

UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI INGEGNERIA ELETTRICA, ELETTRONICA
E INFORMATICA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

ANTONIO IMPALÀ

**Piattaforma web per il monitoraggio di algoritmi di
reinforcement learning**

TESI DI LAUREA

Relatore:
Chiar.mo Prof. S. PALAZZO

ANNO ACCADEMICO 2018/2019

Abstract

Recently, the *reinforcement learning (RL)* has been applied fruitfully in the field of Artificial Intelligence, particularly in robot kinematics [18] and videogames [23], thanks to the fact that it is more efficient when it is necessary to have model that is more adaptable to the working environment. Yet, model training may be complex because it is difficult to establish correct training parameters and to define the reward function which a model seeks to maximise. Moreover, it is important to take into account the model action efficacy in reaching its prefixed goal. Therefore, it is crucial to monitor the model training process.

This work aims at providing a way to monitor RL training, through a web framework that starts and manages the training process. As the results achieved show, the devised system provides a useful overview about the main metrics to control a training process, as well as the opportunity to view the algorithm behaviour in the simulated environment.

Sommario

Negli ultimi anni, il paradigma del *reinforcement learning* (*RL*) ha trovato numerose applicazioni nel campo dell'intelligenza artificiale, ad esempio, nella cinematica dei robot [18] o nei videogame [23], grazie al fatto che si rivela il più efficace in tutti i casi in cui è necessario che il modello abbia maggiore adattabilità all'ambiente circostante. Tuttavia, l'allenamento di questi modelli risulta complesso e problematico: si riscontrano difficoltà nello stabilire sia i parametri di allenamento del modello, sia il segnale di “ricompensa” che il modello stesso cerca di massimizzare. Inoltre, bisogna tenere in considerazione anche l'efficacia dell'azione del modello per il raggiungimento dell'obiettivo prefissato, pertanto è fondamentale per il progettista monitorare il processo di allenamento del modello.

Il presente lavoro si propone quindi di fornire uno strumento per il monitoraggio dell'allenamento di algoritmi di *RL*, tramite una piattaforma web che avvia e si interfaccia con il processo di allenamento del modello. I risultati ottenuti mostrano come il sistema fornisca una vista immediata delle metriche principali per il controllo del processo di allenamento, oltre alla possibilità di visionare direttamente il comportamento dell'algoritmo nell'ambiente simulato.

Indice

1	Introduzione	1
2	Stato dell'arte	7
2.1	Machine Learning	7
2.2	Reti neurali	9
2.3	Reinforcement Learning	18
3	Deep Q-Learning	24
3.1	Allenamento	28
4	Piattaforma web per il monitoraggio di algoritmi di reinforcement learning	31
4.1	Autenticazione, registrazione e recupero delle credenziali . . .	32
4.2	Home	35
4.3	Archivio	40
5	Risultati e conclusioni	42

5.1 Conclusioni	47
Bibliografia	48
Bibliografia e link	48
Elenco delle figure	54

Capitolo 1

Introduzione

Da qualche decennio a questa parte, l’Intelligenza Artificiale (o IA) e i temi ad essa connessi destano sempre più interesse tra i ricercatori per le loro possibili applicazioni, dato che i potenziali campi in cui possono essere usate sono innumerevoli. In generale possiamo vedere l’IA come un insieme di algoritmi che permettono ad un sistema di fornire le prestazioni e i risultati richiesti in modo “intelligente”, cioè proprio come farebbe l’uomo. L’area di ricerca di intelligenza artificiale che finora ha avuto il maggior successo (per quanto non sia ancora in grado di “coprire” tutti gli aspetti dell’intelligenza umana) è il machine learning. Questa definisce tutti i vari algoritmi necessari alla macchina per poter imparare dall’esperienza autonomamente, senza essere stati esplicitamente programmati. Per quanto riguarda invece le possibili applicazioni dell’IA, qualche esempio tra quelle in forte crescita negli ultimi anni sono Computer Vision [2] e Natural Language Processing [3]; inoltre sta acquisendo sempre più importanza l’utilizzo etico, equo e trasparente dell’IA,

necessario a conquistare la fiducia dei consumatori, in particolare per quanto riguarda i dati raccolti e la privacy. Nel settore della computer vision, il volume di mercato nel 2018 ammontava a 3,62 miliardi di dollari, ma in base alle proiezioni attuali, i ricercatori di MarketsandMarkets prevedono che tale cifra salirà a 23 miliardi di dollari entro il 2023 [1]. Vediamo in linea generale di cosa si occupano questi due settori:

- La *Computer Vision* è l'insieme dei processi automatici mirati ad estrarre interpretazioni a partire da dati visuali (immagini e video). Lo scopo principale è quello di riprodurre la vista umana, non solo intesa come acquisizione di un'area, ma anche come interpretazione del contenuto della stessa. Questo sistema permette di analizzare ed elaborare immagini sia nello spettro della luce visibile, sia al di fuori di essa (infrarossi, raggi X, ecc...). Il risultato dell'elaborazione è il riconoscimento di caratteristiche dell'immagine con finalità, ad esempio, nell'ambito della sicurezza. Un problema tipico nel campo della computer vision è quello dell'*Object Recognition*, cioè determinare se sono presenti o meno specifici oggetti all'interno di immagini o video. Di questo problema esistono tre varianti:

- *Recognition*: uno o più oggetti prespecificati o memorizzati possono essere ricondotti a classi generiche insieme alla loro posizione

nella scena;

- *Identification*: viene individuata un’istanza specifica di una classe. Ad esempio l’identificazione di un volto, impronta digitale o veicolo specifico;
 - *Detection*: l’immagine è scandita fino all’individuazione di una condizione specifica. Ad esempio l’individuazione di possibili cellule anormali o tessuti nelle immagini mediche [2].
- *Natural Language Processing* è il processo di trattamento automatico delle informazioni, scritte o parlate, in una lingua naturale. Questo processo è particolarmente complesso a causa dell’ambiguità che può spesso avere il linguaggio umano, pertanto viene suddiviso in fasi:
 - *Analisi lessicale*: scomposizione di un’espressione linguistica in parole;
 - *Analisi grammaticale*: associazione delle parti del discorso a ciascuna parola nel testo;
 - *Analisi sintattica*: arrangiamento delle parole in una struttura sintattica (ad albero: *parse tree*);
 - *Analisi semantica*: assegnazione di un significato (semantica) alla struttura sintattica e, di conseguenza, all’espressione linguistica.

L’assegnazione automatica all’espressione linguistica di un particolare significato, tra i vari possibili, è detta *disambiguazione* [3].

Gli approcci appena descritti, come anche buona parte delle altre applicazioni in crescita in questi anni, si basano sull’apprendimento supervisionato, in cui durante l’allenamento vengono passati all’algoritmo dati già etichettati, cioè esempi ideali costituiti da coppie di input e di output, con l’obiettivo di istruire un modello in grado di elaborare automaticamente previsioni sui valori di uscita di un sistema rispetto ad un input. In questo caso, i dati iniziali in possesso e forniti durante l’allenamento costituiscono l’esperienza del modello. Inoltre, questa tipologia di allenamento permette di ottenere buoni risultati, più velocemente rispetto ad altri metodi; il problema è che necessita di una quantità di dati molto elevata, la cui raccolta è un’operazione costosa che richiede tempo.

Un paradigma di allenamento che non richiede l’annotazione dei dati raccolti in questa fase è il reinforcement learning (o apprendimento per rinfoco); quest’ultimo è una tecnica di machine learning che cerca di realizzare modelli capaci di apprendere e adattarsi ai cambiamenti dell’ambiente circostante. Questo avviene mediante l’assegnazione di “reward”, ovvero delle ricompense pesate rispetto all’efficacia dell’ultima azione compiuta dal mo-

dello stesso. L’obiettivo del modello è quello di massimizzare il valore di queste ricompense.

Questo tipo di allenamento ha avuto molto successo nel campo del gaming e dell’apprendimento di cinematica dei robot [18].

I principali problemi del Reinforcement Learning si incontrano nel capire cosa fare e come stabilire l’azione corretta in base all’ambiente per massimizzare la ricompensa. In pratica si tratta di problemi a circuito chiuso perché le azioni del modello influenzano i suoi input successivi. Inoltre, al modello non vengono comunicate le azioni da intraprendere, come in molte forme di machine learning (apprendimento supervisionato), ma esso deve scoprire quali azioni generano la ricompensa maggiore provandole. Di norma, le azioni non modificano solo la ricompensa immediata ma anche gli stati successivi e, di conseguenza, tutte le relative ricompense. Questo tipo di modello deve tenere conto anche del problema di trovare il giusto compromesso tra esplorazione e sfruttamento, infatti per massimizzare le ricompense esso deve prediligere le azioni già svolte che restituiscono ricompense elevate, tuttavia per trovare queste ultime deve anche esplorarne di nuove, in modo da poter scegliere azioni sempre migliori in futuro [22].

Poniamo infine l’attenzione sul problema relativo al monitoraggio, cioè come verificare che l’allenamento del modello stia dando effettivamente risultati utili, riuscendo a seguire visivamente i progressi dello stesso. L’utilità

del monitoraggio la troviamo proprio in questo, cioè poter vedere se ci sono problemi senza dover aspettare la fine dell’allenamento, che di solito impiega molto tempo. Proprio quest’ultimo è il problema che questo progetto di tesi si propone di risolvere, tramite un’interfaccia web che permetta di vedere in tempo reale i grafici relativi all’andamento delle ricompense e della durata degli episodi, affiancati dai video che li descrivono singolarmente. Il sito web permette all’utente di registrarsi, accedendo dunque alla propria area personale, in cui può avviare l’allenamento di un modello tramite un algoritmo di reinforcement learning attraverso il quale questo imparerà a giocare a Pong, il famoso arcade della Atari. Una volta avviato l’allenamento sul sito saranno disponibili i relativi dati in tempo reale.

Prima di addentrarci nei dettagli dell’interfaccia, per chiarire meglio cosa vuol dire utilizzare algoritmi di reinforcement learning, vedremo i concetti che ne stanno alla base, trattando anche machine learning, reti neurali e algoritmi quali Q-Learning e Deep Q-Learning.

Capitolo 2

Stato dell'arte

2.1 Machine Learning

Il machine learning (o apprendimento automatico) è quella parte dell’Intelligenza Artificiale che raccoglie l’insieme dei metodi utilizzati per migliorare progressivamente le performance di un algoritmo nello svolgimento di compiti assegnatigli. Il termine fu coniato da alcuni ricercatori nel 1959, ma solo nel 1997, Tom M. Mitchell diede la seguente definizione matematica e relazionale del termine: “Si dice che un programma apprende dall’esperienza E con riferimento a alcune classi di compiti T e con misurazione della performance P , se le sue performance nel compito T , come misurato da P , migliorano con l’esperienza E ” [4]; in altre parole un programma “apprende” se si può vedere un miglioramento delle prestazioni dopo lo svolgimento di un compito. In un esempio applicativo di questa definizione, immaginiamo di voler un programma che riconosca una specifica figura all’interno di un’immagine

(attività T : questo può essere eseguito tramite un algoritmo di machine learning con dati relativi alla stessa figura in altre immagini (esperienza E); se l'apprendimento ha avuto successo, il modello riuscirà a riconoscere la figura nelle immagini che gli verranno sottoposte in futuro (misura di prestazione P). Da questo si può facilmente capire che l'obiettivo principale del machine learning è fare in modo che l'algoritmo, generalizzando dalla propria esperienza, riesca a effettuare ragionamenti di tipo induttivo, cioè riesca a portare a termine con successo compiti che non ha mai svolto basandosi solo sugli esempi di allenamento.

Gli algoritmi di machine learning si possono classificare in tre categorie principali che si differenziano in base alle modalità di apprendimento e ai risultati ottenuti:

1. *Apprendimento supervisionato*: al modello viene passato un set di dati contenente una grande quantità di coppie input/output (dataset), in cui gli input sono esempi di dati dello stesso tipo di quelli che dovranno poi essere analizzati dal modello, gli output invece possono essere valori numerici oppure etichette (classi o tag) che classificano gli input ad essi associati. Tramite questi dataset il modello viene allenato, rendendolo capace di svolgere in modo autonomo tutte le attività inerenti all'allenamento. Questo approccio è analogo all'apprendimento umano

supervisionato da un insegnante.

2. *Apprendimento non supervisionato*: in questo caso al modello si passano dataset in cui sono presenti solo input, senza un output associato.

Sarà quindi compito del modello stesso riorganizzare i dati sulla base di caratteristiche comuni associandogli una classe e cercando di effettuare ragionamenti e previsioni sugli input futuri.

3. *Apprendimento per rinforzo* (o *Reinforcement Learning*): in cui il modello interagisce con un ambiente dinamico nel quale, adattandosi ai cambiamenti, cerca di raggiungere un obiettivo. A guidare il modello sono dei “reward”, ricompense il cui valore è pesato in base all’efficacia dell’ultima azione eseguita.

Questi algoritmi, associati a reti neurali artificiali, possono essere utilizzati per simulare relazioni complesse tra ingresso e uscita che altre funzioni analitiche non riescono a rispettare.

2.2 Reti neurali

Nel campo del machine learning, una rete neurale artificiale (o ANN dall’inglese *artificial neural network*) è un algoritmo utilizzato per risolvere problemi di natura complessa non facilmente codificabili. Queste permettono di apprendere sfruttando meccanismi, in parte, simili a quelli dell’intelligenza

umana, permettendo di ottenere prestazioni impossibili per altri algoritmi [5].

Queste infatti riescono a risolvere determinate categorie di problemi ispirandosi al funzionamento del nostro cervello, e alle volte superandolo e trovando soluzioni a noi inaccessibili. Sono definite reti neurali proprio perché il comportamento dei nodi che le compongono ricorda vagamente quello dei neuroni biologici. Infatti, le reti neurali del cervello sono quelle che ci permettono di comprendere l'ambiente e i suoi mutamenti, e di fornire risposte adattive calibrate sulle esigenze che si presentano. A seconda dell'azione esercitata dai neurotrasmettitori, che si occupano della modulazione degli impulsi nervosi, le sinapsi trasmettono l'impulso nervoso agli altri neuroni o lo bloccano.

Un singolo neurone può ricevere simultaneamente segnali da diverse sinapsi e misurarne il potenziale in modo globale, stabilendo se è stata raggiunta la soglia di attivazione per generare a sua volta un impulso nervoso. La configurazione sinaptica delle reti neurali biologiche è dinamica ed è questo fattore a determinare la loro efficienza. Il numero di sinapsi aumenta o diminuisce in base agli stimoli che riceve la rete. Più ne riceve, maggiori connessioni vengono create, e viceversa. Questo porta ad avere una risposta adattiva più equilibrata. È proprio questo il funzionamento che simulano le reti neurali artificiali.

Nelle reti neurali artificiali, i nodi ricevono dati in input, li processano e sono in grado di inviare le informazioni ad altri neuroni. Attraverso cicli più

o meno numerosi di input-elaborazione-output, in cui gli input presentano variabili differenti, diventano in grado di generalizzare e fornire output corretti associati ad input non facenti parte del training set. Queste reti possono essere immaginate come composte da diversi layer di nodi, ciascuno dei quali è collegato ai nodi del layer successivo, come mostrato in Figura 2.1.

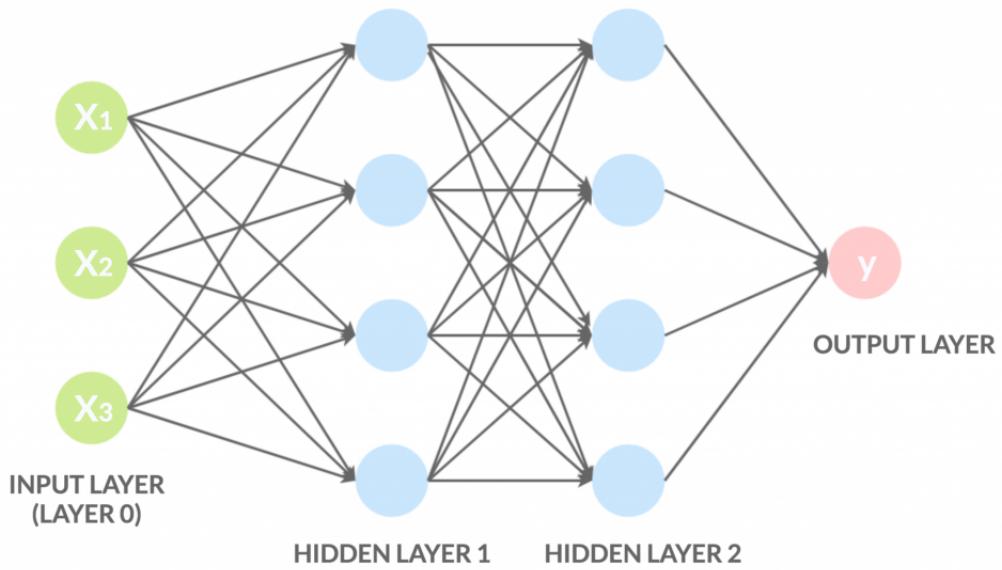


Figura 2.1: Schema di un percepitrone multistrato feedforward

Il numero minimo di strati presenti è tre:

1. *Input layer*: Raccoglie i dati di ingresso.
2. *Hidden layer*: Può esserne presente più di uno e permette l'elaborazione vera e propria.
3. *Output layer*: Presenta il risultato finale.

Le connessioni tra i nodi dei diversi layer sono pesate da fattori moltiplicativi (bias) che rappresentano la “forza” della connessione stessa. Esistono vari tipi di collegamenti tra i nodi:

- *Feedforward*: in queste reti le connessioni collegano i neuroni di un layer con i neuroni del layer successivo. Non sono consentite connessioni all’indietro o verso lo stesso livello.
- *Feedback*: in queste reti sono previste connessioni di feedback verso neuroni dello stesso layer o del precedente. Questo complica il flusso delle informazioni e l’addestramento, dato che richiede di considerare il comportamento in più istanti temporali. Queste reti sono le più adatte per la gestione di sequenze (come ad esempio audio, video o frasi in linguaggio naturale), perché godono di un effetto memoria che al tempo t rende disponibile l’informazione processata a $t - 1$, $t - 2$, ecc. Un particolare tipo di rete neurale ricorrente (RNN) è LSTM (Long Short Term Memory) [6].

Fissato il numero di layer e neuroni della rete, l’addestramento necessario a quest’ultima per funzionare consiste nel determinare il valore dei pesi per ottenere il mapping desiderato tra input e output. Per le reti neurali artificiali valgono algoritmi di allenamento simili a quelli visti per il machine learning, ma con qualche differenza:

- Apprendimento supervisionato: Si fornisce alla rete un dataset di complete input/output in modo che apprenda il legame che li unisce. In questo modo impara ad associare a nuovi input il corretto output. Man mano che la rete elabora le uscite si procede a correggerla, per migliorarne la risposta, variando i pesi (aumentano quelli che determinano le uscite corrette e diminuiscono gli altri). Questo meccanismo si definisce *Error Back-Propagation* (retropropagazione dell'errore). Il problema si riscontra nel determinare un rapporto adeguato tra le dimensione del dataset, quelle della rete e l'abilità a generalizzare richiesta; infatti, un training set troppo ricco e una eccessiva capacità di elaborazione rendono difficile alla rete neurale artificiale la generalizzazione, perché gli input esterni al training set vengono valutati come troppo differenti rispetto a quelli dettagliati che il modello conosce. Viceversa, un training set troppo scarso porta a non avere sufficienti parametri per apprendere a generalizzare. Le reti feedforward MLP come quella in Figura 2.1 utilizzano questo tipo di allenamento [7].

- Apprendimento non supervisionato: Si fornisce alla rete un insieme di variabili di input. Analizzandole, la rete deve creare degli insiemi rappresentativi per categorizzarle. Anche in questo caso il valore dei pesi è dinamico, ma viene modificato dai nodi stessi. Esempi di reti

ad apprendimento supervisionato sono SOM (Self-Organizing Map) e la rete di Hopfield [8] [9] [10].

- Apprendimento per rinforzo: In questo caso non si fornisce alcun dataset alla rete, ma i circuiti neurali imparano esclusivamente dall’interazione con l’ambiente, eseguendo una serie di azioni. In base all’obiettivo, al modello, viene consegnato un premio il cui valore è proporzionale all’efficacia dell’azione appena eseguita; più ci si avvicina efficientemente all’obiettivo, maggiore sarà la ricompensa.

Il singolo nodo di una ANN integra i segnali di ingresso secondo un’apposita funzione di attivazione. Se il valore risultante supera una certa soglia, il nodo trasmette questo valore al layer successivo, altrimenti trasmetterà zero. La funzione di attivazione ha un ruolo fondamentale nel corretto funzionamento della rete ed equivale ad un modello di regressione; questo modello ha lo scopo di trovare un’equazione non lineare che rappresenti i dati in modo abbastanza preciso da spiegarne il comportamento, ma anche abbastanza flessibile da permettere di fare previsioni sui valori di ingresso futuri. Tale funzione è quindi non lineare; inoltre deve essere anche continua e differenziabile quasi ovunque, in modo tale da permettere la Error Back-Propagation.

Le tipologie di funzioni di attivazione sono molteplici, ma le più comuni risultano essere: il gradino, la sigmoide e la rectifier linear unit.

- *Funzione di attivazione a gradino* (Figura 2.2):

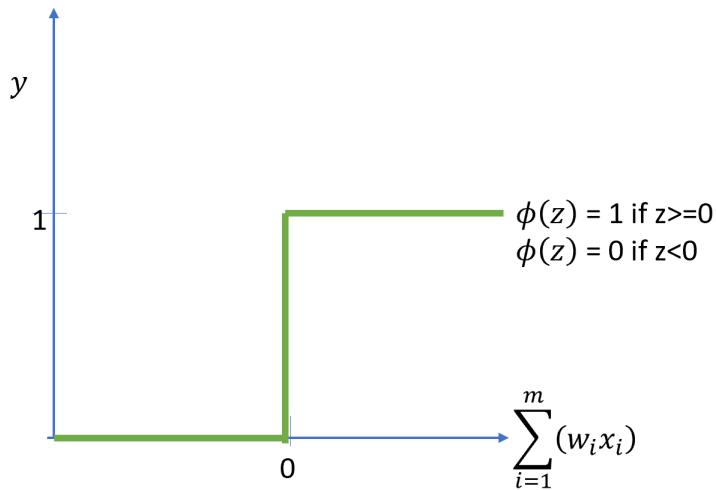


Figura 2.2: Funzione di attivazione a gradino

È quella più simile al funzionamento biologico. Per tutti i valori negativi la risposta rimane zero, mentre sale ad uno appena raggiunge o supera lo zero. In questo modo si ha il vantaggio che è semplice da calcolare, e i valori di output vengono normalizzati tra zero e uno. Nella realtà questa funzione non viene utilizzata a causa della poca stabilità, ma soprattutto perché nel punto in cui cambia valore non è differenziabile, fattore fondamentale nel deep learning, in quanto determina la direzione verso cui orientarsi per l'aggiustamento dei valori. Questo cambiamento brusco, infatti, renderebbe difficile controllare il comportamento della rete, perché potrebbe portare al cambiamento di un peso che migliora il comportamento per un determinato input ma

lo peggiora per altri simili.

- *Funzione di attivazione a sigmoide* (Figura 2.3):

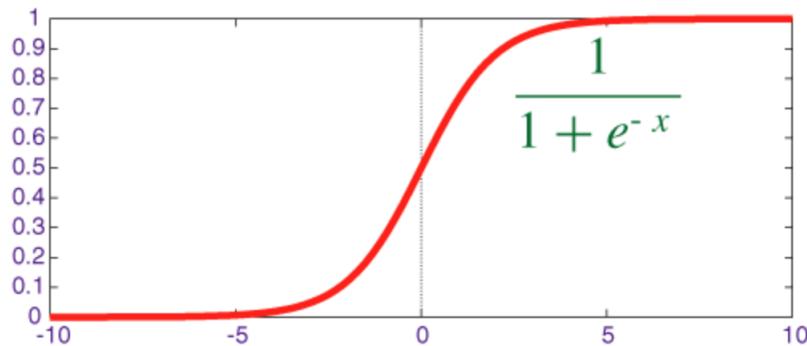


Figura 2.3: Funzione di attivazione a sigmoide

Risolve il problema relativo alla derivabilità del gradino, dato che il passaggio da zero ad uno è più graduale. Il vantaggio di questa funzione è quello di comprimere tutti i valori in un range tra 0 e 1, rendendo il sistema molto stabile anche per grandi variazioni nei valori. Anche questa, però, presenta dei problemi; dato che per valori molto grandi di ingresso la curva è quasi piatta, essa presenta una convergenza lenta con derivata tendente a zero, il che può causare problemi di *vanishing gradient*. Nei punti in cui la derivata della funzione si trova vicino allo zero, infatti, avremo una variazione dei pesi molto piccola (scomparsa del gradiente) e quindi nel complesso un apprendimento lento; possiamo infatti notare che applicando questa funzione nei layer nascosti, quelli rispettivamente più vicini al layer di ingresso imparano molto più

lentamente rispetto ai più lontani [12]. Questo problema la rende poco adatta ai layer intermedi, ma rimane molto valida per quello di uscita [11].

- *Funzione di attivazione Rectifier Linear Unit (ReLU)* (Figura 2.4):

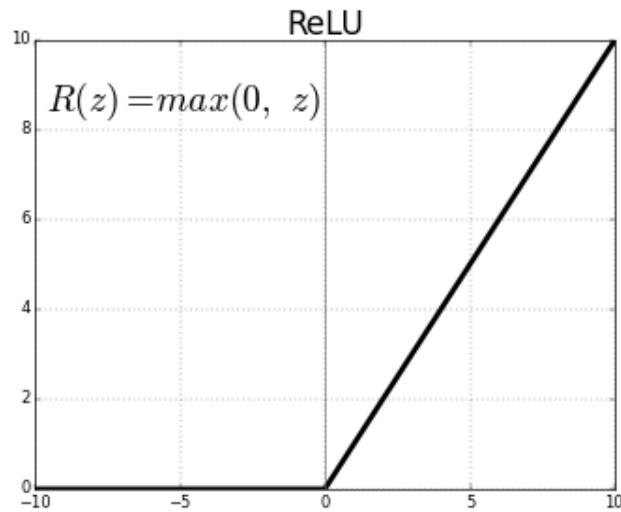


Figura 2.4: Funzione di attivazione a ReLU

Questa funzione appiattisce a zero la risposta a tutti i valori negativi, mentre la lascia invariata per quelli positivi. Grazie al fatto che è molto semplice da calcolare e che riduce di molto il problema del vanishing gradient, questa è molto utilizzata nei layer intermedi delle ANN in cui sono presenti molti calcoli e passaggi [13].

Una rete neurale artificiale può imparare sfruttando il metodo di apprendimento con discesa del gradiente, con l'obiettivo di minimizzare una funzione di costo che rappresenta l'errore commesso dal modello. Una derivata ele-

vata significa pendenza elevata e quindi esso è ancora lontano dal minimo; viceversa una derivata piccola significa pendenza lieve, il che ci indica che si trova vicino al minimo, quindi i passi di avvicinamento sono sempre più piccoli [14].

2.3 Reinforcement Learning

Come abbiamo già accennato nell'introduzione, il reinforcement learning ha avuto molto successo nel campo dell'apprendimento della cinematica dei robot e dei videogame. Riguardo la prima applicazione, questa è una parte cruciale della ricerca sull'IA. Il problema si concentra sull'apprendimento di come coordinare e controllare un corpo flessibile per risolvere compiti in ambienti complessi. I tentativi esistenti di controllare i corpi umanoidi provengono da campi come l'animazione al computer [19] e la biomeccanica [20]. Un metodo molto utilizzato è quello del motion capture [21] che permette di produrre comportamenti specifici; tuttavia questo può generare comportamenti limitanti o comunque difficilmente riutilizzabili per nuove attività. Infatti, in generale, la difficoltà di descrivere accuratamente un comportamento complesso è una sfida comune quando si insegnano le capacità motorie a un sistema artificiale. È stato testato che buoni risultati possono emergere da zero dal corpo interagendo con l'ambiente e utilizzando solo obiettivi semplici come andare avanti senza cadere. Un fattore fondamentale che incide quando

si utilizza il paradigma di Reinforcement Learning è la progettazione della funzione di ricompensa per incoraggiare una particolare soluzione. Nel caso specifico del movimento si è utilizzata una semplice funzione ricompensa basata sui passi avanti [15]. In particolare, sono stati addestrati agenti con una varietà di corpi simulati per fare progressi su diversi tipi di terreni. Da questi risultati emerge un comportamento molto robusto ma poco umano, per cui si è pensato di addestrare modelli ad imitare dati provenienti da motion capture, relativi a camminare, alzarsi da terra, correre e girare. Questi comportamenti più “umani” sono stati poi riutilizzati per eseguire altri compiti [16]. Un’altra possibilità che viene proposta è una rete neurale artificiale basata su modelli generativi all'avanguardia, in grado di apprendere le relazioni tra i diversi comportamenti e di imitare azioni specifiche. Dopo l'allenamento di quest'ultima, il modello riesce a codificare una singola azione osservata e creare un nuovo movimento basandosi su di essa, riuscendo anche a passare da un comportamento ad un altro senza però aver mai osservato una transizione tra di essi [18]. Tramite l'utilizzo di modelli generativi, si hanno a disposizione dati sufficienti per applicare anche approcci supervisionati per eseguire l'apprendimento dell'imitazione; in questo modo però si hanno vulnerabilità ad errori a cascata quando la traiettoria del modello diverge da quella della dimostrazione [17]. Per quanto riguarda invece il campo dei videogame, sono stati utilizzati alcuni giochi Atari 2600 implementati in The

Arcade Learning Environment (ALE), con l’obiettivo di creare un unico modello di rete neurale che, tramite un ingresso visivo, cioè il frame che descrive lo stato attuale della partita, sia in grado di giocare bene al più alto numero possibile di giochi. Sotto questo aspetto il reinforcement learning presenta numerose sfide rispetto all’apprendimento supervisionato. Le applicazioni di maggior successo di quest’ultimo infatti, non solo hanno avuto bisogno di una grande quantità di dati di addestramento etichettati, ma si presuppone anche che i campioni di dati siano indipendenti e la loro distribuzione sia fissa. Gli algoritmi di reinforcement learning al contrario devono essere in grado di apprendere da un segnale scalare (ricompensa) che è spesso sporadico, rumoroso e ritardato. Il ritardo tra azioni e ricompense risultanti, che può essere molto grande, sembra particolarmente scoraggiante se paragonato all’associazione diretta tra input e output riscontrata nell’apprendimento supervisionato. Vediamo inoltre che le sequenze di stati che si incontrano sono altamente correlate e la distribuzione dei dati varia man mano che il modello apprende nuovi comportamenti. Nel caso preso in esame, consideriamo un modello che interagisce con l’ambiente ε , rappresentato dall’emulatore Atari, attraverso una sequenza di azioni, osservazioni e ricompense. In ogni istante il modello seleziona un’azione a_t dal set di quelle consentite, $A = \{1, \dots, K\}$. L’azione viene passata all’emulatore e modifica il suo stato interno ed il punteggio. Lo stato interno dell’emulatore non è però visibile al modello, esso

vedo solo un’immagine $x_t \in \mathbb{R}^d$ restituita dall’emulatore, rappresentata da un vettore di pixel e che rappresenta lo schermo corrente. Inoltre esso riceve una ricompensa r_t che rappresenta il cambiamento del punteggio. Si noti che in generale il punteggio del gioco può dipendere dall’intera sequenza precedente di azioni e osservazioni; il feedback su un’azione può essere ricevuto solo dopo che sono trascorse migliaia di istanti. Pertanto consideriamo sequenze di azioni e osservazioni $s_1 = x_1, a_1, x_2, \dots, a_{t-1}, x_t$, e impariamo strategie di gioco che dipendono da queste. Tutte le sequenze nell’emulatore sono assunte per terminare in un numero finito di istanti. Questo formalismo porta ad un ampio, ma finito, processo decisionale di Markov (MDP) [24] in cui ogni sequenza è uno stato distinto. Come risultato possiamo applicare i metodi di apprendimento standard per gli MDP, utilizzando le sequenze complete s_t come rappresentazioni dello stato all’istante t . L’obiettivo del modello è quello di selezionare le azioni che massimizzano le ricompense future. Assumiamo che le ricompense future siano pesate con un fattore γ ; definiamo le ricompense future attualizzate all’istante t come $R_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$, dove T è l’istante in cui termina il gioco. Definiamo, inoltre, la funzione azione-valore ottimale $Q^*(s', a')$ come il massimo rendimento atteso ottenibile seguendo qualsiasi strategia, dopo aver visto qualche sequenza s ed eseguito alcune azioni a , $Q^*(s, a) = \max_{\pi} E[R_t | s_t = s, a_t = a, \pi]$, dove π è la politica usata per mappare le sequenze in azioni. Questa funzione appena descritta rispet-

ta un importante identità conosciuta come *equazione di Bellman* [25]. Nella pratica applica il principio di ottimalità; tale principio segue l'idea che se il valore ottimale di $Q^*(s', a')$ della sequenza s' all'istante successivo era conosciuto per tutte le possibili azioni a' , allora la strategia ottimale è quella di selezionare a' che massimizza il valore atteso di $r + \gamma Q^*(s', a')$ [23].

L'idea base dietro la maggior parte degli algoritmi di reinforcement learning sarebbe quella di stimare la funzione azione-valore, aggiornandola ad ogni iterazione tramite l'equazione di Bellman. Questo approccio è purtroppo irrealizzabile; dato che tale funzione viene stimata separatamente per ogni sequenza, senza alcuna generalizzazione. Piuttosto è comune utilizzare una funzione di attivazione (non lineare in caso di reti neurali artificiali) per stimare la funzione azione-valore, $Q(s, a; \theta) \approx Q^*(s, a)$. Definiamo *Q-Network* una ANN che ha una funzione di attivazione con pesi θ . Tale rete può essere allenata minimizzando una sequenza di funzioni di perdita $L_i(\theta_i)$ che cambia ad ogni iterazione i modificando i relativi pesi. Se questi ultimi sono aggiornati dopo ogni istante e i nuovi obiettivi sono sostituiti da singoli campioni provenienti da distribuzioni del comportamento ρ e dall'emulatore ε rispettivamente, allora si tratta i algoritmi Q-Learning [26] [23]. Si noti che questi algoritmi sono privi di un modello dell'ambiente; essi risolvono i compiti direttamente usando campioni dall'emulatore ε , senza costruire esplicitamente una stima di ε .

Probabilmente l'episodio riguardo il reinforcement learning applicato ai giochi più conosciuto è TD-gammon, in cui un software ha imparato a giocare a backgammon interamente tramite RL, raggiungendo alla fine una bravura superiore a quella di qualsiasi giocatore umano. TD-gammon utilizzava un algoritmo di rinforzo privo di modello simile al Q-Learning, approssimando il valore della funzione utilizzando un percepitrone multistrato con un layer nascosto.

Capitolo 3

Deep Q-Learning

Le recenti scoperte nel campo della computer vision e dello speech recognition (riconoscimento vocale) si sono affidate al sempre più efficiente allenamento di reti neurali profonde (con due o più layer nascosti) su set di dati molto ampi. Gli approcci di maggior successo vengono addestrati direttamente con input non elaborati, usando aggiornamenti leggeri basati sulla discesa stocastica del gradiente. Fornendo a questa tipologia di reti sufficienti dati, è spesso possibile acquisire migliori rappresentazioni rispetto a quelle ottenute manualmente (che vanno dopo passate al modello) [30]. L'approccio descritto fino ad ora, utilizzato per permettere al modello di giocare, lega un algoritmo di reinforcement learning ad una rete neurale profonda che opera direttamente su immagini RGB ed elabora in modo efficiente i dati di allenamento utilizzando il metodo della discesa stocastica del gradiente.

L'architettura TD-Gammon di Tesauro ha fornito un punto di partenza per tale approccio. Questa architettura aggiorna i parametri di una rete che

stima i valori di una funzione, direttamente da campioni, $s_t, a_t, r_t, s_{t+1}, a_{t+1}$, presi rispetto alle scelte effettuate dalla policy interagendo con l’ambiente (o giocando se consideriamo ad esempio il caso del backgammon). Al contrario del TD-Gammon e di altri approcci simili che si possono trovare in rete, per l’allenamento sui giochi Atari è stato utilizzata una tecnica nota come “experience replay” [31], in cui si memorizza l’esperienza del modello in ogni fase, $e_t = (s_t, a_t, r_t, s_{t+1})$, in un set di dati $D = (e_1, \dots, e_N)$, quest’ultimi raggruppati come episodi in una *replay memory* (normalmente un buffer ciclico in cui inserire i dati per poterli poi recuperare in un secondo momento). Durante il ciclo interno dell’algoritmo, vengono applicati gli aggiornamenti Q-Learning (o aggiornamenti minibatch), tramite campioni di esperienza estratti casualmente tra quelli memorizzati. Dopo aver compiuto l’aggiornamento appena descritto, il modello seleziona ed esegue un’azione secondo la ϵ -greedy policy; questo algoritmo, in generale, prevede una parte iniziale in cui si provano le varie azioni e si ottengono le relative ricompense, dopodichè per ogni azione si calcola un valore relativo all’efficacia della stessa e, scelto un valore di ϵ compreso tra 0 e 1 (generalmente molto piccolo, es. 0.1), si genera un valore casuale sempre compreso tra 0 e 1: se questo valore supera ϵ allora si esegue l’azione per cui il valore calcolato è il più grande, altrimenti si esegue, con uguale probabilità, una qualsiasi tra le possibili azioni [32]. Poiché l’utilizzo di episodi con lunghezza arbitraria come input per la rete neurale può essere

difficoltoso, nel caso specifico si è utilizzata una funzione Q che lavora su un numero n di episodi prodotti da una funzione φ . L'algoritmo completo è chiamato *Deep Q-Learning*.

Questo approccio presenta numerosi vantaggi rispetto al Q-learning standard [22]. Per prima cosa ogni elemento della replay memory, che rappresenta l'esperienza del modello, può essere potenzialmente utilizzato in molti aggiornamenti dei pesi, il che consente una maggiore efficienza dei dati. In secondo luogo, l'apprendimento effettuato attraverso campioni consecutivi è inefficiente, a causa delle forti correlazioni tra i campioni; la randomizzazione dei campioni interrompe queste correlazioni e quindi riduce la varianza degli aggiornamenti. In terzo luogo, quando si apprende attraverso la policy, i parametri correnti determinano il prossimo campione di dati su cui vengono formati i parametri. Ad esempio, se l'azione col valore massimo (più efficiente) prevede di spostarsi a sinistra, i campioni di allenamento saranno dominati da campioni dal lato sinistro; se l'azione col valore massimo passa poi a destra, cambierà anche la distribuzione dell'allenamento. Tuttavia potrebbe succedere che il modello si blocchi in loop indesiderati e i parametri rimangano fermi in un minimo locale o addirittura divergano [33]. Utilizzando l'experience replay, la distribuzione del comportamento viene mediata su molti dei suoi stati precedenti, appianando l'apprendimento ed evitando oscillazioni o divergenze nei parametri. Si noti che quando si apprende me-

diante l’experience replay è necessario apprendere off-policy (poiché i nostri parametri sono diversi da quelli utilizzati per generare il campione), il che motiva la scelta del Q-Learning [22].

In pratica, l’algoritmo presentato memorizza solo gli ultimi N elementi di esperienza nella replay memory, e campiona in modo casuale e uniforme da D quando si effettuano gli aggiornamenti dei parametri. Questo approccio è per alcuni aspetti limitato poiché il buffer di memoria non differenzia le transizioni importanti e sovrascrive sempre con transizioni recenti a causa della dimensione N finita della memoria. Allo stesso modo, il campionamento uniforme dà uguale importanza a tutte le transizioni nella memoria di riproduzione. Una strategia di campionamento più sofisticata potrebbe enfatizzare le transizioni dalle quali possiamo apprendere di più, in modo simile al *prioritized sweeping* [34]. Le *reti neurali convoluzionali* (CNN dall’inglese *Convolutional Neural Network*) che vengono allenate con questo approccio sono definite *Deep Q-Networks* (DQN) [23]. Le CNN sono un tipo di ANN feed-forward in cui il pattern di connettività tra i neuroni è ispirato dall’organizzazione della corteccia visiva animale, e sono molto utilizzate nel riconoscimento di immagini e video [2], nei sistemi di raccomandazione [35], nell’elaborazione del linguaggio naturale [3] e, recentemente, in bioinformatica (una disciplina dedicata alla risoluzione di problemi biologici a livello molecolare con metodi informatici) [36].

3.1 Allenamento

Dalla documentazione consultata si vede che sono stati fatti esperimenti su sette, tra i più popolari, giochi ATARI: Beam Rider, Breakout, Enduro, Pong, Q * bert, Seaquest, Space Invaders. Per tutti e sette sono stati utilizzati: stesso algoritmo di apprendimento, stessa architettura di rete e stesse impostazioni di iperparametri; in modo da dimostrare che l'approccio usato è abbastanza robusto da funzionare su una grande varietà di giochi, senza dover inserire dati specifici. La valutazione dei modelli è avvenuta sui giochi originali, tuttavia, durante l'allenamento, è stata apportata una modifica alla struttura delle ricompense. Questo perché la scala dei punteggi varia notevolmente da un gioco all'altro, quindi sono state impostate tutte le ricompense positive ad 1, quelle negative a -1 e lasciando invariate le ricompense nulle; in questo modo si permette l'utilizzo della stessa velocità di apprendimento su più giochi. Allo stesso tempo, però, potrebbe influire sulle prestazioni del modello in quanto non è in grado di distinguere tra premi di diversa entità. In questi esperimenti è stato utilizzato l'algoritmo *RMSProp* [27], un algoritmo di ottimizzazione progettato per reti neurali, con *minibatch* di 32 elementi; col termine *minibatch* si intendono i sottogruppi in cui vengono suddivisi dati di allenamento, per esempio, supponiamo di avere dei dati di addestramento che abbiano 32000 istanze e che la dimensione del minibatch

sia di 32, allora avremo 1000 minibatch. L’allenamento ha avuto una durata in frame pari a 10 milioni; inoltre il modello selezionava un’azione ogni k fotogrammi, anzichè ogni fotogramma, questo ha permesso di giocare k volte in più senza aumentare significativamente il tempo di esecuzione. Per tutti i giochi tranne Space Invaders è stato utilizzato $k = 4$, infatti in quest’ultimo si è notato che per questo valore i laser risultavano invisibili a causa del periodo in cui lampeggiano; per risolvere il problema si è quindi usato $k = 3$, questa è stata l’unica modifica tra i valori degli iperparametri.

Nell’apprendimento supervisionato è facile tracciare le prestazioni di un modello durante l’allenamento valutandolo sui set di addestramento. Nell’apprendimento per rinforzo, invece, valutare accuratamente i progressi di un agente durante l’allenamento è più impegnativo; poiché nel caso d’uso considerato, la metrica di valutazione utilizzata è la ricompensa totale raccolta durante un episodio o un gioco calcolata su un numero di giochi, essa viene calcolata periodicamente durante l’allenamento. Questo metodo di valutazione può risultare poco accurato considerando che piccoli cambiamenti nei pesi possono portare a grandi cambiamenti sulle azioni del modello.

I primi due grafici da sinistra in Figura 3.1 come la ricompensa totale media si evolve durante l’allenamento nei giochi Seaquest e Breakout; in entrambi non si vede una netta crescita, dando l’impressione che l’algoritmo di apprendimento non stia facendo progressi costanti. Un’altra metrica, più

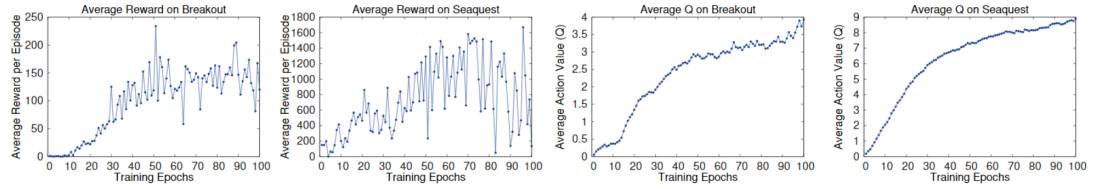


Figura 3.1: Andamento metriche di valutazione

stabile, è la stima della funzione azione-valore Q della policy; questa fornisce una stima del fattore di sconto ottenibile seguendo una policy a partire da un determinato stato. Vengono raccolti un set fisso di stati eseguendo una politica casuale prima dell'inizio dell'allenamento e monitorata la media dei valori massimi stimati dalla funzione Q per questi stati. I due grafici sulla destra in Figura 3.1 mostrano che la Q media prevista aumenta in modo molto più regolare rispetto alla ricompensa totale media ottenuta dal modello; lo stesso andamento si ottiene sugli altri giochi. In aggiunta ad un miglioramento abbastanza regolare della Q prevista durante l'allenamento, non sono stati riscontrati problemi di divergenza [28]. Tuttavia uno svantaggio nell'utilizzo della funzione Q per valutare le prestazioni dell'algoritmo è che questa viene stimata dal modello stesso, per cui meno attendibile rispetto alla valutazione attraverso i reward. I risultati ottenuti dimostrano che il metodo utilizzato è valido per l'addestramento di grandi reti neurali utilizzando il reinforcement learning e la discesa stocastica del gradiente [29] [23].

Capitolo 4

Piattaforma web per il monitoraggio di algoritmi di reinforcement learning

L'elaborato sviluppato si propone di risolvere la problematica relativa al monitoraggio del paradigma Reinforcement Learning già introdotta nel capitolo 1. Per fare ciò mette a disposizione un'interfaccia web con le seguenti funzionalità generali:

- Servizio di registrazione, autenticazione e recupero delle credenziali.
- Avvio dell'allenamento del modello sul gioco Pong dell'Atari.
- Grafici relativi a durata e reward dei singoli episodi con la possibilità di riprodurre a video ogni episodio, il tutto in tempo reale.
- Caricare e visionare vecchi allenamenti ancora in memoria con la possibilità di eliminarli.

Questa interfaccia web è stata realizzata mediante la release 6.0 del framework Laravel, un framework open source specializzato nello sviluppo di applicazioni web. Nel dettaglio viene presentata la seguente struttura delle pagine:

- Autenticazione, registrazione e recupero delle credenziali.
- Home.
- Archivio.

4.1 Autenticazione, registrazione e recupero delle credenziali

La pagina di registrazione si presenta all’utente come mostrato in Figura 4.1.

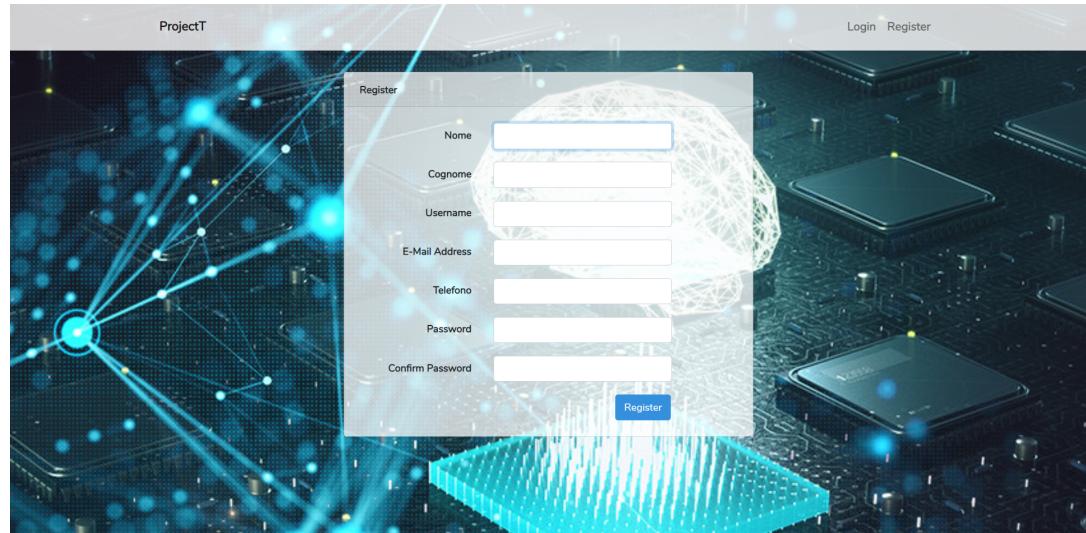


Figura 4.1: Pagina di registrazione.

L’operazione di registrazione è indispensabile per l’utente ai fini dell’utilizzo del sistema. Questa permette l’inserimento dei seguenti dati personali:

- Nome e cognome dell’utente.
- Username, nome con cui si fa riferimento all’utente all’interno del sito.

Questo campo è univoco e la pagina implementa un controllo aggiuntivo su quest’ultimo che permette di determinare se l’username inserito è disponibile o meno: ciò avviene semplicemente inserendo la parola all’interno del campo e deselezionandolo, con quest’ultimo che diventerà verde se l’username è disponibile o rosso se non lo è.

- Indirizzo email.
- Numero di telefono.
- Password; inoltre è presente un secondo campo che permette di confermare la password inserita in modo da evitare errori.

Tutti i dati sopra elencati vengono inviati al server che, dopo un controllo aggiuntivo, li salva su database, nella tabella “*users*” dedicata agli utenti (4.10); in particolare è stato utilizzato il database relazionale MySQL. La comunicazione tra client e server avviene tramite richieste asincrone, cioè il client non si mette in attesa della risposta, così da non bloccare la pagina.

La pagina di autenticazione si presenta all’utente come mostrato in Figura 4.2.

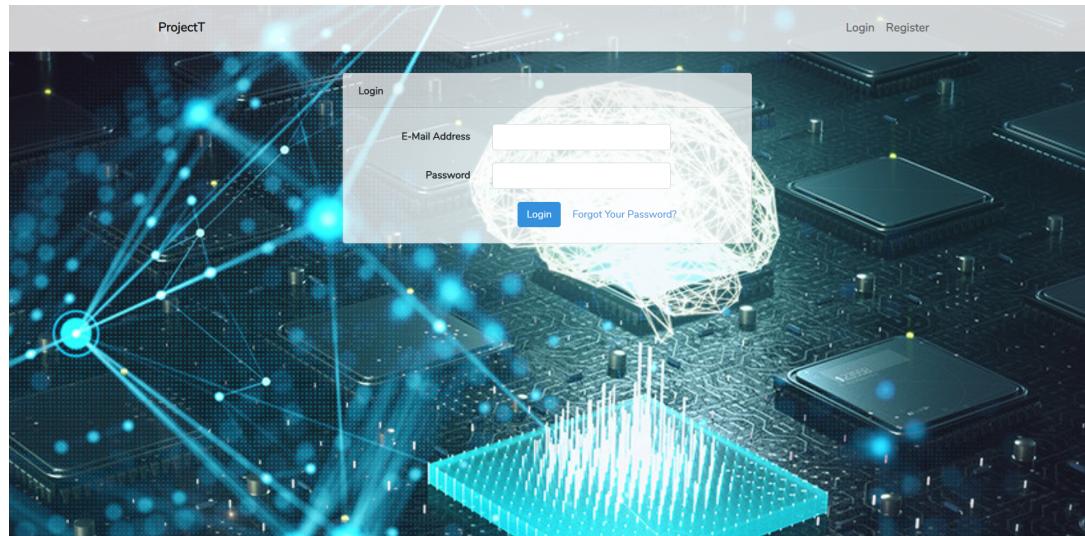


Figura 4.2: Pagina di autenticazione.

Come vediamo dalla figura l’autenticazione avviene tramite i campi email e password precedentemente salvati su database tramite la procedura di registrazione. Premendo invece su “*Forgot Your Password?*” si viene reindirizzati alla pagina di recupero credenziali (Figura 4.3).

A screenshot of a web browser showing a 'Reset Password' page. The page has a light gray header with 'Login' and 'Register' links. Below it is a 'Reset Password' form. The form contains an 'E-Mail Address' input field, which is currently empty. Below the input field is a blue 'Send Password Reset Link' button. The overall design is clean and modern.

Figura 4.3: Pagina per il recupero delle credenziali.

Questa permette l’inserimento dell’email e, dopo la verifica che sul database sia presente, invia alla stessa una mail contenente il link per l’inserimento

di una nuova password da sostituire alla vecchia (Figura 4.4).

The image shows a web-based password reset form titled "Reset Password". It contains three text input fields: "E-Mail Address", "Password", and "Confirm Password". Below these fields is a blue rectangular button labeled "Reset Password". In the top right corner of the form area, there are two small links: "Login" and "Register". The entire form is set against a light gray background.

Figura 4.4: Pagina per l'inserimento della nuova password.

Una volta completata questa, l'autenticazione oppure la registrazione si viene automaticamente reindirizzati alla home del sito; al contempo viene creata una sessione che tiene conto dell'id utente in modo da accertarne l'identità durante la navigazione sullo stesso.

4.2 Home

Subito dopo l'accesso la home del sito si presenta all'utente come mostrato in Figura 4.5.

Questa pagina mostra in alto a sinistra l'username dell'utente che ha fatto l'accesso (recuperato grazie ai dati di sessione); in alto a destra, invece, sono presenti i seguenti pulsanti:

- *Logout*: Permette all'utente l'uscita dal sito eliminando i dati di sessione; per poter ritornare alla home è necessaria nuovamente l'autenticazione.

- *Archivio:* Reindirizza l'utente ad una pagina in cui sono presenti i vecchi allenamenti del modello non ancora eliminati (paragrafo 4.3).

Più in basso troviamo il nome del gioco su cui verrà effettuato l'allenamento, affiancato dai pulsanti “*START*” e “*STOP*”; questi servono rispettivamente ad avviare l'allenamento ed a interromperlo.

Premendo su “*START*” il client invia la relativa richiesta al server; quest'ultimo:

1. Crea sul sistema una cartella il cui nome è l'identificativo dell'allenamento, su cui verranno salvati i dati relativi allo stesso. I dati sono suddivisi in cartelle che rappresentano i diversi episodi giocati dal modello, all'interno delle quali vengono salvati, dall'algoritmo di allenamento, i



Figura 4.5: Home del sito quando non sono attivi allenamenti.

frame della partita; questi frame, nella pratica, rappresentano gli stati dell’ambiente di allenamento.

2. Salva all’interno della tabella “*history*” del database le seguenti informazioni:

- *id*: Identificativo univoco dell’allenamento.
- *PID (Process Identifier)*: Identificativo univoco relativo al processo che si occupa dell’allenamento.
- *user id*: Identificativo univoco dell’utente.
- *directory*: Il percorso assoluto della cartella creata al passo 1.
- *active*: Valore booleano (cioè può essere solo vero o falso) che indica se l’allenamento è attualmente attivo: questo viene impostato su “vero” quando si preme start e su “falso” quando si preme stop (o termina l’allenamento).

3. Crea il processo a cui passa l’algoritmo del modello e la cartella su cui salvare i dati dell’allenamento.

Allo stesso tempo, lato client, il sito appare all’utente come in Figura 4.6; la pagina in questa fase richiede al server, ad intervalli di tempo regolari, la presenza di nuovi aggiornamenti da mostrare all’utente.

Il server, ad ogni richiesta ricevuta, controlla all'interno della cartella passata all'algoritmo se sono stati completati nuovi episodi; nel caso in cui sono presenti, si occuperà di:

1. Recuperare i percorsi assoluti di tutti i frame dell'episodio, ordinarli e scriverli su un file di testo.
2. Avviare un'istanza di *ffmpeg* (software utilizzato per l'elaborazione di audio e video) a cui passare il file appena creato in modo che questa generi un video a partire dai frame.
3. Recuperare i dati relativi al reward e alla durata (in frame) dell'episodio da inserire nei grafici.
4. Inviare al client i dati raccolti e i percorsi dei video di ogni episodio.

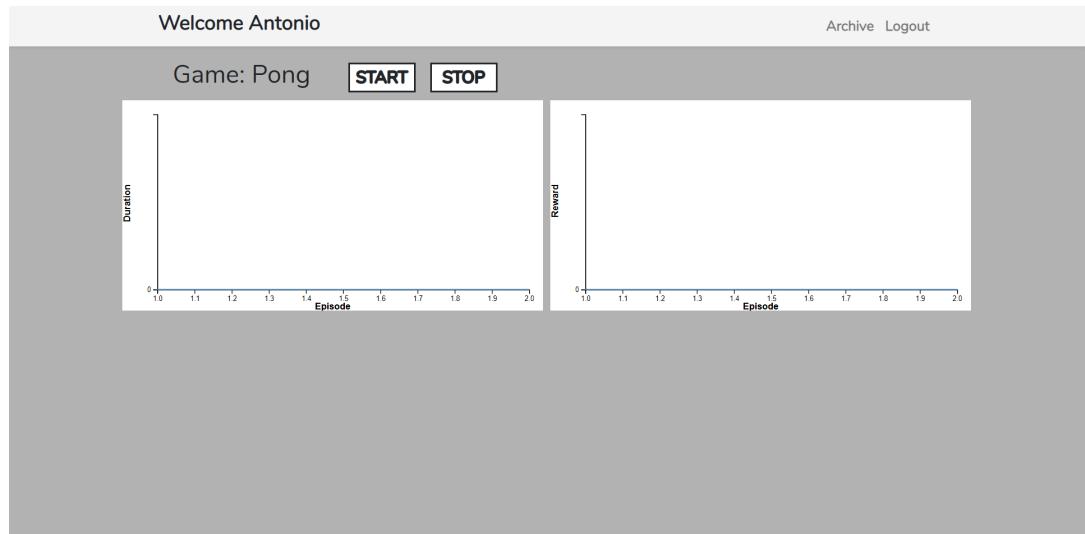


Figura 4.6: Home del sito quando si preme il pulsante start.

Il server inoltre elimina tutti i frame che sono stati già usati per i video in modo da ottimizzare lo spazio su disco.

Il client gestisce i dati ricevuti riempiendo la home come in Figura 4.7, rendendoli consultabili dall’utente.

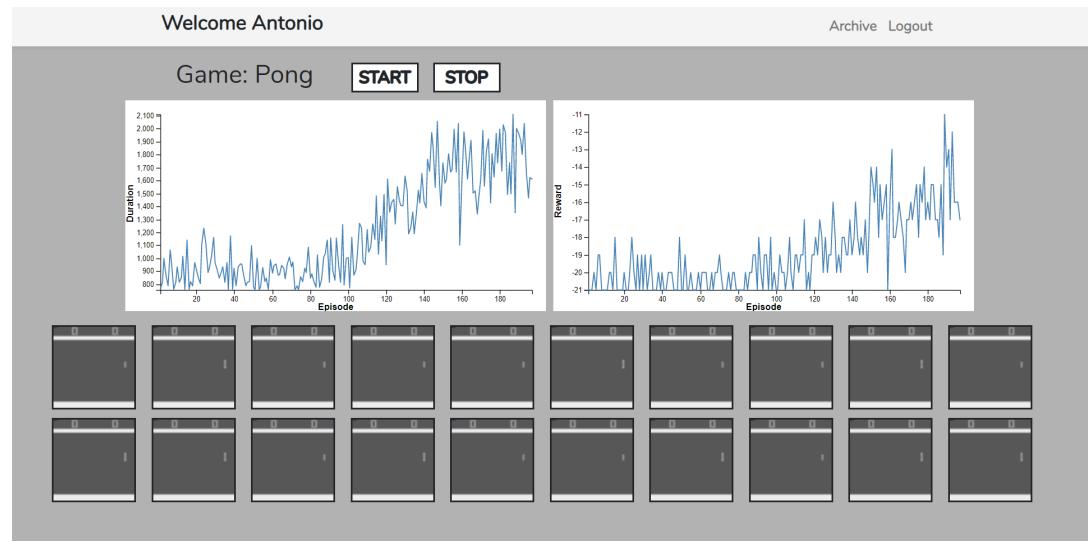


Figura 4.7: Home del sito durante l’allenamento.

In particolare, i grafici sono interattivi, cioè passando il mouse sopra la curva mostra il valore di ascisse e ordinate; i video mostrati sono sempre 20 e vengono selezionati a distanza regolare tra il primo e l’ultimo episodio generato durante l’allenamento. Cliccando sul video si aprirà una vista come in Figura 4.8 che ne permette la visione.

4.3 Archivio

La sezione archivio, raggiungibile dall'utente tramite il pulsante “Archive” della home, si mostra come in Figura 4.9. Questa pagina permette all'utente di visualizzare sulla home gli allenamenti già conclusi in passato semplicemente cliccando su quello che si desidera vedere. Gli allenamenti sono indicati tramite la data e l'ora dell'avvio. Inoltre cliccando sull'icona del cestino, che compare solo trascinando il mouse sull'episodio, è possibile eliminarlo dal sistema rimuovendo anche la relativa istanza dal database.

Lo schema relazionale del database è mostrato in Figura 4.10.

Si vede che il campo *user id* della tabella *history* è legato al campo *id* della tabella *users*, questo perchè è presente una “foreign key” che lega i



Figura 4.8: Home del sito durante la riproduzione.

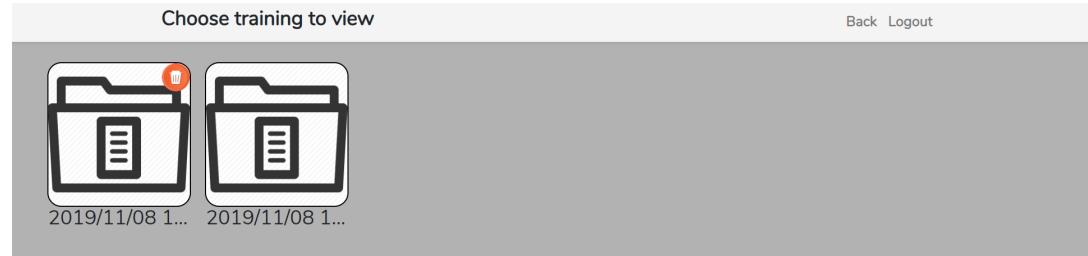


Figura 4.9: Archivio degli allenamenti conclusi.

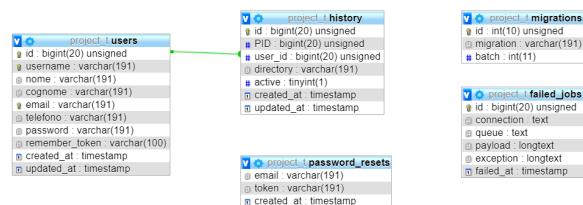


Figura 4.10: Schema relazionale del database.

due, cioè non può esistere un valore nella colonna user id che non è presente nella colonna id. Le altre tabelle di cui non si è parlato sono utilizzate dal framework Laravel.

Capitolo 5

Risultati e conclusioni

Lo scopo di questa tesi è quello di proporre una possibile soluzione al problema inerente al monitoraggio del modello durante l’allenamento attraverso il paradigma del reinforcement learning. Possiamo vedere infatti come si evolvono le sue capacità nello svolgere il compito assegnatogli, ovvero, nel caso specifico, imparare a giocare a Pong. Inizialmente, come mostrato in Figura 5.1, i valori delle ricompense sono molto negativi perché il modello non ha ancora appreso a giocare, tantochè il risultato finale delle prime partite è quasi sempre 21 a 0 (in cui 0 è il punteggio del modello) (Figura 5.2).

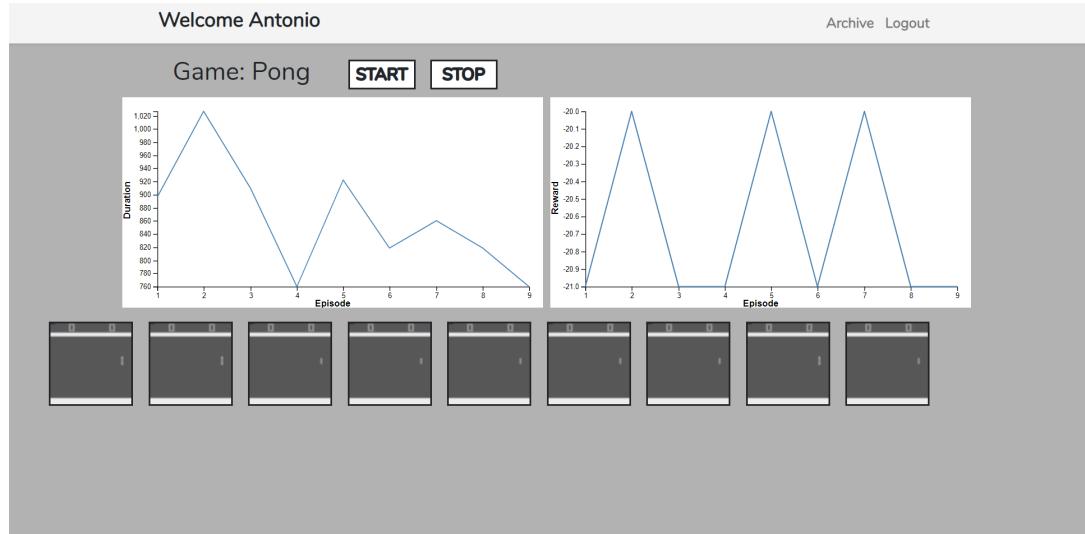


Figura 5.1: Andamenti della durata (in frame) e dei reward iniziali.



Figura 5.2: Risultato finale delle prime partite.

Dopo le prime 4 ore di allenamento notiamo alcuni progressi: come dimostra la Figura 5.3, la durata e i reward sono aumentati. Questo significa che il modello sta effettivamente imparando dall'esperienza e che l'algoritmo di

allenamento funziona, dato che le partite si concludono orientativamente con un punteggio di 21 a 15 (in cui 15 è il punteggio del modello) (Figura 5.4).

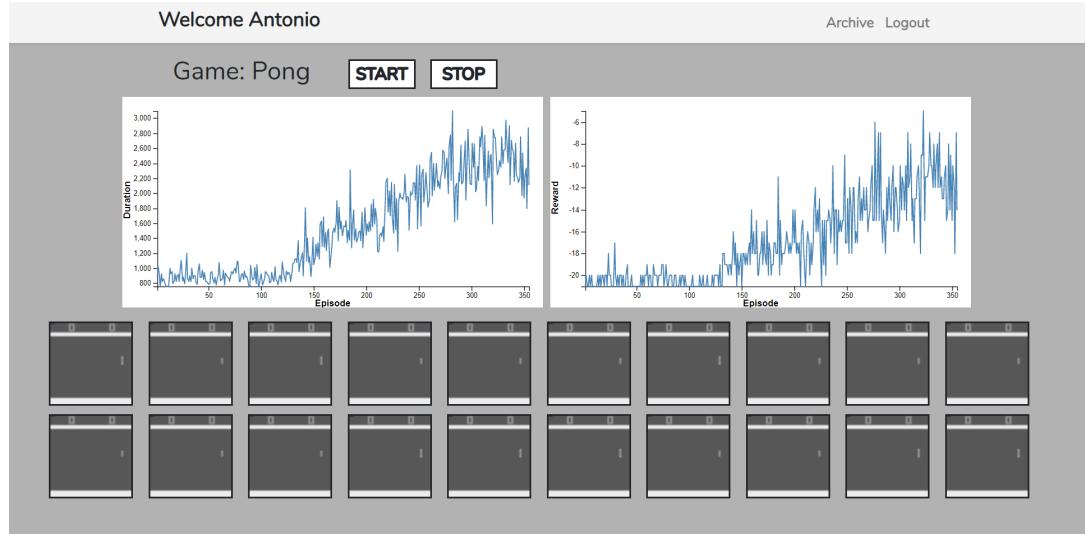


Figura 5.3: Andamenti della durata (in frame) e dei reward dopo le prime 4 ore.



Figura 5.4: Risultato finale dopo le prime 4 ore.

Proseguendo ad osservare l'allenamento, in un tempo relativamente ri-

stretto di circa 10 ore, possiamo notare che il modello ottiene vittorie schiaccianti (la Figura 5.6 mostra un risultato di 1 a 21 a suo favore), infatti, come si vede in Figura 5.5, la durata delle partite diminuisce progressivamente mentre il valore delle ricompense aumenta gradualmente fino a stabilizzarsi intorno al valore massimo.

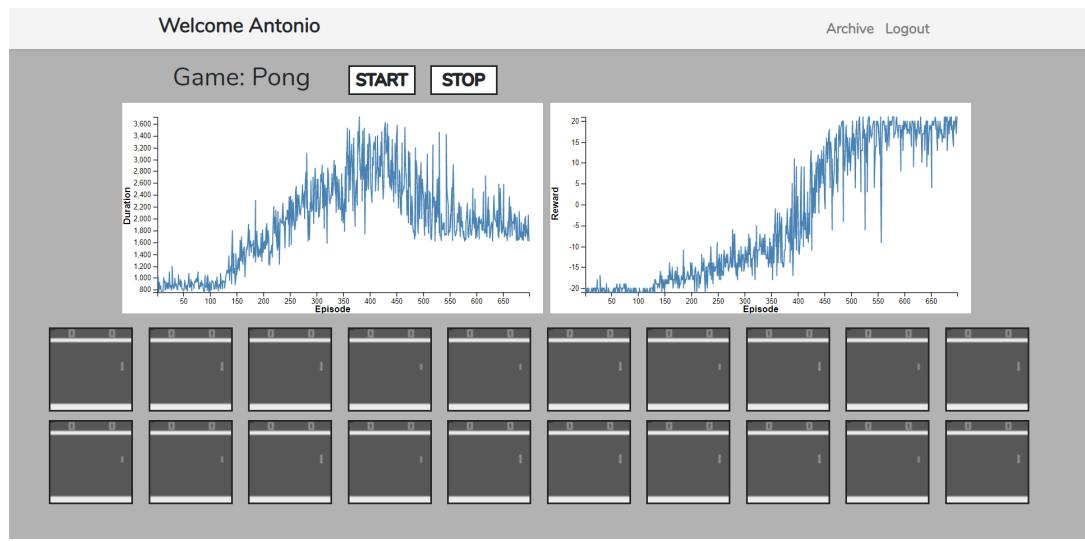


Figura 5.5: Andamenti della durata (in frame) e dei reward dopo circa 10 ore.



Figura 5.6: Risultato finale delle partite dopo circa 10 ore.

5.1 Conclusioni

Alla luce dei risultati ottenuti, la piattaforma dimostra la sua utilità e intuitività nel seguire l’evoluzione del processo di allenamento di un algoritmo di RL, permettendo al progettista di valutarne le prestazioni quantitativamente (tramite i grafici) e qualitativamente (tramite l’ispezione visuale dei singoli episodi). Tra i possibili sviluppi riguardo all’elaborato vi è l’aggiunta di nuovi ambienti di allenamento in cui monitorare il modello e la definizione di un’interfaccia di programmazione standard per permettere ai progettisti di testare modelli e ambienti personalizzati. I recenti risultati pubblicati in letteratura, relativi all’apprendimento mediante Q-Learning [23] e, riguardanti in particolare l’adattamento del modello ai vari ambienti, hanno destato

molto interesse nella comunità scientifica, in quanto le sue proprietà adattive potrebbero, in futuro, gettare le basi di un'intelligenza artificiale capace di adattarsi in modo ottimale ai cambiamenti imprevedibili del mondo reale.

Bibliografia

- [1] Reply, *Intelligenza Artificiale: tutti i trend con la Sonar*,
<https://www.reply.com/it/topics/artificial-intelligence-and-machine-learning/ai-trend-report>, [27/10/2019]
- [2] David Forsyth, Jean Ponce, *Computer Vision A Modern Approach*, Jean Ponce, 2002.
- [3] Isabella Chiari, *Introduzione alla linguistica computazionale*, Bari, Laterza, 2007.
- [4] Mitchell T., *Machine Learning*, McGraw Hill, 1997, p. 2.
- [5] Zaidah Ibrahim, Daliela Rusli, *PREDICTING STUDENTS' ACADEMIC PERFORMANCE: COMPARING ARTIFICIAL NEURAL NETWORK, DECISION TREE AND LINEAR REGRESSION*, 2007.
- [6] Abdulmajid Murad, Jae-Young Pyun, *Deep Recurrent Neural Networks for Human Activity Recognition*, 2017.

- [7] Shubhabrata Datta, Malay K. Banerjee, *Optimizing parameters of supervised learning techniques (ANN) for precise mapping of the input-output relationship in TMCP steels*, 2004.
- [8] Terence D. Sanger, *AN OPTIMALITY PRINCIPLE FOR UNSUPERVISED LEARNING*, <http://papers.nips.cc/paper/139-an-optimality-principle-for-unsupervised-learning.pdf>, [02/11/2019].
- [9] T. Kohonen, *The self-organizing map*, IEEE, 1990.
- [10] G.Joya, M.A.Atencia, F.Sandoval, *Hopfield neural networks for optimization: study of the different dynamics*, 2001.
- [11] Michael Nielsen, *Neural Networks and Deep Learning*, http://neuralnetworksanddeeplearning.com/chap1.html#sigmoid_neurons, [02/11/2019].
- [12] Michael Nielsen, *Neural Networks and Deep Learning*, http://neuralnetworksanddeeplearning.com/chap5.html#the_vanishing_gradient_problem, [02/11/2019].
- [13] Konstantin Eckle, Johannes Schmidt-Hieber, *comparison of deep networks with ReLU activation function and linear spline-type methods*, 2018.

- [14] Michael Nielsen, *Neural Networks and Deep Learning*, http://neuralnetworksanddeeplearning.com/chap1.html#learning_with_gradient_descent, [02/11/2019].
- [15] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, David Silver, *Emergence of Locomotion Behaviours in Rich Environments*, <https://arxiv.org/abs/1707.02286>, [28/10/2019].
- [16] Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, Nicolas Heess, *Learning human behaviors from motion capture by adversarial imitation*, <https://arxiv.org/abs/1707.02201>, [28/10/2019].
- [17] Ziyu Wang, Josh Merel, Scott Reed, Greg Wayne, Nando de Freitas, Nicolas Heess, *Robust Imitation of Diverse Behaviors*, <https://arxiv.org/abs/1707.02747>, [28/11/2019].
- [18] Nicolas Heess, Josh Merel, Ziyu Wang, *Producing flexible behaviours in simulated environments*, <https://deepmind.com/blog/article/producing-flexible-behaviours-simulated-environments>, [28/10/2019].

- [19] Prem Kalra, Nadia Magnenat-Thalmann, Laurent Moccozet, Gael Sannier, Amaury Aubel, Daniel Thalmann, *Real-Time Animation of Realistic Virtual Humans*, <https://pdfs.semanticscholar.org/7a99/604b1d18e4ab8b43f2a27825ba17bd2171e4.pdf> [31/10/2019]
- [20] Tao Xiao, Yun-Feng Fu, "Biomechanical Modeling of Human Body Movement" in *Journal of Biometrics & Biostatistics*, 2016
- [21] Christian Ott, Dongheui Lee, Yoshihiko Nakamura, *Motion capture based human motion recognition and imitation by direct marker control* in Humanoids 2008 - 8th IEEE-RAS International Conference in Humanoid Robots, Daejeon, South Korea, 1-3 Dec. 2008, IEEE, 20 February 2009.
- [22] Richard S. Sutton, Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.
- [23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller, "Playing Atari with Deep Reinforcement Learning". DeepMind Technologies.
- [24] Alexander L. Strehl, Lihong Li, Michael L. Littman, *Reinforcement Learning in Finite MDPs: PAC Analysis*, Sridhar Mahadevan, 2009.

- [25] I. Capuzzo Dolcetta, H. Ishii, “Approximate solutions of the bellman equation of deterministic control theory” in *Applied Mathematics and Optimization*, vol. 11, issue 1, 1984, pp 161-181.
- [26] Christopher JCH Watkins, Peter Dayan, *Q-learning. Machine learning*, 8(3-4):279-292,1992.
- [27] Geoffrey Hinton, *Neural Networks for machine learning*, lezione 6,
<https://www.coursera.org/learn/neural-networks/home/welcome>.
- [28] Michael Fairbank, *The Divergence of Reinforcement Learning Algorithms with Value-Iteration and Function Approximation*, <https://arxiv.org/pdf/1107.4606.pdf>, [04/11/2019].
- [29] A. Shapiro, Y. Wardi, “Convergence analysis of gradient descent stochastic algorithms” in *Journal of Optimization Theory and Applications*, vol. 91, issue 2, pp 439-454.
- [30] Alex Krizhevsky, Ilya Sutskever, Geoff Hinton, “Imagenet classification with deep convolutional neural networks” in *Advances in Neural Information Processing Systems*, 25, pages 1106-1114, 2012.
- [31] Long-Ji Lin, *Reinforcement learning for robots using neural networks*, Technical report, DTICDocument, 1993.

- [32] Michael Wunder, Michael Littman, Monica Babes, *Classes of Multiagent Q-learning Dynamics with ϵ -greedy Exploration*, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.170.2201&rep=rep1&type=pdf>, [06/11/2019].
- [33] John N. Tsitsiklis, Benjamin Van Roy, *An analysis of temporal-difference learning with function approximation*. *Automatic Control, IEEE Transactions on*, 42(5):674-690, 1997.
- [34] Andrew Moore, Chris Atkeson, *Prioritized sweeping: Reinforcement learning with less data and less real time*. *Machine Learning*, 13:103-130, 1993.
- [35] Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen, *Deep content-based music recommendation*, a cura di C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani e K. Q. Weinberger, Curran Associates, Inc., 1 gennaio 2013, pp. 2643-2651.
- [36] Arthur M. Lesk, *Introduction to Bioinformatics*, Oxford University Press, 2019.

Elenco delle figure

2.1	Schema di un percepitrone multistrato feedforward	11
2.2	Funzione di attivazione a gradino	15
2.3	Funzione di attivazione a sigmoide	16
2.4	Funzione di attivazione a ReLU	17
3.1	Andamento metriche di valutazione	30
4.1	Pagina di registrazione.	32
4.2	Pagina di autenticazione.	34
4.3	Pagina per il recupero delle credenziali.	34
4.4	Pagina per l'inserimento della nuova password.	35
4.5	Home del sito quando non sono attivi allenamenti.	36
4.6	Home del sito quando si preme il pulsante start.	38
4.7	Home del sito durante l'allenamento.	39
4.8	Home del sito durante la riproduzione.	40
4.9	Archivio degli allenamenti conclusi.	40

4.10 Schema relazionale del database.	41
5.1 Andamenti della durata (in frame) e dei reward iniziali.	43
5.2 Risultato finale delle prime partite.	43
5.3 Andamenti della durata (in frame) e dei reward dopo le prime 4 ore.	44
5.4 Risultato finale dopo le prime 4 ore.	45
5.5 Andamenti della durata (in frame) e dei reward dopo circa 10 ore.	46
5.6 Risultato finale delle partite dopo circa 10 ore.	46

Ringraziamenti

Desidero ringraziare il professore Simone Palazzo per la disponibilità e l'aiuto fornитоми durante lo sviluppo della presente tesi. Ringrazio inoltre tutti i colleghi, amici e familiari che mi hanno sostenuto e aiutato lungo questo percorso. Un ringraziamento particolare va alla mia fidanzata Deborah per il sostegno e la pazienza dimostratami in questi anni.