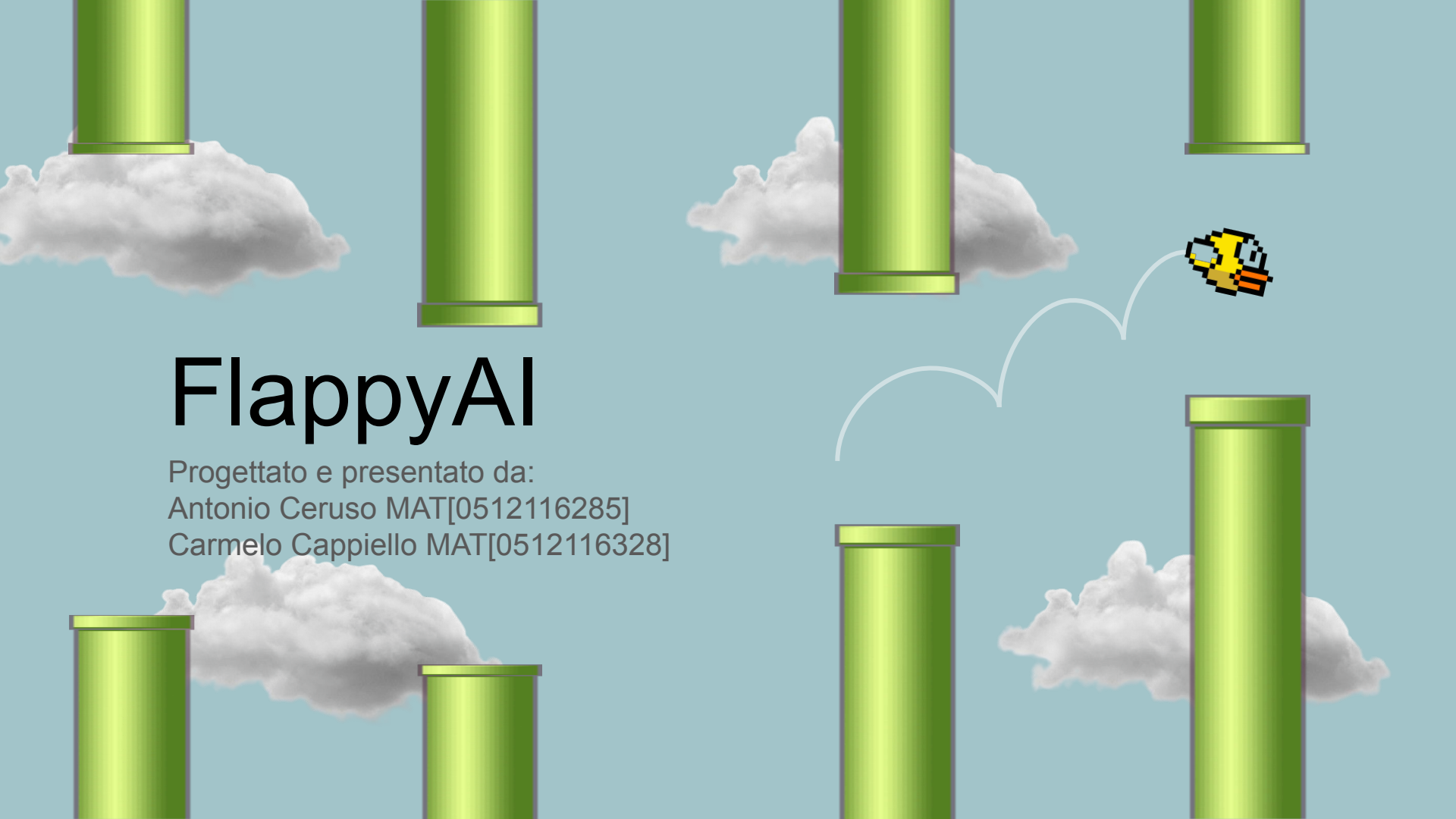
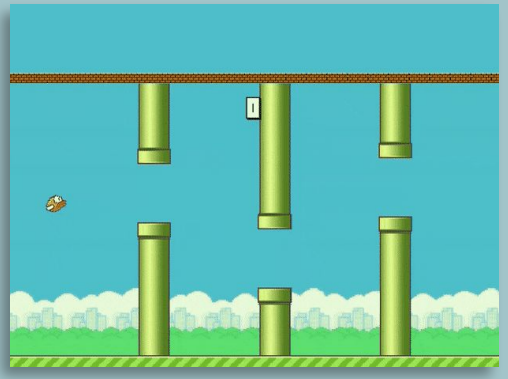
FlappyAI

Progettato e presentato da:
Antonio Ceruso MAT[0512116285]
Carmelo Cappiello MAT[0512116328]



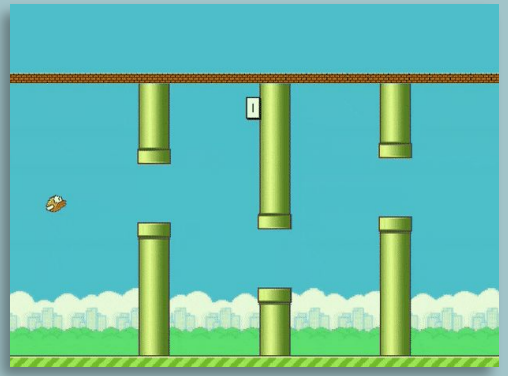


Cos’è FlappyAI?

FlappyAI nasce dalla curiosità di osservare come due approcci di training permettano a un'intelligenza artificiale di giocare in modo perfetto a un videogioco.

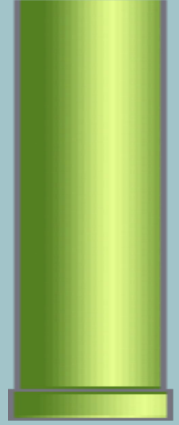
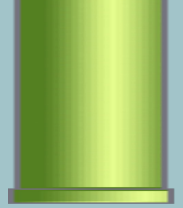
FlappyAI implementa il famosissimo gioco del Flappy Bird mediante due approcci:

- Il primo approccio sfrutta un classico algoritmo di ricerca locale.
- Il secondo approccio invece sfrutta il deep learning e quindi le reti neurali per il training del modello.



Che gioco è Flappy Bird e in cosa consiste?

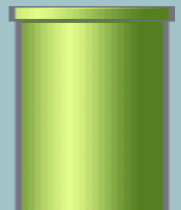
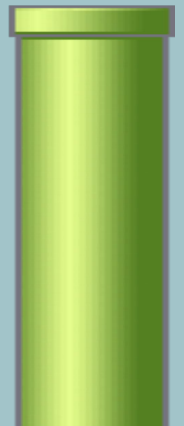
Flappy bird è uno dei giochi più storici, famosi e iconici dello scorso decennio.
Consiste in un gioco a scorrimento orizzontale, dove lo scopo è fare più punti.
L'uccellino dovrà attraversare dei tubi **senza scontrarsi** con essi, ogni tubo che attraversa corrisponde ad un punto ottenuto, l'obiettivo è **accumulare quanti più punti possibili**.
Se si scontra con un tubo o con i bordi del gioco, la partita termina.

Implementazione del gioco

Per sviluppare entrambi i modelli è stata necessaria un'implementazione del gioco vero e proprio, che però sfruttasse l’input umano per l’avanzamento.
Tale implementazione del gioco è stata reperita nella seguente repository GitHub, sviluppata da MaxRohovsky:

- https://github.com/MaxRohowsky/flappy-bird

Modifiche preliminari all’implementazione del gioco

Abbiamo apportato una serie di modifiche al gioco per adattarlo alle nostre esigenze e per rendere più semplice e veloce il training per i modelli:

- Rimosso game over screen
- Implementata versione senza grafica
- Varie modifiche alla fisica/bilanciamento del gioco


FlappyAI:
Implementazione con Hill Climbing

Hill climbing

Overview

L’Hill Climbing è un algoritmo di ricerca locale che punta all’individuazione di una soluzione ottima ad un problema, partendo da una soluzione iniziale e muovendosi verso soluzioni vicine che migliorano il risultato.

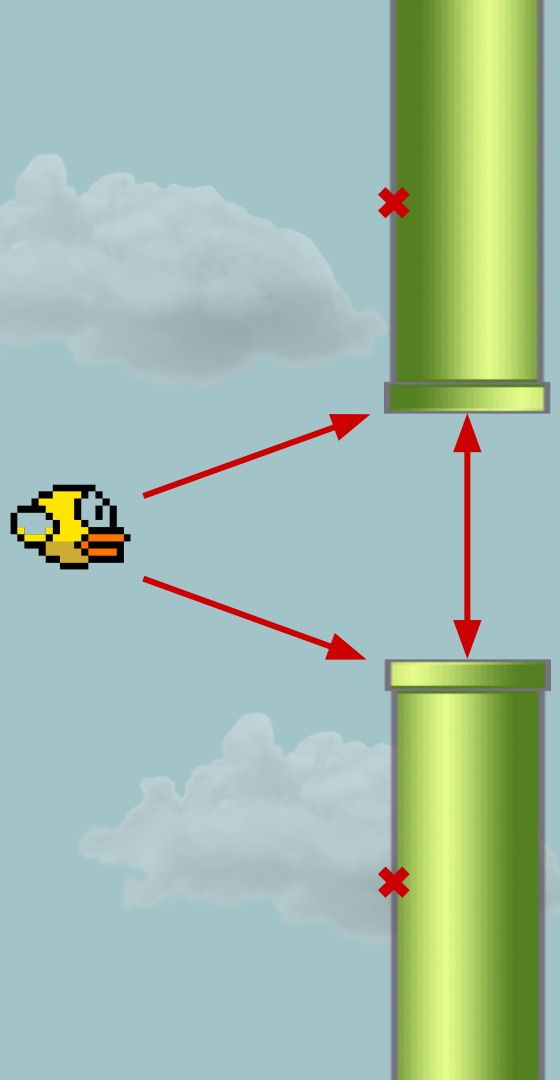
A ogni passo, l'algoritmo confronta gli stati adiacenti e sceglie quello che offre il miglioramento maggiore, procedendo iterativamente fino a raggiungere un **massimo/minimo locale**, da cui non è più possibile migliorare ulteriormente.

Hill climbing

Overview

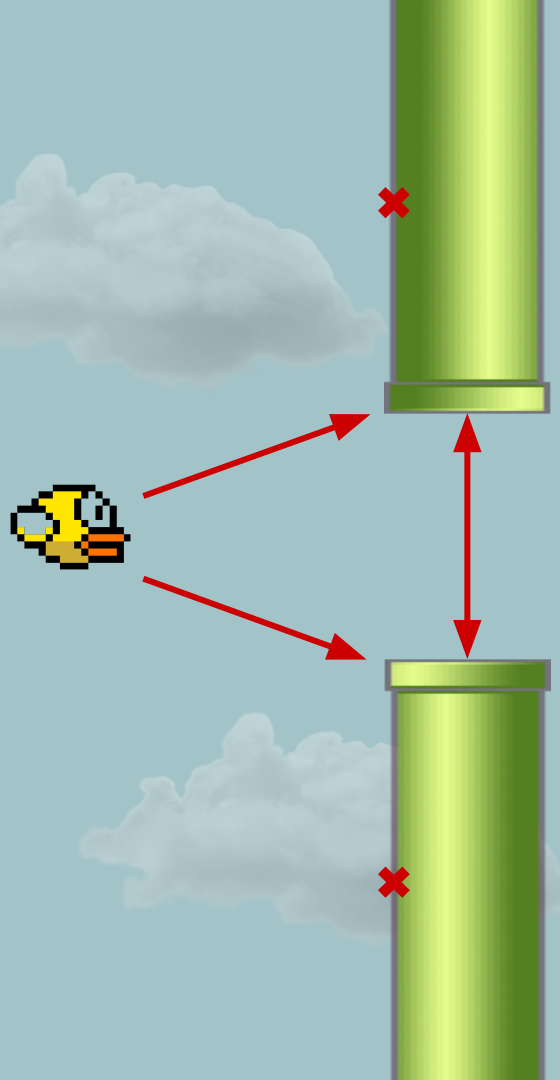
Tale approccio si può dividere in 4 step principali:

1. Si parte da una soluzione casuale.
2. Vengono valutate le soluzioni vicine.
3. Viene presa la soluzione migliore tra quelle vicine, **se migliora** rispetto alla situazione attuale.
4. Il processo si ripete finché non viene trovato un punto dove nessuna soluzione vicina è migliore.



Strategia

- Per ogni singolo fotogramma il modello valuta due azioni possibili:
  - FLAP
  - NON FLAP
- Simula il futuro per un certo orizzonte di passi
- Valuta quale azione minimizza la distanza verticale dal centro del gap tra i tubi.
- Sceglie l'azione che porta il bird più vicino al centro del varco.



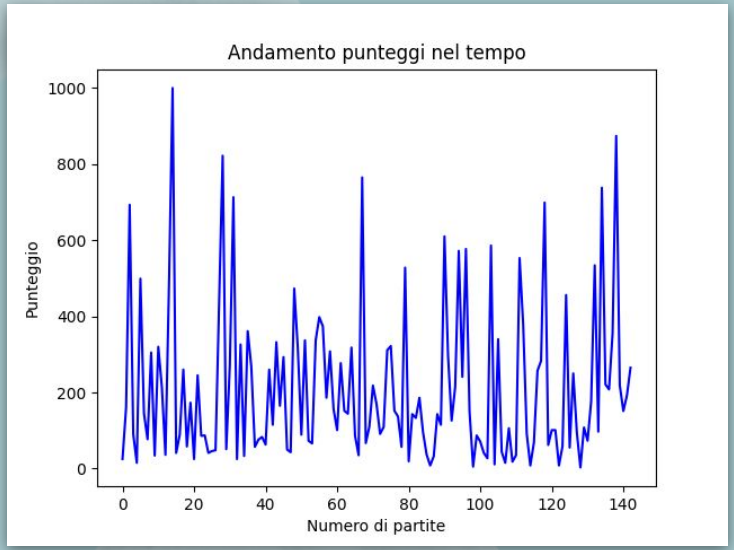
Strategia

- Per simulare senza alterare il gioco corrente, viene creata una **copia esatta** dello stato del gioco.
  In sostanza un clone vero e proprio.

Per la valutazione delle scelte fatte vengono presi in considerazione due casi principali:

- Se l'uccellino è morto, allora punteggio molto negativo.
- Più piccola è la distanza dell'uccellino dal centro del gap, più sta seguendo la traiettoria corretta e quindi maggiore sarà il punteggio per la valutazione.

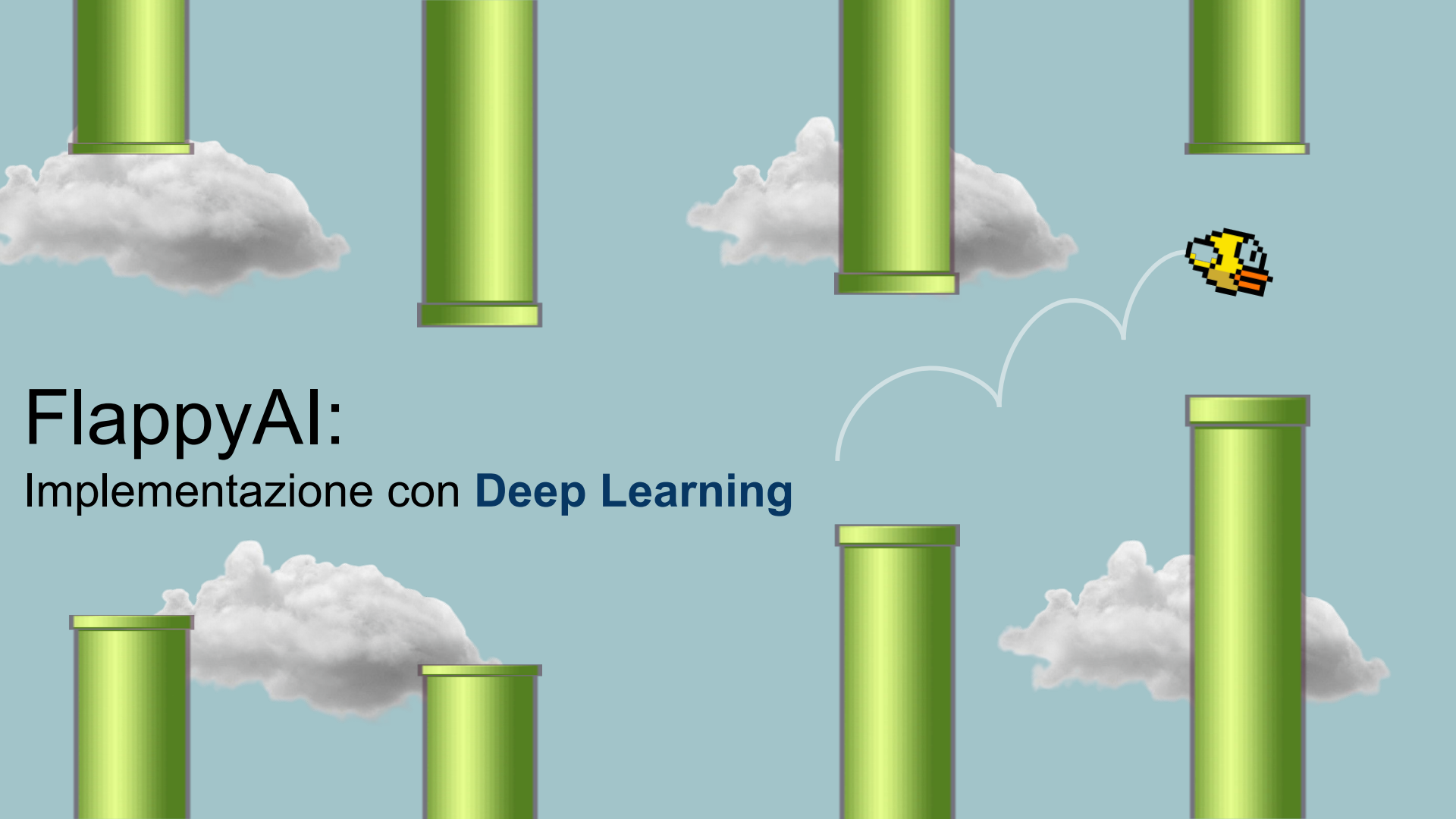
Grafico dell’andamento degli score di Hill Climbing



Considerazioni

Da un’analisi approfondita delle performance è emerso che:

- **Alta variabilità**: il punteggio oscilla moltissimo da partita a partita.
- **Nessuna crescita costante**: l'Hill Climbing, in questa versione, non sta imparando in modo progressivo.
- **Presenza di outlier**: ogni tanto ci sono partite eccezionalmente buone.

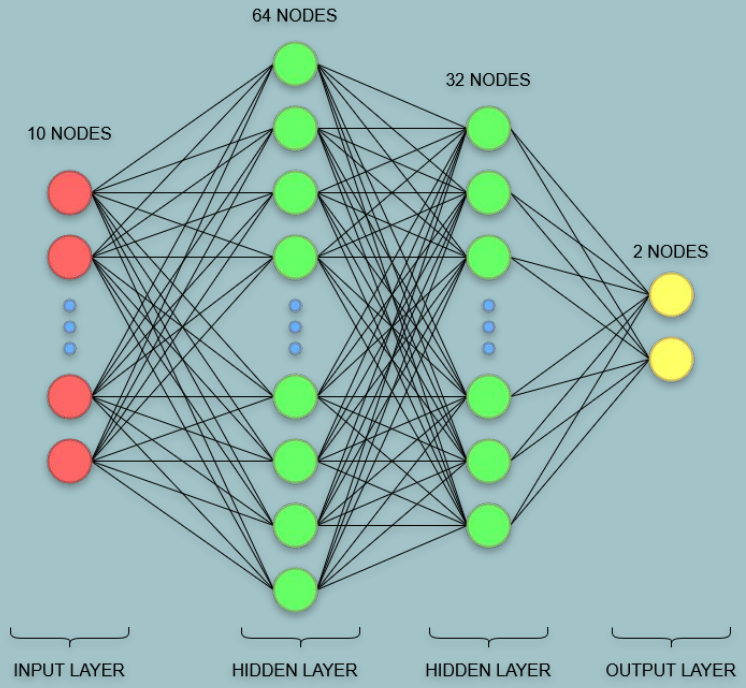

FlappyAI:
Implementazione con **Deep Learning**

Deep Learning

Overview

Il deep learning è un branch del machine learning che si basa sull'utilizzo delle **reti neurali** per cogliere relazioni profonde tra le feature di un dataset, che modelli tradizionali non riescono ad individuare.

Una rete neurale è formata da layer di neuroni artificiali dove ogni layer passa informazioni al successivo.
Alla fine la rete restituirà un output (es. "salta", "non salta").

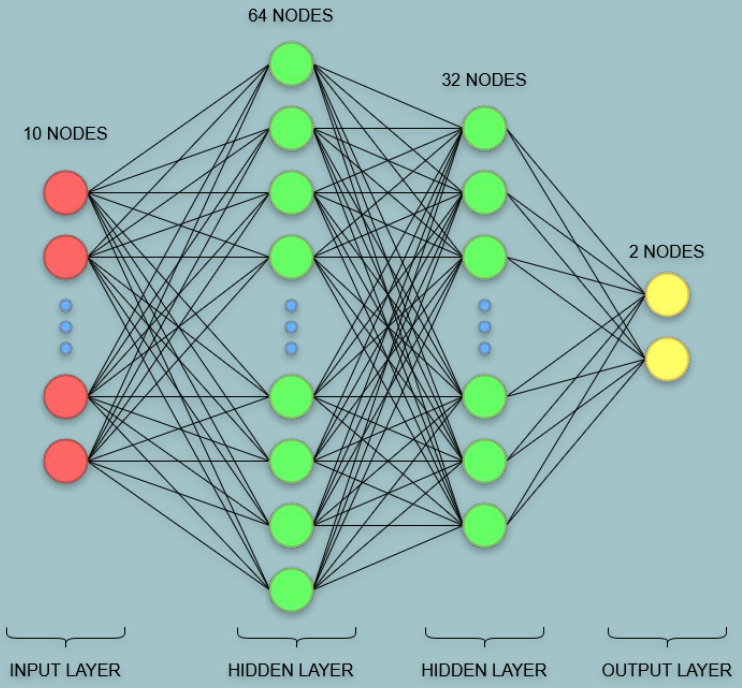


Deep Learning

Implementazione

Nel nostro caso abbiamo utilizzato una rete neurale con 4 layer:

- **Layer di input** con 10 nodi
- **Hidden layer** con 64 nodi (ReLu)
- **Hidden layer** con 32 nodi (ReLu)
- **Layer di output** con 2 nodi (salta/non salta)



Deep Learning

Dati in input

Nelle reti neurali, i dati in input vengono organizzati in **tensori**, spesso sotto forma di array multidimensionali. Questo approccio è vantaggioso perché consente di eseguire efficientemente operazioni vettoriali e matriciali, che sono **altamente parallelizzabili**. Grazie a questa struttura, le GPU possono elaborare grandi quantità di dati contemporaneamente, **accelerando notevolmente il processo di training** del modello.

Deep Learning

Dati in input

Nel nostro caso i dati in input dati al modello derivano proprio dallo **stato attuale della partita**, come:

- Posizione attuale
- Altezza ottimale
- Distanza orizzontale dal prossimo set di tubi
- Altezza tubo inferiore
- Altezza tubo superiore
- Altri valori utili all'apprendimento

Per garantire che il modello apprenda in modo efficace e stabile, abbiamo normalizzato questi valori, portandoli su una scala compresa tra 0 e 1.

Reward e penalty

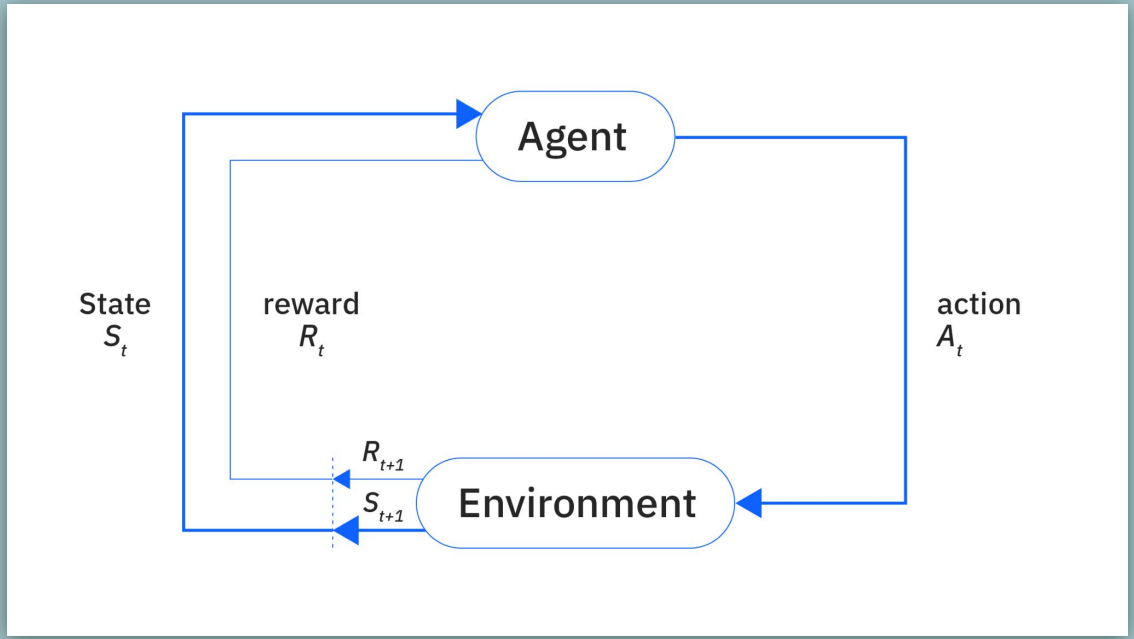
Breve overview sul reinforcement learning

Per realizzare il nostro progetto ci siamo rivolti ad un branch del machine learning chiamato il **reinforcement learning**.

Si basa su un concetto molto semplice:
per ogni azione intrapresa dal modello, il gioco **restituirà un numero** in base allo stato della partita attuale, **positivo** se ottimale, **negativo** in caso contrario.

Reward e penalty

Breve overview sul reinforcement learning



Reward e penalty

Implementazione reward/penalty

In base ai reward ottenuti, il modello sarà **incentivato o meno** a compiere di nuovo determinate azioni.

Abbiamo modellato i reward in modo da incoraggiare il modello a spostarsi in **situazioni favorevoli.**

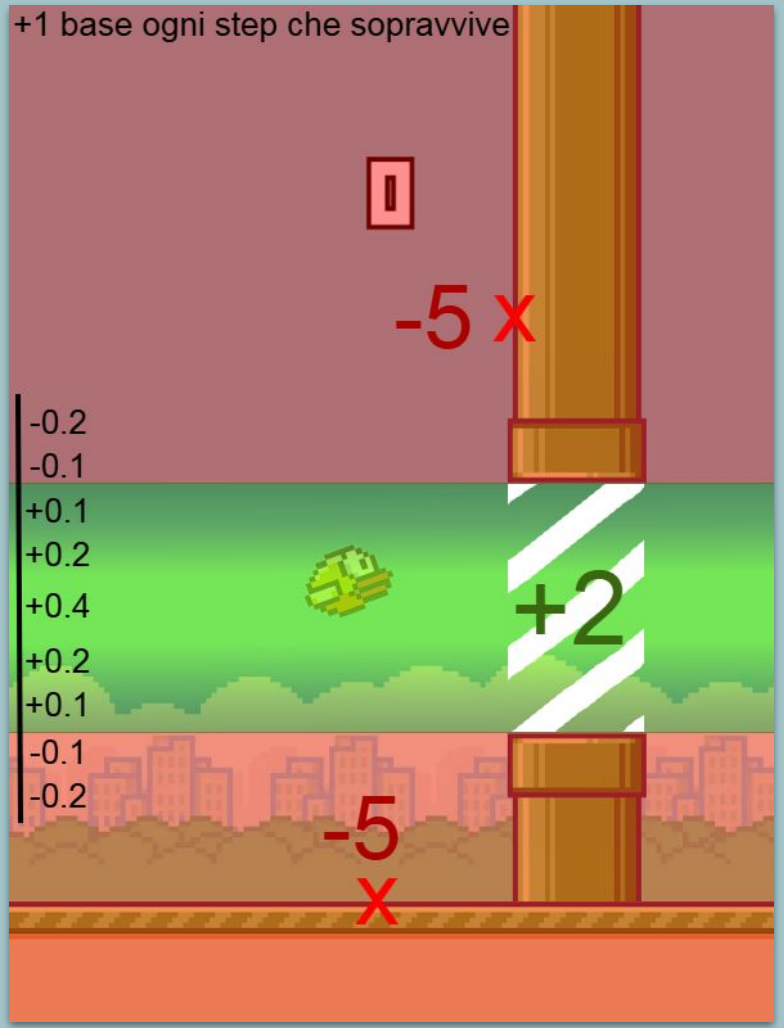
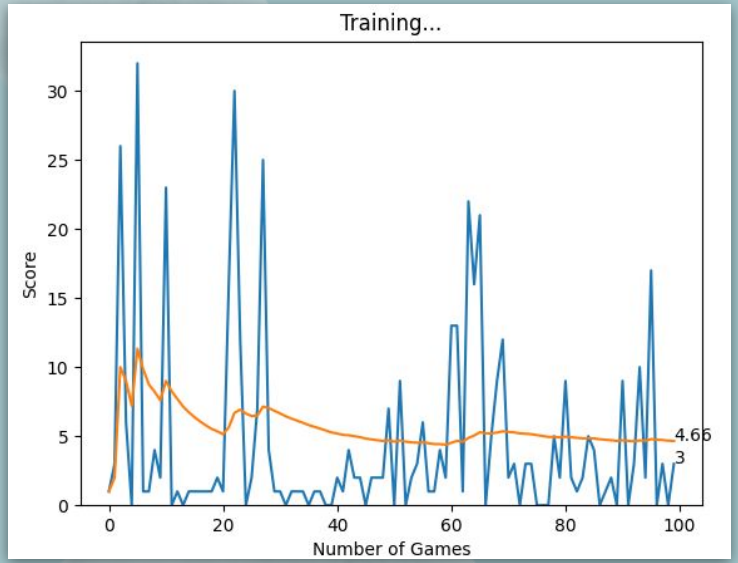


Grafico dell’andamento degli score del modello Deep Learning



Considerazioni

Dall’analisi approfondita delle performance emergono le seguenti considerazioni:

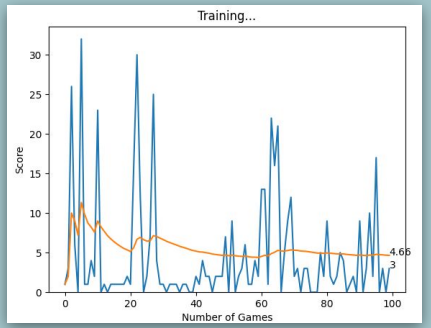
- **Estrema variabilità**: anche in questo modello le prestazioni sono molto altalenanti.
- **Performance non ottimali**: i risultati complessivi non sono particolarmente brillanti, probabilmente a causa della conoscenza limitata dell’ambito specifico.
- **Tempi di training elevati**: per raggiungere questi livelli di performance, il modello ha richiesto un tempo di addestramento considerevole.

Considerazioni finali

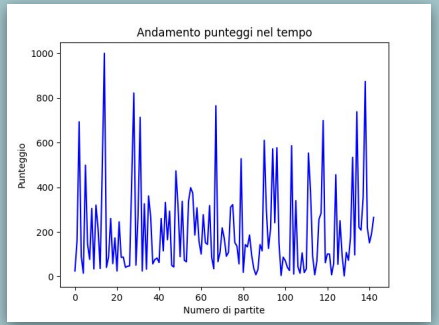
Confronto tra i due modelli

Dai risultati ottenuti possiamo notare che il modello con Hill Climbing è stato un successo rispetto a quello implementato con Deep Learning. Questo principalmente deriva dal fatto che quest'ultimo era un "primo approccio" al mondo del deep learning con le nostre limitate conoscenze.

DEEP LEARNING



VS



HILL CLIMBING

Considerazioni finali

Confronto tra i due modelli

Un altro aspetto positivo del modello Hill Climbing è che, a differenza del modello basato su Deep Learning, **non ha bisogno di una lunga fase di addestramento**. Nel nostro caso, per il modello Deep Learning sono state necessarie oltre **10.000 partite** giocate, mentre Hill Climbing si basa su una **strategia euristica** e può iniziare a giocare in modo ottimale fin da subito, senza bisogno di training.

Conclusioni

È stato molto interessante osservare il confronto tra i due modelli, sebbene ci aspettassimo prestazioni superiori da parte del modello basato su deep learning. Le limitate conoscenze della materia e l'utilizzo di un'architettura seppur funzionante ma approssimativa e semplificata non ha favorito il raggiungimento dei risultati sperati.

Nonostante ciò, siamo riusciti a realizzare una versione soddisfacente del gioco utilizzando l'algoritmo Hill Climbing. Nel complesso le prestazioni risultano buone, lasciando ampio margine di un possibile miglioramento futuro.

Grazie per l’attenzione!

Progettato e presentato da:
Antonio Ceruso MAT[0512116285]
Carmelo Cappiello MAT[0512116328]