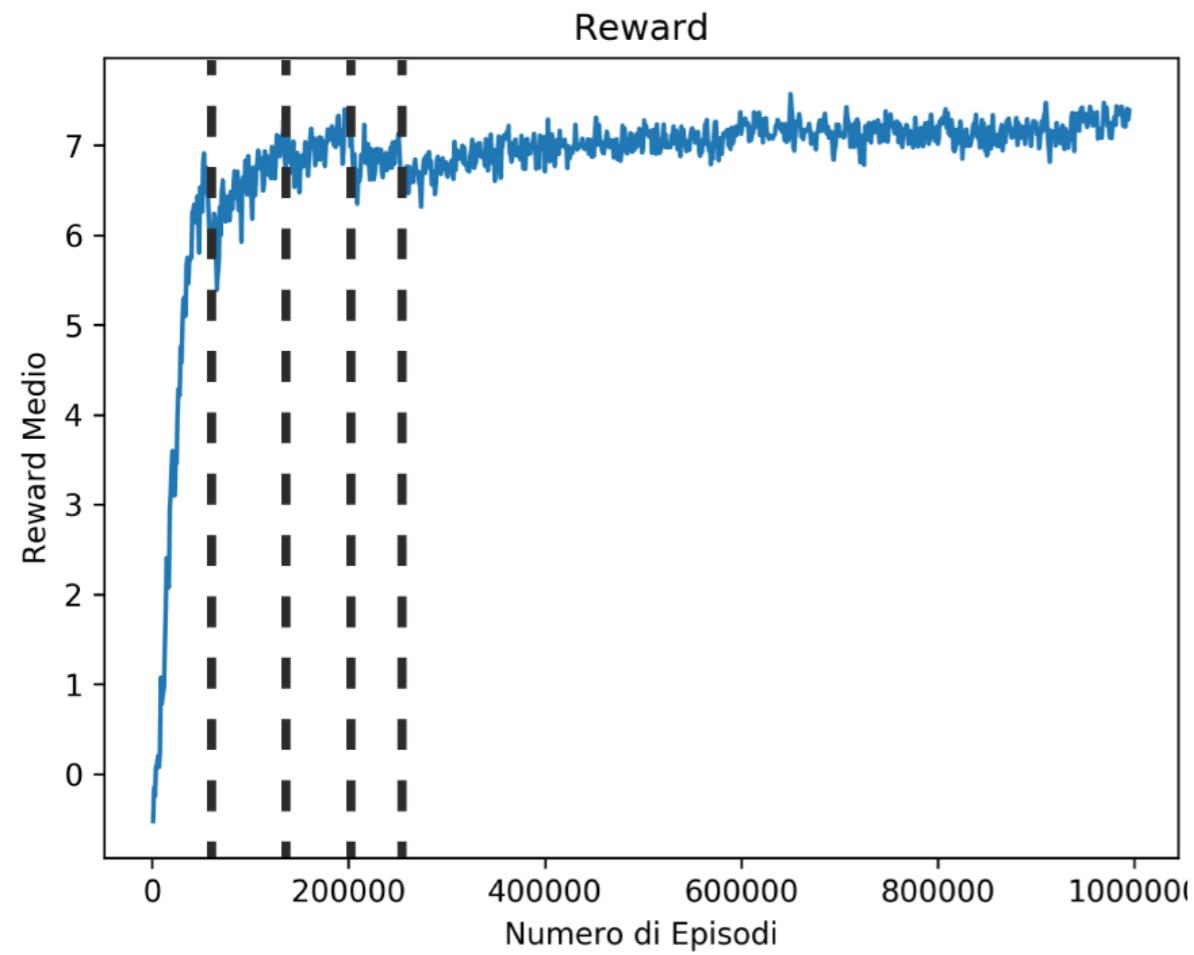
Agente Super-Umano

- Allenando l'agente **contro vecchie versioni** di sé stesso abbiamo creato un nemico molto intelligente con capacità super-umane.



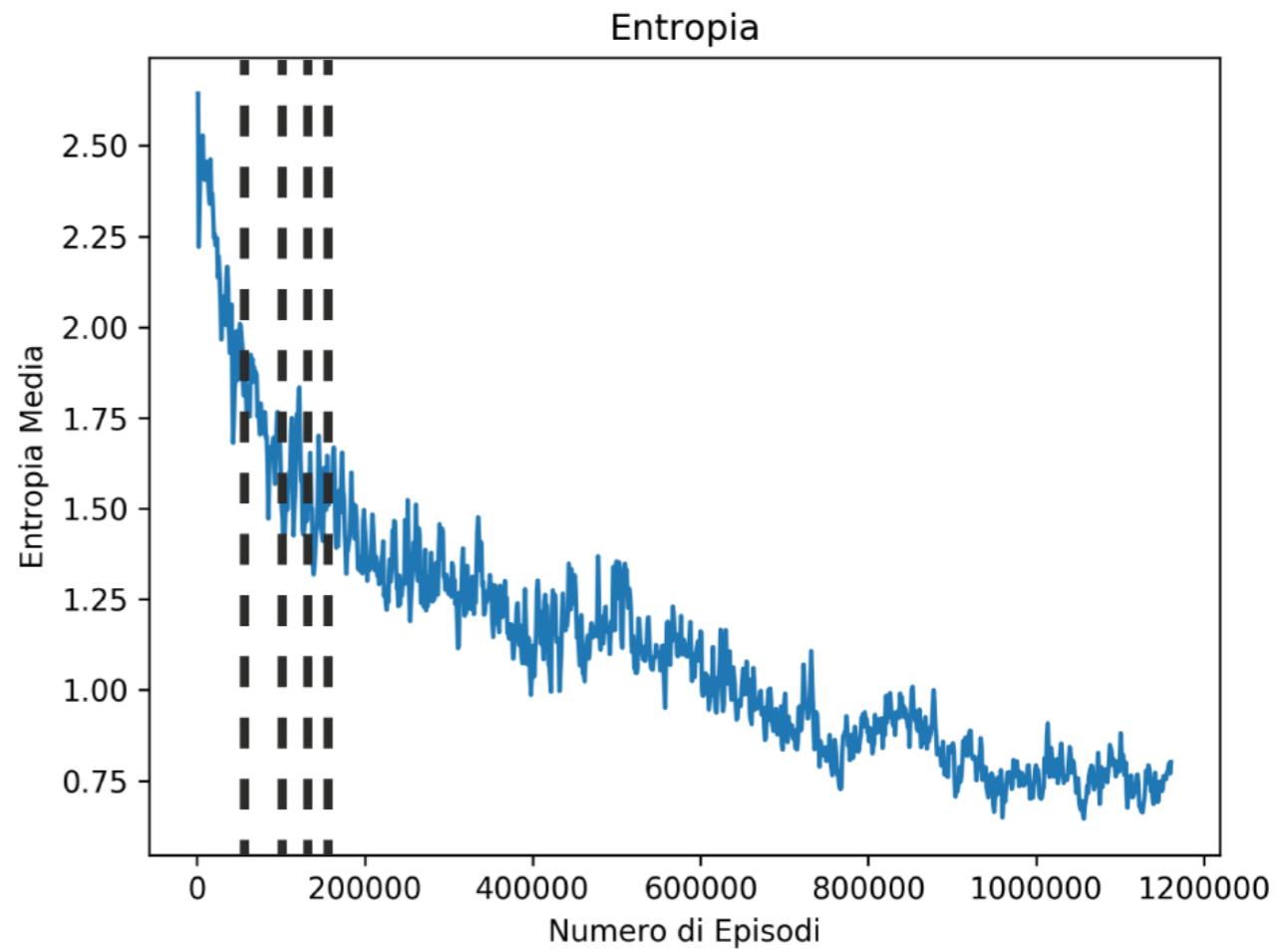
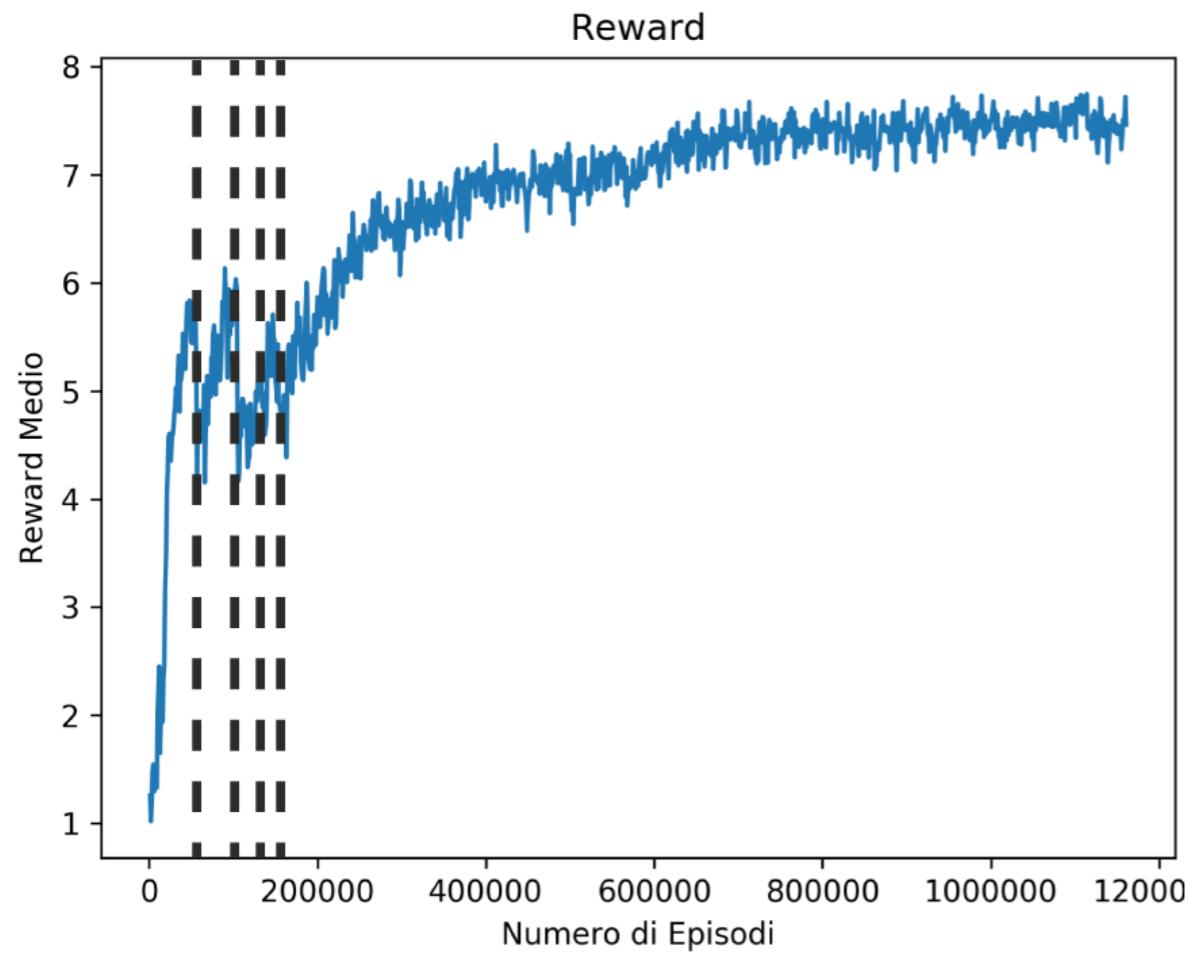
Risultati

- Classe guerriero:



Risultati

- Classe arciere:



Risultati

- Classe ranger:

