UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Tecniche di Intelligenza Artificiale applicate al gioco degli Scacchi

RELATORE

Prof. **Fabio Palomba**

Università degli studi di Salerno

CANDIDATO

Vincenzopio Amendola

Matricola: 0512106935

Anno Accademico 2021-2022

"Non sono mai riuscito a capire gli scacchi. C'è troppa scienza per essere un gioco, e c'è troppo gioco per essere scienza"

- Emanuel Lasker

Sommario

Il contesto applicativo della tesi è incentrato sullo studio di tecniche di Intelligenza Artificiale applicate al gioco degli scacchi, trattate in due diversi moduli. Nel primo modulo è stato considerato il problema della ricerca della mossa migliore nel corso di una partita: è stata dunque programmata da zero una scacchiera sulla quale sono state applicate diverse tecniche di Intelligenza Artificiale di ricerca delle mosse possibili nel contesto di una normale partita. Lo studio sugli algoritmi utilizzati ha infine portato all'analisi dei tempi di esecuzione e al confronto degli stessi. Nel secondo modulo sono state adottate tecniche di Machine Learning per la creazione e l'addestramento di un modello di rete neurale in grado di giocare autonomamente delle partite a scacchi. Sono state infine analizzate le performance sportive del modello attraverso alcune partite, delle quali sono stati forniti gli esiti, contro altri modelli già esistenti.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto applicativo

Lo studio proposto è incentrato sul gioco degli scacchi. Nello specifico vengono passate a rassegna diverse tecniche di Intelligenza Artificiale volte a ricercare (e compiere) una mossa valida sulla scacchiera nel corso di una partita regolare. Vengono resi noti i due differenti approcci adottati per il conseguimento degli obiettivi fissati:

- **Primo approccio**: viene progettata e programmata un'implementazione da zero della scacchiera, sulla quale degli algoritmi di Intelligenza Artificiale valutano lo stato corrente della scacchiera ed effettuano la migliore mossa legale tra quelle disponibili;
- **Secondo approccio**: viene costruita e addestrata una rete neurale che gioca simulando le mosse di un giocatore umano.

1.2 Motivazioni e Obiettivi

Fra i giochi più popolari al mondo, gli scacchi possono essere giocati ovunque (all'aperto, in circolo, online) e la vastità del numero di giocatori è stata tale da favorire lo sviluppo di diverse Federazioni (tra le quali, la più importante, la **Fédération Internationale des Échecs - FIDE**) con conseguenti tornei e competizioni in tutto il mondo. Le motivazioni della stesura del presente elaborato sono da ricercare nella natura intrinseca del gioco stesso.

Claude Shannon, ingegnere e matematico statunitense, nel suo *"Programming a Computer for Playing Chess"*[1] fornì una stima di 10^{120} partite possibili, dimostrando impraticabile l'idea di affrontare il problema della complessità degli scacchi con la forza bruta[1]. Di fronte a tali numeri, non si può non rimanere disorientati e affascinati allo stesso tempo, tenendo anche in considerazione che il numero di atomi nell'universo è stimato intorno a 10^{80}[2]. Gli obiettivi finali sono dunque da ricercare nelle motivazioni stesse; la vera protagonista del presente lavoro di tesi è infatti la **complessità** del gioco degli scacchi, che viene analizzata, studiata e approfondita nei paragrafi seguenti, non senza un'attenta critica e analisi accurata sui risultati raggiunti.

1.3 Risultati

Gli algoritmi di ricerca e di apprendimento sfruttati nei due diversi moduli offrono un quadro completo sulle moderne tecniche di Intelligenza Artificiale che non solo vengono applicate su diverse piattaforme disponibili online ma sono tutt'oggi in continua evoluzione. I risultati ottenuti non vogliono aprire nuovi orizzonti a differenti approcci sullo studio, ma fanno più da panoramica generale a tecniche già esistenti. Come anticipato nel paragrafo 1.1, tali risultati sono stati ottenuti attraverso due diversi moduli di cui si fornisce una breve sintesi, ma che verranno trattati nello specifico nel capitolo 3:

- **Modulo 1**: i classici approcci adottati per la rappresentazione della scacchiera e dei pezzi in gioco (ognuno con le rispettive mosse legali) vengono progettati e riprogrammati da zero per cercare di comprendere meglio la natura degli studi evoluti nel corso del tempo. Attraverso un algoritmo di Intelligenza Artificiale si accede a una lista di mosse possibili (considerando lo stato corrente della scacchiera) e viene giocata la mossa migliore secondo una precisa funzione obiettivo.

- **Modulo 2 - Caissa**: è stato costruito Caissa, un modello di Machine Learning addestrato seguendo un approccio ben preciso. Questo modello è stato poi confrontato con diversi modelli già esistenti al fine di ricavare una valutazione che ne indichi le prestazioni.

Al fine di comprendere al meglio i risultati appena menzionati, è opportuno discutere delle regole che vengono ancora oggi osservate nelle partite disputate a scacchi.

[1] Victor Allis, informatico olandese, anni dopo stimò la complessità essere di almeno 10^{123}, "basata su una media del fattore di ramificazione di 35 e una durata media di gioco di 80 coppie di mosse".

1.4 Come si gioca a scacchi

1.4.1 Preparazione della scacchiera

Le regole ufficiali del gioco degli scacchi[3] prevedono che la scacchiera sia inizialmente disposta in modo che ogni giocatore abbia una casella chiara nell'angolo inferiore destro. Ai due angoli opposti sono collocate le due torri, seguite dai due cavalli e dai due alfieri. La regina occupa sempre la casella del proprio colore, e il re è collocato accanto alla regina. Tutta la seconda traversa è invece occupata dai pedoni.

Figura 1.1: Disposizione dei pezzi all'inizio della partita

1.4.2 Come si muovono i pezzi

Ogni tipologia di pezzo segue delle regole ben precise nei movimenti. Non è possibile, ad esempio, muovere dei pezzi attraversando altri pezzi (eccetto per il cavallo, che può "saltarli") e ogni casella può essere occupata soltanto da un pezzo per volta. Negli scacchi è possibile liberare una casella dall'occupazione avversaria, **catturando** il pezzo interessato ed eliminandolo dalla scacchiera fino alla fine della partita. Come accennato in precedenza, ogni pezzo si muove in modo diverso dall'altro. In particolare:

- il **re** può muoversi soltanto di una casella per volta nelle 8 direzioni disponibili (fig. 1.2), ma non è possibile spostarlo in una casella che lo metterebbe sotto **scacco** (minacciato da un pezzo avversario);

- la **regina** può muoversi in qualsiasi direzione seguendo una linea retta sia in orizzontale che in diagonale di quante caselle vuole (fig. 1.3). Come tutti gli altri pezzi, in caso di cattura di un pezzo avversario il suo movimento si conclude nella casella precedentemente occupata dal pezzo catturato;

- la **torre** può muoversi di quante caselle si desidera, ma soltanto in avanti, indietro e di lato (fig. 1.4). Le due torri, quando ben collegate, si proteggono a vicenda e risultano estremamente utili;

- l'**alfiere** può muoversi in diagonale (fig. 1.5), e in virtù di questa caratteristica ogni alfiere rimane sullo stesso colore fino alla fine della partita (uno sulle caselle chiare e l'altro sulle caselle scure);

- il **cavallo** segue dei movimenti "a L" (fig. 1.6), avanzando di due caselle in una direzione e poi di un'altra casella a 90°;

- il **pedone** può muoversi di una sola casella in avanti (oppure anche due se è la sua prima mossa), ma cattura i pezzi avversari in diagonale di una casella (fig. 1.7). Non è possibile indietreggiare, né catturare all'indietro. Inoltre, se un pedone raggiunge il lato opposto della scacchiera può trasformarsi in qualsiasi altro pezzo.

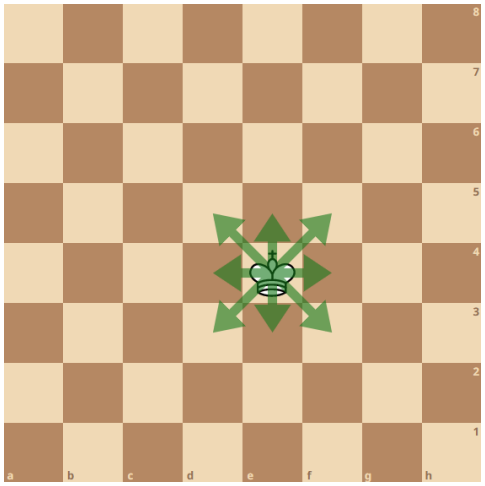

Figura 1.2: Movimenti del re

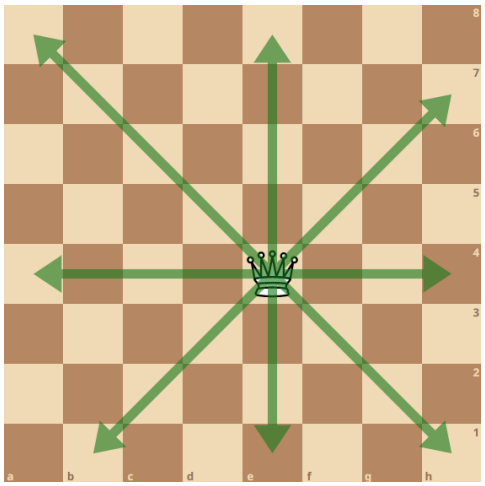

Figura 1.3: Movimenti della regina

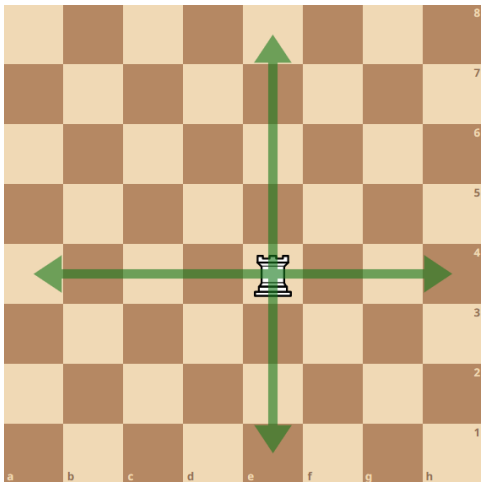

Figura 1.4: Movimenti della torre

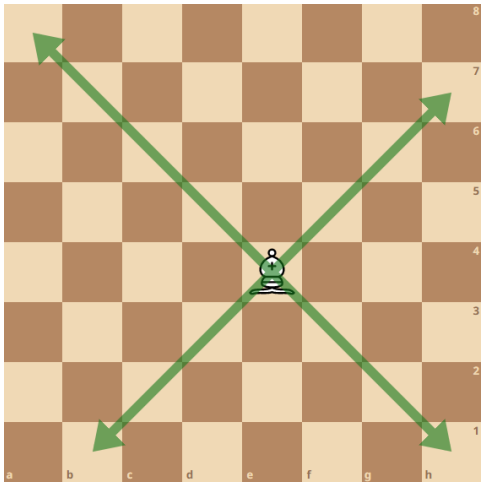

Figura 1.5: Movimenti dell'alfiere

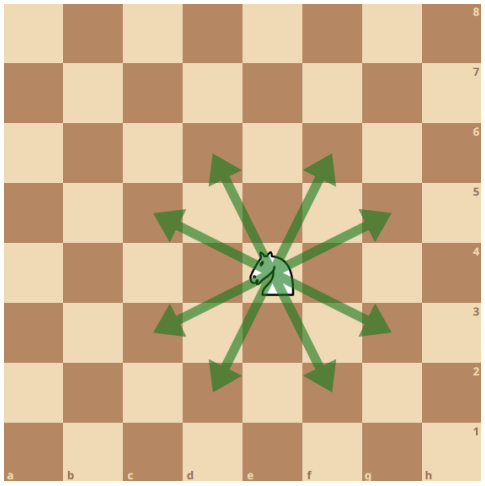

Figura 1.6: Movimenti del cavallo

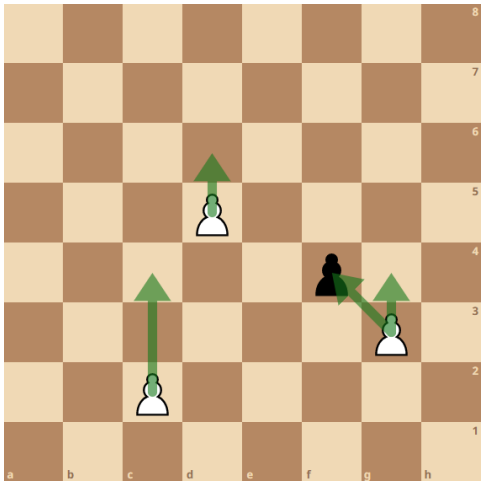

Figura 1.7: Movimenti del pedone

Negli scacchi è possibile effettuare l'**arrocco**: si tratta di una mossa speciale che consente non solo di mettere il proprio re al sicuro, ma anche di sviluppare una torre portandola in gioco. Per effettuare l'arrocco si sposta il re di due caselle verso destra o sinistra e si sposta la torre di una casella accanto al re, ma in direzione opposta. Tuttavia, non è possibile effettuare l'arrocco se il re o la torre in questione siano già stati mossi in un turno precedente, o se le caselle interessate nell'arrocco siano occupate da altri pezzi o controllate da pezzi avversari.

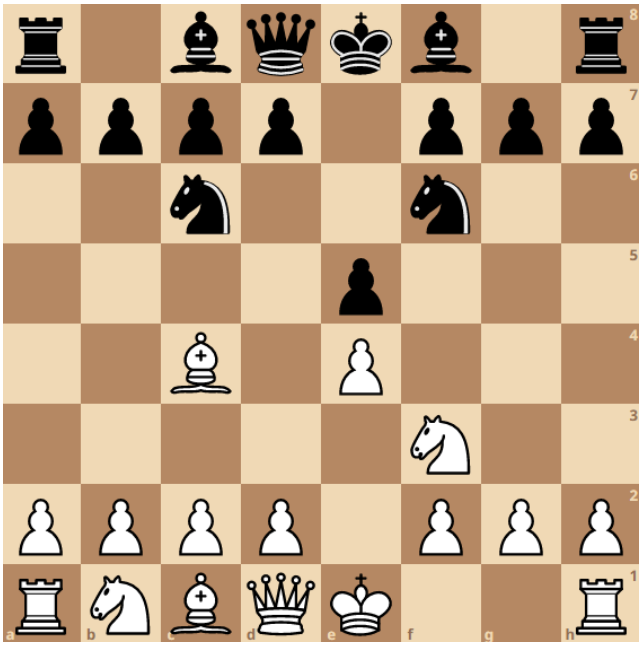

Figura 1.8: Preparazione all'arrocco

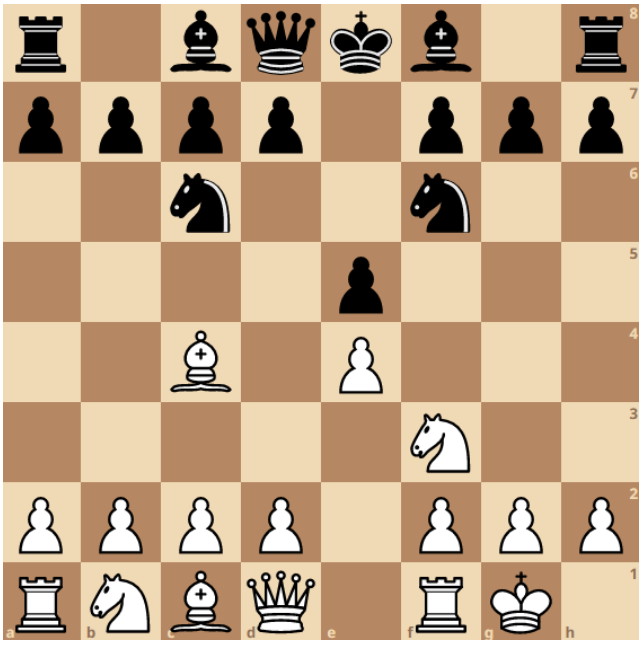

Figura 1.9: Posizione dopo l'arrocco

Un'altra regola speciale degli scacchi riguarda la cattura **en passant** dei pedoni: se un pedone muove di due caselle alla sua prima mossa portandosi accanto a un pedone avversario (saltando la casella su cui sarebbe stato catturato), l'avversario ha la possibilità di catturare quel pedone occupando la casella aventi in diagonale. Questa speciale cattura deve però essere effettuata nel turno immediatamente successivo all'avanzamento del pedone, altrimenti non sarà più possibile farlo.

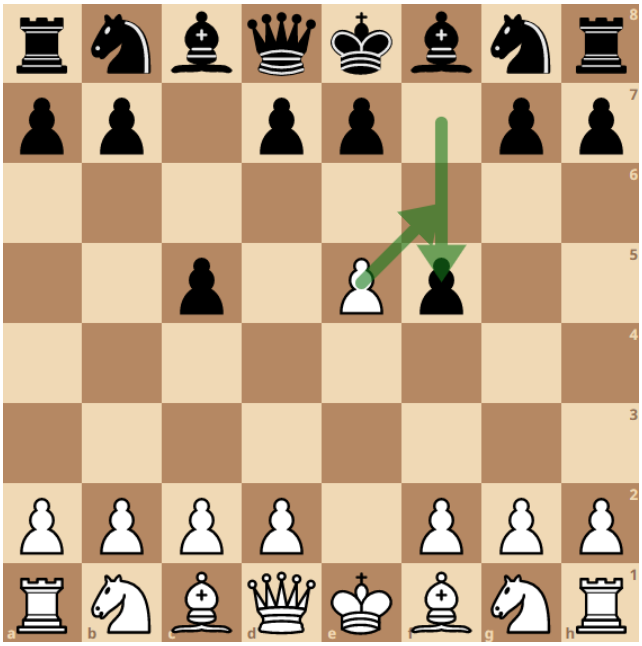

Figura 1.10: Posizione prima della cattura en passant

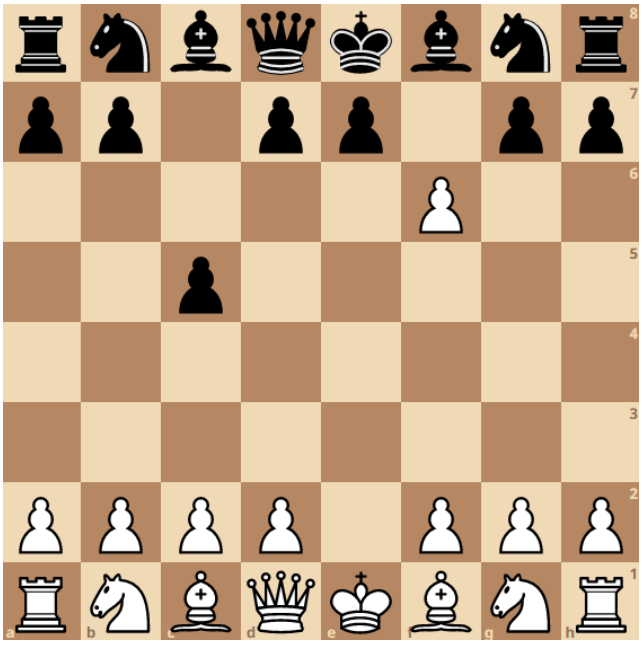

Figura 1.11: Cattura en passant

Nel gioco degli scacchi, è **sempre** il giocatore con i pezzi bianchi a muovere per primo. Questo privilegio porta un leggero vantaggio al bianco, che nella maggior parte dei casi imposterà piani d'attacco e di difesa già dalla prima mossa.

1.4.3 Vincere una partita a scacchi

L'obiettivo del gioco degli scacchi è quello di attaccare il re avversario, cercando di impedirgli di sottrarsi alla minaccia. Esistono, infatti, tre modi per difendere il proprio re da uno scacco: spostandolo in una casella libera e non minacciata, interponendo un proprio pezzo oppure catturando il pezzo che minaccia il re. Se il re non può liberarsi dallo scacco in nessuno dei tre modi, la partita viene dichiarata finita per **scacco matto**. Esistono casi in cui una partita di scacchi si conclude con una **patta**, cioè in parità, senza vincitori. Ciò accade principalmente per 5 motivi:

- tutti i propri pezzi sono stati catturati, tranne il re. Non è possibile eseguire alcuna mossa legale, ma contemporaneamente il re non è sotto scacco. In questo caso, la partita finisce per **stallo**:
- i giocatori possono semplicemente accordarsi e smettere di giocare;
- non è possibile dare scacco matto in alcun modo (per esempio, un re e un alfiere contro un re);
- è stata ripetuta la stessa posizione per la terza volta nel corso della partita;
- sono state giocate 50 mosse consecutive senza catturare alcun pezzo.

Cercare di proteggere i propri pezzi è uno dei punti chiave per vincere la partita. Di solito i giocatori di scacchi assegnano a ciascun pezzo un numero che ne indichi il valore, in modo da portare avanti delle strategie vincenti. Il **re** ha il valore più alto, poiché la sua perdita causerebbe la sconfitta. La **regina** vale 9 punti, essendo il pezzo più importante dopo il re data la sua elevata libertà di movimento che potrebbe causare problemi all'avversario. Una **torre** vanta ben 5 punti, e due torri usate in sinergia sarebbero più potenti di un'unica regina. All'**alfiere** e al **cavallo** vengono assegnati valori pressoché simili, intorno ai 3 punti. Il **pedone** vale 1 punto. Sebbene esistano dei valori convenzionali comunemente assegnati, è bene tenere a mente che il valore corretto di ciascun pezzo dipende in realtà dalle sue capacità considerate in precisi momenti di una partita in corso[4].

1.5 Struttura della tesi

La trattazione del lavoro di tesi è strutturata secondo il seguente elenco:

- **Introduzione**: viene fornita una panoramica dello studio effettuato, con particolare attenzione alle motivazioni, agli obiettivi e ai risultati del lavoro svolto;
- **Ruolo del gioco degli scacchi nell'Intelligenza Artificiale**: l'attenzione viene spostata sulla storia che ha portato il gioco degli scacchi ad essere trattato con le tecniche moderne e sull'analisi di alcune delle tecnologie attualmente in uso;
- **Progettazione e implementazione**: si esaminano nel dettaglio la progettazione e l'implementazione del lavoro svolto;
- **Conclusioni e sviluppi futuri**: vengono presentate riflessioni e considerazioni di carattere generale con eventuali riferimenti agli sviluppi futuri.

CAPITOLO 2

Ruolo del gioco degli scacchi nell'Intelligenza Artificiale

Questo capitolo illustra lo stato dell'arte e i lavori presenti in letteratura sugli aspetti di ricerca trattati nel nostro studio. Prima di dare un'occhiata alla varietà degli approcci adottati per affrontare il problema della complessità degli scacchi, è opportuno fare un passo indietro per capire meglio di *che cosa* si parla.

2.1 Teoria dei giochi

La **teoria dei giochi**[5] è una branca della matematica che studia l'interazione strategica tra gli individui nell'ambito di un **gioco**. In questo senso viene considerato gioco uno scenario in cui i contendenti (giocatori) seguono un determinato tipo di comportamento (strategia) al fine di massimizzare la propria vincita. Nel complesso l'esito finale del gioco coincide con il risultato della sequenza delle strategie adottate dai giocatori e dai loro avversari. Chiarito lo scopo e il significato dei giochi, questi vengono divisi in 5 gruppi principali:

- **giochi cooperativi e non cooperativi**: nei giochi cooperativi i giocatori possono formare alleanze per massimizzare le probabilità di vincita. Ciò non è possibile nei giochi non cooperativi;
- **giochi simmetrici e asimmetrici**: nei giochi simmetrici tutti i partecipanti hanno gli stessi obiettivi, ed è fondamentale elaborare strategie vincenti. Nei giochi asimmetrici, invece, i giocatori hanno obiettivi diversi;

- **giochi ad informazione perfetta e ad informazione imperfetta**: nei giochi ad informazione perfetta tutti i giocatori possono vedere le mosse degli altri giocatori. Le mosse sono invece nascoste nei giochi ad informazione imperfetta;
- **giochi simultanei e sequenziali**: nei giochi simultanei i giocatori possono giocare le proprie mosse simultaneamente, piuttosto che a turni come avviene nei giochi sequenziali;
- **giochi a somma zero e a somma non zero**: nei giochi a somma zero la vincita di un giocatore è equilibrata dalla perdita di un altro. Nei giochi a somma non zero, invece, più giocatori possono trarre vantaggio dalle vincite di un altro.

Il gioco degli scacchi è pertanto considerato un gioco non cooperativo e simmetrico, poiché le partite sono solitamente disputate tra due singoli avversari che puntano entrambi a vincere la partita. Ciascun giocatore è tenuto a muovere i propri pezzi a turno davanti agli occhi dell'avversario, e alla fine della partita la vincita di un giocatore comporta la perdita dell'altro: queste caratteristiche rendono gli scacchi un gioco ad informazione perfetta, sequenziale e a somma zero.

2.1.1 Teoria dei giochi e Intelligenza Artificiale

I giochi ritenuti più interessanti nell'ambito dell'Intelligenza Artificiale sono quelli **a somma zero** con **informazione perfetta**, solitamente a turni e a due giocatori. Le azioni degli agenti sono pertanto sequenziali, e le loro funzioni di utilità alla fine della partita sono uguali ma di segno opposto (vittoria +1, sconfitta -1). I primi approcci dell'Intelligenza Artificiale allo studio dei giochi si ebbero già nel 1950 proprio con gli scacchi. Da quel punto in poi le macchine hanno pian piano superato le capacità degli esseri umani anche nella dama e in molti altri giochi[6]. L'interesse verso i giochi è dovuto alla **facilità** nella loro rappresentazione contrapposta alla **difficoltà** nel risolverli[7]. Un gioco può infatti essere definito come un **problema di ricerca** avente le seguenti caratteristiche:

- lo **stato iniziale**, che specifica la configurazione di partenza del gioco stesso;
- il **giocatore** a cui tocca fare una mossa in un dato stato;
- l'**insieme delle mosse** lecite in un determinato stato;
- il **modello di transizione**, cioè il risultato di una mossa;
- il **test di terminazione** che controlla se la partita è finita oppure no;

- la **funzione obiettivo** che definisce il valore numerico finale per un gioco che termina in un particolare stato per un particolare giocatore.

L'**obiettivo** principale nello studio di questi problemi è dunque la ricerca di una strategia da adottare considerando le mosse dell'avversario; più in particolare si ricerca una strategia **ottima** che porti ad un risultato almeno pari a quello di qualsiasi altra strategia, assumendo che si stia giocando contro un giocatore infallibile. Tale ricerca si traduce nella visita dell'**albero di gioco**, ossia un albero in cui i nodi sono gli stati del gioco e gli archi le mosse. La visita dell'albero di gioco degli scacchi è rimandata al Capitolo 3.

2.2 Breve panoramica sui motori scacchistici

2.2.1 Turochamp

Come accennato nel capitolo 1, la complessità degli scacchi è ciò che li rende una delle sfide più interessanti dell'Intelligenza Artificiale. Alan Turing, uno dei padri dell'informatica, fu tra i primi ad interessarsi in maniera concreta al problema, progettando *Turochamp*[1], un algoritmo per giocare a scacchi[8]. La sua debolezza fu però evidenziata da Garri Kasparov in una conferenza del 2012, che mostrò che l'algoritmo era in grado di valutare un numero molto limitato di varianti[9]. Il contributo di *Turochamp*, in ogni caso, gettò delle importanti basi per l'evoluzione delle moderne tecniche di ricerca minimax.

2.2.2 Deep Blue

Fu solo nel 1996 che si cominciarono a temere le enormi potenzialità dei computer in ambito scacchistico: in quell'anno fu disputata una partita in condizioni normali di torneo tra Kasparov (allora campione del mondo) e *Deep Blue*, un computer progettato da IBM appositamente per giocare a scacchi. La partita si concluse con l'abbandono da parte del campione dopo 40 mosse. Nel 1997, in occasione della rivincita, Kasparov abbandonò dopo sole 19 mosse[2]. Questi eventi aprirono le basi ai moderni motori scacchistici, che nel corso

[1] Ideato nel 1948, *Turochamp* nacque ben prima di un calcolatore che fosse in grado di leggere ed eseguire il programma. Ciò portò lo stesso Turing a valutare la "bontà" dell'algoritmo, analizzando le mosse con carta e penna.

[2] Alcune mosse di *Deep Blue* risultarono a Kasparov piuttosto creative e incomprensibili, al punto da sospettare che la macchina stesse ricevendo un supporto umano nel corso della partita. Effettivamente il codice del programma fu modificato tra una sfida e l'altra, permettendo alla macchina di non cadere nelle trappole del campione nelle mosse finali della partita.

degli anni hanno dimostrato di saper tener testa anche ai migliori giocatori di scacchi[10]. La potenza computazionale di *Deep Blue* era dovuta al **parallelismo massivo**: furono utilizzati ben 480 processori (progettati per il gioco degli scacchi), che eseguivano un algoritmo scritto in linguaggio C in grado di calcolare 200 milioni di posizioni al secondo. Le funzioni di valutazione erano scritte in forma generale, mentre la lista delle aperture fu fornita dai campioni Illescas, Fedorowicz e De Firmian.

2.2.3 Stockfish

Stockfish è un motore scacchistico open-source considerato uno dei più forti in assoluto. Correntemente il motore implementa una rete neurale con *deep learning*. La scacchiera viene rappresentata tramite **bitboard** (una struttura dati in grado di immagazzinare lo stato di ogni casella della scacchiera all'interno di una parola di 64 bit) e la computazione viene supportata da 512 thread. L'algoritmo di ricerca ad albero è implementato con **potatura alfa-beta**; in tal modo il numero di nodi da ricercare viene drasticamente ridotto sulla base della funzione di valutazione (la valutazione di una mossa termina non appena viene dimostrato che è peggiore di una mossa già valutata in precedenza). Inoltre, mosse come catture vincenti sono valutate prima di altre mosse, riducendo ulteriormente la profondità dell'albero da visitare e garantendo una maggior efficienza dell'algoritmo di ricerca[11].

2.2.4 AlphaZero

L'evoluzione continua delle tecniche di **Machine Learning** portò il team di Google DeepMind alla realizzazione di un algoritmo basato su tecniche di apprendimento automatico. *AlphaZero* è infatti guidato da una rete neurale convoluzionale addestrata per rinforzo. In questo modo la qualità di una mossa è valutata da un valore numerico di "ricompensa", incoraggiando il motore ad effettuare scelte simili in futuro (viceversa, un valore numerico di "punizione" allontana il motore a considerare determinate mosse). *AlphaZero* fu addestrato per 9 ore su una singola macchina munita di 4 TPU[3] giocando contro *Stockfish 8*. Effettivamente la natura dell'algoritmo consentì ad *AlphaZero* di giocare senza libro di apertura e chiusura, arrivando a scoprire e giocare autonomamente diverse aperture che a *Stockfish* vennero invece fornite manualmente[12].

[3]Una **T**ensor **P**rocessing **U**nit è un microprocessore per applicazioni specifiche nel campo delle reti neurali.

2.3 Considerazioni sui motori scacchistici analizzati

Il fascino ludico e allo stesso tempo scientifico del gioco degli scacchi ha sin da subito catturato l'attenzione non solo degli appassionati sportivi ma anche di diverse personalità rilevanti in campo informatico. La nascita dei primi motori scacchistici ha favorito la creazione di diverse piattaforme online, tra le quali spiccano **Chess.com** e **Lichess**, che offrono, tra le diverse funzionalità, la possibilità di sostenere partite di scacchi contro motori scacchistici di tutto rispetto. L'incessante studio del gioco degli scacchi continua ancora oggi a proporre nuove idee, nuovi motori scacchistici, contribuendo in modo significativo ai miglioramenti di quelli invece già esistenti[13]. I due moduli sviluppati nel presente lavoro di tesi si avvicinano ad un background già esistente in maniera semplice e basilare. Si avverte, da un lato, la volontà di sviluppare una piattaforma semplice, pratica e snellita, differente dalle già esistenti, che offra unicamente la possibilità di giocare partite contro un'Intelligenza Artificiale di cui viene fornita un'analisi delle tecniche adottate per la ricerca delle mosse; dall'altro, si intende sviluppare un motore scacchistico con le tecniche essenziali, la cui semplicità lo distingue in maniera netta dai suoi simili, e analizzarne le performance attraverso partite competitive contro altri motori per comprendere quanto sia fondamentale l'approccio tecnico iniziale per la realizzazione dei più noti motori scacchistici.

CAPITOLO 3

Progettazione e Implementazione

Vero cuore della tesi, in questo capitolo vengono analizzate nel dettaglio la progettazione e l'implementazione dei due diversi moduli sviluppati. Pur condividendo le stesse finalità, i due moduli implementati presentano caratteristiche ben distinte tra loro. La scelta di tale distinzione deriva in primis dalla volontà di cimentarsi nello sviluppo di un'applicazione che consenta a un giocatore di fronteggiare un'Intelligenza Artificiale più o meno competitiva in una partita. In secundis, l'analisi delle prestazioni derivante dalla suddetta applicazione avrebbe poi spinto alla creazione di un modello di Machine Learning e al confronto con diversi motori scacchistici, al fine di fornire un'analisi delle prestazioni sempre più accurata e approfondita.

3.1 Primo Modulo

L'applicazione è stata realizzata in linguaggio Python. L'intenzione è stata quella di sviluppare un'applicazione utilizzabile da un utente umano per giocare una partita contro un'Intelligenza Artificiale: si è dunque ritenuta opportuna la progettazione di un'interfaccia grafica per l'interazione uomo-macchina. Attraverso la libreria *pygame* è stato possibile creare l'interfaccia in modo programmatico, rappresentando le caselle con array di array di dimensione 8x8 e sfruttando semplici metodi per caricare le immagini dei pezzi precedentemente salvate nella cartella di lavoro. Inoltre, assieme ai pezzi, sono state distinte in maniera visivamente chiara e precisa le caselle chiare da quelle scure.

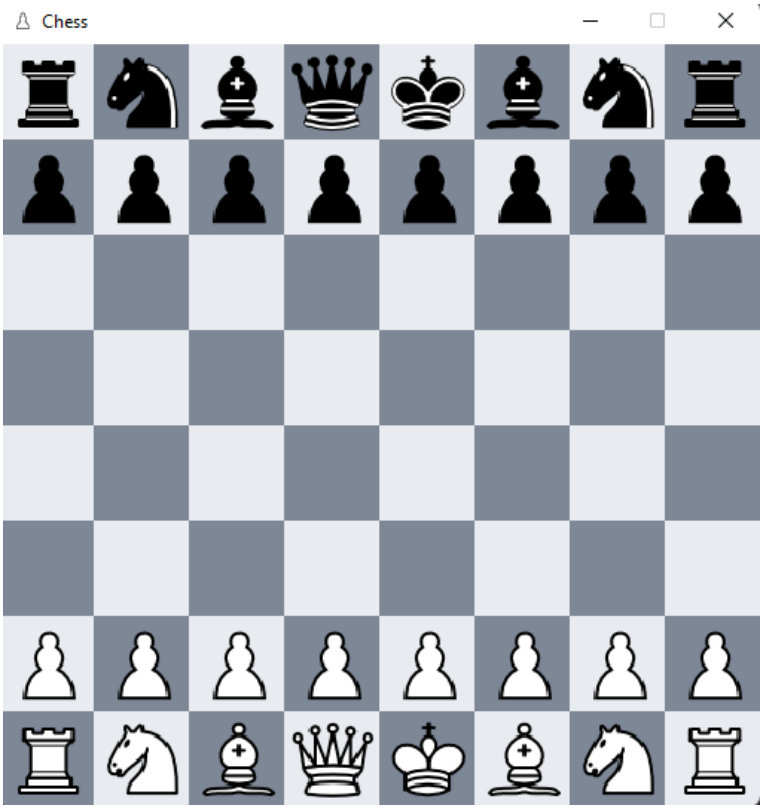

Figura 3.1: Interfaccia grafica vista dall'utente

A seguito della rappresentazione iniziale della scacchiera sono stati programmati i metodi che si occupano della generazione delle mosse possibili per ciascun pezzo, per impedire all'utente di effettuare mosse non legali[1]. Analogamente, ad ogni mossa giocata viene effettuato un controllo nel caso in cui il re sia sotto scacco o nel caso in cui vi sia una situazione di scacco matto, dichiarando partita finita nel secondo caso.

[1]Una mossa non legale è una mossa che pone il re sotto scacco oppure che non libera il re da uno scacco subito nella mossa precedente.

Figura 3.2: Partita vinta dal bianco

3.1.1 Ricerca delle mosse legali

Tutta la realizzazione di quanto citato finora è stata effettuata in modo conforme alle tecniche di Intelligenza Artificiale che sono state utilizzate per la generazione delle mosse. L'obiettivo è stato quello di riprogrammare algoritmi di ricerca applicati a una rappresentazione della scacchiera che fosse familiare al programmatore, per comprenderne a fondo il funzionamento.

Algoritmo naive

Una prima, semplice soluzione al problema è stata provata mediante l'utilizzo di un algoritmo naive di ricerca delle mosse. Consideriamo il solito caso in cui il giocatore muove i pezzi bianchi e l'Intelligenza Artificiale i pezzi neri. Dopo una qualsiasi mossa di apertura da parte del bianco, il nero ha un numero molto ristretto di opzioni tra cui scegliere: spostare uno dei pedoni in avanti di una o due caselle, o spostare un cavallo in una delle due caselle legali a inizio partita. Sfruttando un metodo che ritorna la lista di mosse legali data in input una specifica situazione sulla scacchiera, l'algoritmo naive pesca a caso una delle mosse tra

quelle disponibili. Chiaramente, un simile approccio sarebbe difficilmente preso in seria considerazione nello sviluppo di un'applicazione del genere, preferendo una soluzione che tenga conto di diversi parametri (mosse storicamente giuste, valore dei pezzi, minacce possibili ai pezzi dell'avversario, trappole in apertura...). Per tale motivo, si è ritenuto necessario lo studio e l'approfondimento di altre metodologie.

Algoritmo greedy

Come specificato nel paragrafo 1.4.3, nel gioco degli scacchi è fondamentale tenere in considerazione non solo lo stato corrente della scacchiera, ma anche il valore di ciascun pezzo (di solito, un pedone vale molto meno della regina). Sebbene sia vero che talvolta è lo stato stesso della scacchiera a determinare il valore dei pezzi (in alcune situazioni può essere utile sacrificare un pezzo di valore alto per vincere la partita), per convenzione si è deciso di assegnare a ciascun pezzo un valore intero di default. Questi valori sono sfruttati in un metodo di valutazione dello stato corrente della scacchiera, che indica quanto la situazione in corso sia più o meno favorevole per la vittoria.

Più preciso e affidabile dell'algoritmo precedentemente considerato, l'algoritmo greedy tende più a mettere in difficoltà l'avversario piuttosto che a vincere la partita, spesso in maniera controproducente. La priorità di questa strategia è infatti quella di catturare i pezzi dell'avversario, anche a costo di mettere a rischio i propri. In ogni caso, è interessante notare come, tenendo in considerazione il valore dei pezzi, quest'algoritmo favorisca la cattura di pezzi "pesanti" piuttosto che di altri.

Algoritmo minimax

La strategia minimax è una soluzione studiata appositamente per i giochi a somma zero, come nel caso degli scacchi. Lo scopo, come suggerisce il nome, è quello di **minimizzare la massima perdita** possibile, analizzando l'albero delle decisioni e valutando l'utilità di un nodo di trovarsi nello stato corrente (a patto che entrambi i giocatori scelgano mosse ottime fino alla fine della partita). A differenza dei due algoritmi precedentemente analizzati, dunque, salta subito all'occhio la prima peculiarità dell'algoritmo, che tiene in stretta considerazione le mosse dell'avversario e gioca comportandosi di conseguenza. Attraverso l'algoritmo minimax il problema di ricerca della mosse migliore è stato ridotto al problema della visita dell'albero delle decisioni. Analizzando la complessità intrinseca del gioco degli scacchi trattata nei capitoli precedenti si è ritenuto necessario effettuare una leggera modifica all'algoritmo

minimax che tenesse in considerazione la profondità con cui scendere nell'albero, portando il problema a livelli più o meno trattabili in termini di complessità temporale (l'analisi nel dettaglio delle prestazioni è rimandata al paragrafo 3.1.2).

Algoritmo negamax con potatura alfa-beta

L'algoritmo negamax costituisce una variante dell'algoritmo minimax. La caratteristica principale dell'algoritmo è quella di considerare il valore del giocatore A in un certo gioco come la negazione del valore del giocatore B. Così, il giocatore che deve muovere cercherà una mossa che massimizzi la negazione del valore della posizione risultante dalla mossa: così facendo, basta un singolo calcolo per ricavare il valore di tutte le posizioni. Questa è chiaramente una semplificazione rispetto al minimax. Un ulteriore miglioramento all'algoritmo è stato apportato considerando la **potatura alfa-beta**. L'idea è in realtà applicabile a qualsiasi albero, e consente di ridurre significativamente la complessità di tempo necessaria all'esplorazione. Consideriamo un qualsiasi nodo n raggiungibile: se esiste un'alternativa m migliore, allora n e tutti i suoi discendenti non saranno mai esplorati, portando a una vera e propria "potatura" dell'albero. L'efficacia dell'algoritmo, in ogni caso, dipendente dall'ordinamento dei nodi visitati, che dipende da gioco a gioco. Nel caso degli scacchi sarebbe una buona idea considerare dapprima le mosse che catturano i pezzi avversari, poi le mosse che li minacciano e infine le altre.

3.1.2 Analisi delle prestazioni

Non è stata fornita un'analisi delle prestazioni per tutti gli algoritmi discussi finora. Le forti limitazioni di alcune delle strategie adottate per risolvere il problema della ricerca di una mossa non hanno favorito un interesse significativo: la "casualità" a cui si affida l'algoritmo naive e la bassa considerazione delle mosse dell'avversario dell'algoritmo greedy non consentirebbero un'indagine interessante. Al contrario, data la natura della strategia minimax che si serve di un vero e proprio albero di ricerca in cui ciascun nodo rappresenta una mossa legale, si è ritenuto opportuno procedere all'analisi prestazionale del suddetto algoritmo. Per sua natura, l'algoritmo minimax non è efficiente in termini di complessità temporale: essendo essa governata dalla dimensione m dell'albero, nel caso peggiore la ricerca avrà complessità $O(b^m)$, con tempi di esecuzione relativamente elevati.

Profondità	Prima Mossa	Seconda Mossa	Terza Mossa
1	0,26	0,56	0,94
2	1,39	3,02	4,02
3	41,74	82,96	192,34

Tabella 3.1: Tempo di visita dell'albero (in secondi) al variare della profondità (senza potatura)

Tempi del genere rendono il problema intrattabile. Si notano sin da subito, a una profondità non eccessivamente elevata, i tempi d'attesa per la ricerca dopo appena la terza mossa, che in una partita di circa venti o trenta mosse sarebbero notevolmente elevati. Per ovviare al problema è stata aggiunta la tecnica della **potatura**, che consente di evitare di visitare tutti i nodi dell'albero, alla variante negamax. I risultati delle prestazioni dell'algoritmo negamax con potatura alfa-beta sono forniti di seguito.

Profondità	Prima Mossa	Seconda Mossa	Terza Mossa
1	0,15	0,19	0,35
2	0,86	1,32	2,85
3	11,74	17,77	59,81

Tabella 3.2: Tempo di visita dell'albero (in secondi) al variare della profondità (con potatura)

La complessità esponenziale dell'algoritmo permane, ma si può notare dai dati un andamento sostanzialmente "meno esponenziale" dell'algoritmo precedentemente adottato, con tempi di esecuzione conseguenzialmente ridotti.

3.1.3 Miglioramenti della funzione di valutazione

A seguito dell'analisi degli algoritmi sono stati effettuati degli esperimenti sul funzionamento delle tecniche di Intelligenza Artificiale apportando modifiche al metodo di valutazione della posizione sulla scacchiera. Come indicato nel paragrafo dedicato all'algoritmo greedy era stato originariamente considerato il valore di ciascun pezzo, ottenendo un valore più alto per un pezzo come la Regina piuttosto che per un pedone. Questo paragrafo è invece dedicato alla descrizione di una nuova funzione di valutazione, che tiene traccia non solo del valore di un pezzo, ma anche dell'utilità della sua posizione sulla scacchiera.

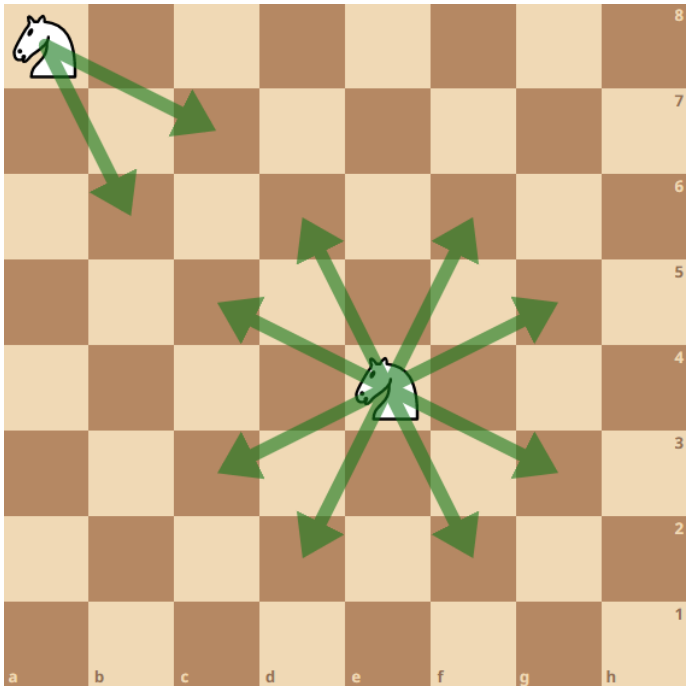

Figura 3.3: Utilità delle caselle per il cavallo

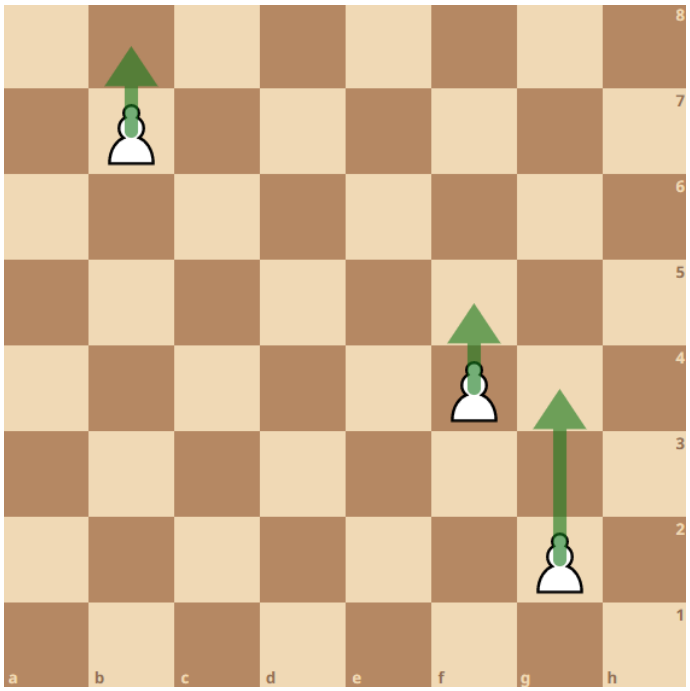

Figura 3.4: Utilità delle caselle per il pedone

La **figura 3.3** indica l'utilità del cavallo di trovarsi in una certa posizione sulla scacchiera: un cavallo mantenuto verso il centro della scacchiera è chiaramente più forte di un cavallo mantenuto verso i bordi, essendo in quest'ultimo caso più limitato nei movimenti. Analogamente, le posizioni considerate nella **figura 3.4** suggeriscono un'idea simile: per un pedone è

importante non solo controllare le caselle in diagonale, ma anche cercare di arrivare al lato opposto della scacchiera (al quale è assegnato il valore più alto) per arrivare a promozione.

Cambiamenti nelle prestazioni

Sebbene sia stata apportata una modifica alla funzione di valutazione, la complessità dell'algoritmo di ricerca non è variata in modo significativo. Il motivo è da ricercare, come già spiegato nel paragrafo dedicato, alla natura intrinseca dell'algoritmo. I continui miglioramenti apportati alla funzione di valutazione non sarebbero abbastanza per ridurre i tempi di visita dell'albero delle decisioni. Una riduzione notevole dei tempi di esecuzione era stata già ottenuta attraverso la tecnica della potatura alfa-beta.

3.2 Secondo Modulo - Caissa

La modifica della funzione di valutazione del primo modulo gettava le basi per la ricerca di una metodologia completamente nuova, tenendo sempre in considerazione le motivazioni e gli obiettivi del lavoro in corso. La volontà di raccogliere i risultati del modulo precedentemente discusso e di evitare di fornire nuovi algoritmi la cui discussione sarebbe stata ridondante si è dunque tradotta nell'intenzione di trattare il problema del gioco degli scacchi da un approccio differente. In questa seconda parte del capitolo si esaminano tecniche di **Machine Learning**, che sono state sperimentate in un modulo tutto nuovo. I risultati si sono tradotti nella creazione di **Caissa**, un motore scacchistico in grado di comprendere lo stato della scacchiera e giocare di conseguenza.

3.2.1 Machine Learning e reti neurali artificiali

La teoria dell'apprendimento è quella branca dell'Intelligenza Artificiale che si occupa, per l'appunto, dell'apprendimento degli agenti intelligenti, permettendogli di operare in ambienti inizialmente sconosciuti in maniera sempre più appropriata. Tale miglioramento viene effettuato attraverso l'operato stesso dell'agente, dunque in base all'esperienza, ma ciò non preclude necessariamente all'agente di imparare per mezzo di dati che gli sono stati forniti in input. Uno degli aspetti maggiormente considerati nel contesto analizzato nel lavoro di tesi è quello delle **reti neurali**: si tratta di un modello computazionale organizzato in diversi strati, ognuno dei quali scambia le informazioni con quello successivo per garantire un'elaborazione dell'informazione sempre più accurata. L'idea di base è che ogni rete neurale sia composta da singole unità chiamate **neuroni**, tutte diversamente collegate tra loro in

diversi livelli. Ogni neurone svolge uno specifico task, ma l'interconnessione di diversi neuroni garantisce l'esecuzione di task più complessi.

3.2.2 Come funziona una rete neurale?

Il funzionamento di una rete neurale artificiale è strettamente collegato alle reti neurali del cervello umano, i cui neuroni formano una rete complessa e interconnessa favorendo l'elaborazione delle informazioni. I neuroni artificiali che costruiscono una rete neurale artificiale funzionano allo stesso modo, formando una rete che di base possiede tre livelli:

- **Livello di Input**: i neuroni si occupano della ricezione dei dati dell'ambiente esterno, che vengono trasferiti al livello successivo dopo essere stati elaborati e categorizzati;
- **Livello nascosto**: i neuroni appartenenti a questo livello analizzano l'output del livello precedente, e continuano ad elaborare l'informazione anche trasferendole ad altri livelli nascosti;
- **Livello di Output**: restituisce il risultato finale dell'intera elaborazione.

Nel caso di milioni di neuroni artificiali connessi su molteplici livelli nascosti si parla di reti di **deep learning**. In questo contesto alle connessioni tra un nodo e l'altro viene assegnato un **peso**, cioè un numero che indica quando un nodo sia stimolato oppure soppresso. Le reti neurali profonde necessitano di milioni di esempi dati in input proprio per compensare l'elevato numero di livelli nascosti tra loro. Proprio per questo motivo l'addestramento di Caissa è stato effettuato sulla base di milioni di partite di scacchi già giocate (3.2.3). Inoltre, data la tipologia di dati da elaborare nel gioco degli scacchi è stato deciso di modellare una rete neurale **convoluzionale** (Convolutional Neural Network, **CNN** d'ora in poi), essendo queste particolarmente adatte per l'elaborazione di dati di tipo 2D. Non solo, le CNN si sono spesso dimostrate adatte per lo studio di problemi dalla complessità simile a quella degli scacchi, ad esempio nel gioco del Go[14]. Una corretta modellazione dei dati porterebbe dunque le CNN a giocare in modo competitivo contro gli esseri umani.

3.2.3 Preparazioni preliminari alla fase di addestramento

Per lo sviluppo del modulo è stata abbandonata l'implementazione precedentemente programmata, favorendo l'utilizzo della API *python chess*: le esigenze hanno infatti portato alla ricerca di un'implementazione che fosse stabile, completa, e continuamente supportata e manutenuta dagli sviluppatori. Inoltre, è stata utilizzata la piattaforma **Jupyter Notebook**,

utile al programmatore per visionare in maniera chiara e user friendly i pezzi sulla scacchiera. Attraverso l'ausilio della piattaforma **Colab** di Google è stato recuperato un dataset da dare in input al modello di rete neurale per l'addestramento, contenente milioni di partite di scacchi già giocate. L'obiettivo è stato infatti quello di valutare partite note e addestrare il modello a giocare in modo competitivo regolandosi di conseguenza. La valutazione delle posizioni delle partite fornite in input è stata effettuata mediante il motore *Stockfish 15* che, sulla base di diversi parametri come ad esempio il vantaggio materiale, la sicurezza del proprio re, la coordinazione dei propri pezzi e diversi altri fattori calcolati dalla funzione obiettivo, fornisce in output un intero positivo (vantaggio del bianco) o negativo (vantaggio del nero).

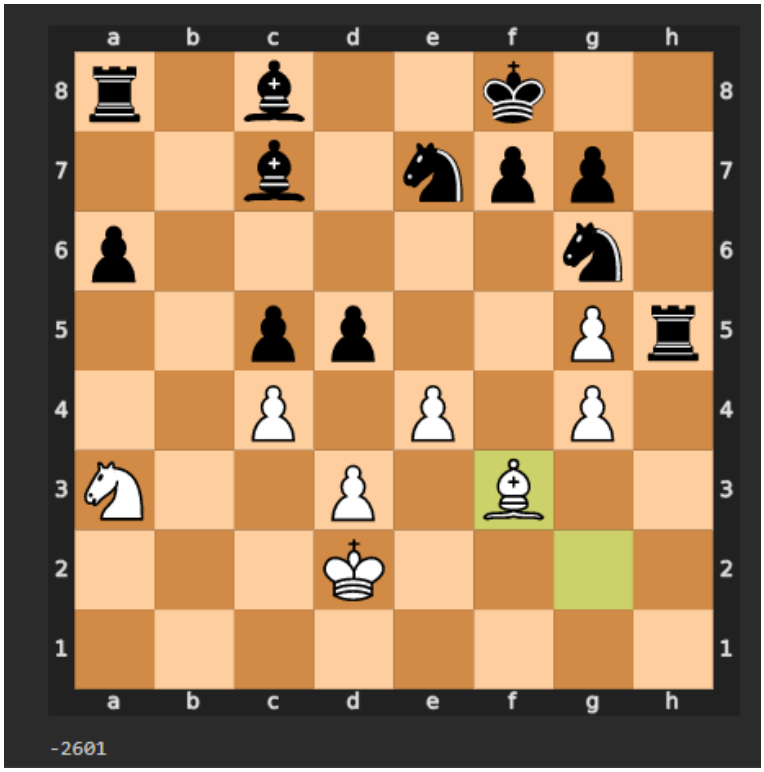

Figura 3.5: Vantaggio materiale e posizionale del nero con rispettivo punteggio

La rappresentazione originale della scacchiera implementata dalla API utilizzata è stata rielaborata per il problema in questione, rendendola più adatta alla CNN in esame. Questa nuova configurazione è strettamente collegata ai neuroni del modello, e permette lo scambio di informazioni sui tipi di pezzi che sono minacciati e su quelli che possono invece portare avanti un piano d'attacco per vincere la partita in base al tipo di pezzo e alla sua posizione sulla scacchiera.

3.2.4 Costruzione del modello e addestramento

Lo sviluppo è avvenuto con **Keras**, una libreria in Python adatta all'apprendimento automatico e alla creazione di modelli di rete neurale. La rete neurale prende in input le partite contenute del dataset, e per ciascuna viene effettuata la conversione dei dati citata nel paragrafo precedente, in maniera graduale per ogni stato della scacchiera; i neuroni dei livelli successivi passano poi ad apprendere le relazioni che intercorrono tra i dati. Infine, il risultato di tutto l'addestramento è stato registrato in modo persistente con la creazione di **Caissa**.

Ricerca delle mosse

Infine, è stato implementato un algoritmo minimax di ricerca delle mosse che usa Caissa come funzione di valutazione: sebbene si tratti di una soluzione adottata anche nella stesura del primo modulo è interessante notare che, nonostante la differenza di background, la strategia minimax si sia rivelata adatta anche per il funzionamento di un modello di rete neurale.

3.2.5 Punteggio ELO e valutazione del modello

Nell'ambito dei giochi a somma zero è diffuso il metodo di valutazione **ELO** per calcolare le abilità di un giocatore. Si tratta sostanzialmente di un numero il cui valore aumenta o diminuisce a seconda della vittoria o della sconfitta di una partita contro un altro avversario. Per la valutazione delle performance di Caissa, dunque, non è stata effettuata un'analisi da un punto di vista computazionale come era stato fatto per il primo modulo. Questo perché eventuali miglioramenti da apportare a Caissa sarebbero pensati per incrementare la sua performance sportiva: pertanto, è stato scelto un approccio "ad alto livello" sulla base di diverse partite giocate contro altri motori scacchistici preparati dal programmatore. I primi (pochi) incontri sono stati giocati contro il motore *Stockfish 13*. L'intenzione era quella di fissare un upper bound della valutazione di Caissa, essendo *Stockfish* uno dei migliori motori scacchistici moderni con un ELO di circa 3564 alla versione considerata[15].

	Vittorie	**Sconfitte**	**Patte**
Gamma (1190)	0	98	2
Beta (870)	0	85	15
Alpha (512)	0	64	36

Tabella 3.3: Esiti delle partite giocate da Caissa contro Alpha, Beta e Gamma

Come previsto, *Stockfish* ha avuto la meglio in tutte le partite disputate. Sono poi state giocate partite contro motori scacchistici con un punteggio ELO stimato che varia tra i 500 e i 1200 punti. Nello specifico, sono state giocate 100 partite contro ciascun modello: Alpha (512), Beta (870) e Gamma (1190). Nella sua prima versione, in realtà, il punteggio ELO di Caissa è stato stimato al di sotto dei 500 punti, essendo stato battuto persino da Alpha. Nel prossimo capitolo verranno discusse diverse idee che potrebbero essere applicate per migliorare le prestazioni di Caissa e aumentare il suo punteggio. Non solo, verrà anche ripreso il discorso sulle prestazioni degli algoritmi utilizzati nel primo modulo sviluppato.

CAPITOLO 4

Conclusioni e sviluppi futuri

Questo capitolo vuole riassumere il lavoro svolto e analizzarne da un punto di vista critico i risultati ottenuti. Vengono in seguito proposti diversi punti di vista e spunti di riflessione che potrebbero essere utilizzati in futuro per migliorare e rifinire tutto il lavoro svolto in entrambi i moduli.

4.1 Riflessioni sui risultati complessivi

Gli obiettivi del presente lavoro di tesi sono stati raggiunti in maniera conforme a quanto scritto nel Capitolo 1. L'implementazione *from scratch* del primo modulo è stata di grande aiuto per comprendere non solo la complessità di un algoritmo che ricercasse le mosse disponibili ma anche l'importanza nel considerare una funzione obiettivo che valuti lo stato della scacchiera. Inoltre, un lavoro di questo tipo ha stimolato l'idea per la creazione di Caissa, un modello di rete neurale che ha imparato a giocare a scacchi nella sua fase di addestramento. Di seguito vengono esaminati nel dettaglio i punti di forza e di debolezza di ciascuno dei due moduli sviluppati.

4.1.1 Primo Modulo

Sebbene da un lato la "semplice" e visivamente intuitiva implementazione sia risultata favorevole per l'utilizzo di algoritmi di ricerca delle mosse, dall'altro l'analisi delle prestazioni trattata nel paragrafo 3.1.2 ha lasciato emergere dei limiti in termini di tempi di esecuzione degli algoritmi utilizzati.

Threading

Una possibile soluzione al problema riguarderebbe l'utilizzo di diversi **thread**, ciascuno dedicato alla visita di porzioni dell'albero da eseguire in tempi significativamente contenuti. Il main thread, dopo aver atteso le esecuzioni dei diversi thread, raccoglierebbe le informazioni ricevute e le elaborerebbe per decidere la mossa migliore da giocare data la situazione sulla scacchiera. L'obiettivo di questo approccio sarebbe volto dunque a ridurre i tempi di esecuzione del programma, e non la complessità temporale che è invece intrinseca all'algoritmo.

Negascout

Benché la tecnica della potatura alfa-beta abbia ridotto i tempi di visita degli alberi di gioco, si potrebbe optare per la scelta di un algoritmo che in alcuni casi possa addirittura essere più veloce della potatura stessa, a condizione che l'albero sia accuratamente ordinato. Un corretto ordinamento porterebbe l'algoritmo Negascout ad assumere che il nodo corrente sia già quello migliore, producendo un numero maggiore di tagli della potatura alfa-beta. Se venisse visitato un nodo effettivamente migliore di quello considerato si dovrebbe ripercorrere una ricerca normale alfa-beta su quel nodo; un ordinamento scorretto dell'albero porterebbe dunque a tempi di esecuzione più elevati rispetto a un comune algoritmo con potatura alfa-beta. L'adozione dell'algoritmo Negascout nel gioco degli scacchi porterebbe a un incremento delle prestazioni di circa il 10%[16] rispetto alla strategia utilizzata nel modulo implementato.

4.1.2 Secondo Modulo

L'addestramento di Caissa effettuato con le metodologie descritte nel paragrafo 3.2 ha portato alla creazione di un modello dalle performance sportive non del tutto soddisfacenti. Tali limiti vanno probabilmente ricercati nelle tecniche utilizzate per l'addestramento della rete neurale. Con tecniche di apprendimento supervisionato come quelle utilizzate viene fornito un insieme di dati già etichettati e pensati proprio per l'addestramento, ma la natura stessa dei dati rimane pressoché invariata. Di seguito viene proposta un'alternativa alla metodologia di apprendimento utilizzata, che tenga conto di una diversa **funzione** che entra in gioco nell'elaborazione dei dati forniti in input alla rete neurale.

Apprendimento per rinforzo (*reinforcement learning*)

Una buona scelta del tipo di addestramento della rete neurale potrebbe ricadere nella metodologia di apprendimento per **rinforzo**. In questo tipo di addestramento l'agente viene addestrato per effettuare delle decisioni in un determinato ambiente in base ai dati percepiti. L'azione effettuata è poi accompagnata da un valore di **ricompensa** proporzionato alla qualità dell'azione svolta, per incoraggiare oppure scoraggiare quella tipologia di azioni in futuro. Benché esistano diverse **funzioni di rinforzo**, esse possono essere ricondotte alla formula:

$$v_{t+1} = (1 - \alpha)v_t(s) + \alpha\Delta_{t+1} \tag{4.1.1}$$

dove $0 < \alpha \leq 1$ e Δ_{t+1} è la ricompensa per una data azione. Il miglioramento delle azioni è effettuato in modo da massimizzare il valore della **funzione di valore d'azione**, cioè di quella funzione che valuta, appunto, l'azione eseguita in uno stato. Un noto algoritmo di apprendimento per rinforzo è il **Q-learning**: viene costruita una Q-table le cui righe e colonne rappresentano rispettivamente i possibili stati dell'ambiente e le ricompense ottenute in quello stato dopo un'interazione con esso da parte del modello. In questo modo, una volta riempita la tabella, dato uno determinato stato l'agente conosce già l'azione da eseguire per ottenere il valore di ricompensa più alto. Benché il Q-learning si presti bene alla risoluzione di problemi dalla complessità contenuta, nel gioco degli scacchi sarebbe improponibile cercare di costruire (e completare!) una tabella che tenga traccia di tutti i possibili stati della scacchiera. Tuttavia, potrebbe rivelarsi utile sfruttare i tanti livelli intermedi di una rete neurale profonda per costruire un algoritmo di *deep reinforcement learning* che tenga conto dei numerosi stati possibili della scacchiera e che possa dunque pesare il valore di ricompensa adattandolo alle circostanze.

4.2 Integrazione dei due moduli

La realizzazione di tutto il lavoro stimolerebbe un'integrazione dei due moduli in maniera complementare. Potrebbe essere utile sfruttare l'implementazione del primo modulo per consentire a un giocatore umano di giocare contro il computer e conservare in un file le mosse e gli esiti delle partite. Questo file, se adeguatamente grande, potrebbe essere messo a disposizione in futuro per l'addestramento di una rete neurale (adottando le convenzioni del caso). Un lavoro del genere porterebbe dunque alla creazione di un motore scacchistico dallo stile di gioco affine a quello del giocatore, che potrebbe sfruttare il motore come strumento di *Self-Play* per migliorare il proprio punteggio ELO.

Bibliografia

[1] Claude E Shannon. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950. (Citato a pagina 2)

[2] Wikipedia. Numero di shannon — wikipedia, l'enciclopedia libera, 2021. (Citato a pagina 2)

[3] Fédération Internationale des Échecs. Fide handbook. `https://handbook.fide.com/chapter/E012018`. (Citato a pagina 3)

[4] Wikipedia. Valore dei pezzi degli scacchi — wikipedia, l'enciclopedia libera, 2019. [Online; in data 6-settembre-2022]. (Citato a pagina 8)

[5] Ferdinando Colombo. *Introduzione alla teoria dei giochi*. Carocci, 2003. (Citato a pagina 10)

[6] Jonathan Schaeffer, Neil Burch, Yngvi Bjornsson, Akihiro Kishimoto, Martin Muller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *science*, 317(5844):1518–1522, 2007. (Citato a pagina 11)

[7] Stuart J Russell and Peter Norvig. *Intelligenza artificiale. Un approccio moderno*, volume 1. Pearson Italia Spa, 2005. (Citato a pagina 11)

[8] Michele Godena and Bruno Codenotti. *L'eterna sfida: Scacchi e intelligenza artificiale da Turing ad AlphaZero*. HOEPLI EDITORE, 2021. (Citato a pagina 12)

[9] Garry Kasparov and Frederic Friedel. Reconstructing turing's "paper machine". *EasyChair Preprint*, (3):20, 2017. (Citato a pagina 12)

[10] Monty Newborn. *Kasparov versus Deep Blue: Computer chess comes of age*. Springer Science & Business Media, 2012. (Citato a pagina 13)

[11] Wikipedia contributors. Stockfish (chess) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Stockfish_(chess)&oldid=1105171756`, 2022. (Citato a pagina 13)

[12] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. (Citato a pagina 13)

[13] Matthew Sadler. The tcec15 computer chess superfinal: A perspective. *J. Int. Comput. Games Assoc.*, 41(3):164–167, 2019. (Citato a pagina 14)

[14] Christopher Clark and Amos Storkey. Training deep convolutional neural networks to play go. In *International conference on machine learning*, pages 1766–1774. PMLR, 2015. (Citato a pagina 23)

[15] Austin A Robinson. Teaching ai to play chess like people. 2021. (Citato a pagina 25)

[16] Alexander Reinefeld. An improvement to the scout tree search algorithm. *J. Int. Comput. Games Assoc.*, 6(4):4–14, 1983. (Citato a pagina 28)

Siti Web consultati

- Chess.com – `www.chess.com`
- Lichess – `www.lichess.org`
- pygame – `https://www.pygame.org/docs/`
- python-chess – `https://python-chess.readthedocs.io/en/latest/`
- Tensorflow, Keras – `https://www.tensorflow.org/api_docs/python/tf/keras/`
- Wikipedia – `www.wikipedia.org`

Ringraziamenti

Quando ho cominciato il mio percorso universitario mi è capitato diverse volte di proiettarmi con la mente verso il futuro e chiedermi - come tutti - come si sarebbe concluso il mio percorso. Quali ambienti avrei frequentato, a quali persone mi sarei affezionato e quanti successi e fallimenti avrei dovuto vivere. E mentre sono nella mia stanza a scrivere queste parole balenano nella mia mente ricordi della mia vita accademica che conserverò per sempre. Il calore e l'affetto costantemente trasmessi dalle meravigliose persone che ho avuto modo di conoscere mi hanno fornito l'energia che mi ha permesso di farmi forza nei momenti più infelici, e la serenità di godermi appieno le innumerevoli gioie che questo percorso ha saputo regalarmi.

Tutta la realizzazione del presente lavoro di tesi è stata supervisionata dal costante e attento supporto del mio relatore Prof. Fabio Palomba e del Dott. Giammaria Giordano. Sin da subito sono stati disponibili ad ascoltare le mie idee, arricchendole con preziosi consigli che hanno senza dubbio apportato grandi miglioramenti al progetto di partenza. Doveroso porre loro i miei più sinceri ringraziamenti.

Un ringraziamento dal profondo del cuore al mio amico Vincenzo Emanuele. È stata la prima persona ad essere messa al corrente del progetto, e il suo appoggio ha rappresentato un'importante spinta per la realizzazione finale. Gli sono grato per avermi sostenuto e per aver creduto in me fino alla fine. Sei un caro amico.

Voglio ringraziare i miei insostituibili compagni di corso e amici Alessio e Alessandro. Abbiamo realizzato diversi progetti insieme, abbiamo trascorso ore preziose in ateneo a studiare e a confrontarci, e il vostro contributo è stato fondamentale per il conseguimento dei miei obiettivi. Grazie ad entrambi.

L'umorismo e l'euforia dei miei grandi amici Francesco, Luigi e Simone hanno significativamente contribuito a rendere piacevoli e indimenticabili le giornate trascorse in Università

nell'ultimo anno. Sono sempre stato grato per la vostra presenza nella mia vita accademica e non solo. Vi ringrazio in maniera sincera.

Ci tengo a ringraziare Antonio. Ho ancora impressa nella mia mente quella pausa caffè dopo le lezioni, passata a discutere insieme dei prossimi appelli d'esame ai quali non ero minimamente intenzionato a partecipare. Ma dalla nostra conversazione è nata la determinazione che mi ha portato a sostenere gli esami con quel senso di sfida che mi ha accompagnato fino alla fine. Se sono qui adesso è anche grazie a te.

Un ringraziamento a Lorenzo ed Hermann, due preziosi compagni i cui consigli e la cui saggezza hanno sempre saputo indirizzarmi vero la direzione giusta. I risultati del mio percorso sono dovuti anche alla vostra compagnia. Vi ringrazio.

Grazie al mio caro amico Giuseppe. Apprezzo molto il rapporto che abbiamo costruito insieme tra i banchi di Liceo, e mi rende felice sapere che la nostra amicizia perdura negli anni. Ti sono grato per non aver mai smesso di credere in me.

Ringrazio di cuore la mia amica Viviana. Con la tua dolcezza e la tua ironia sei sempre riuscita a farmi vivere l'Università a cuor leggero. Sono felice di aver condiviso il mio percorso anche insieme a te, e ti sono grato per esserci sempre stata.

Le ultime settimane del mio percorso universitario sono state accompagnate dalla piacevole presenza della mia amica Anna Linda. Mi hai sempre ascoltato nei miei momenti di apprensione di fine percorso, e ti ringrazio per esserci stata.

Ho inoltre avuto modo di gioire della compagnia della mia amica Karina. Le nostre diverse origini e culture mi hanno aiutato a crescere e ad aprire la mia mente. Ti sei costantemente interessata al mio percorso universitario, e sei sempre riuscita a farmi percepire il tuo sostegno anche a lunga distanza. *Thank you!*

Un sincero ringraziamento alla mia cara amica Anna. Hai da sempre costituito una forte presenza nella mia vita, e ti sei mostrata disponibile sin da subito nel supportarmi concretamente nel mio lavoro. Mi riempie il cuore di gioia sapere di poter contare sulla persona squisita che sei e che sei sempre stata. Grazie.

Un profondo riconoscimento va a tutti i miei colleghi che, in un modo o nell'altro, hanno contribuito a farmi sentire il calore e l'affetto di cui parlavo poco fa. Mentre scrivo ho bene a mente tutti i vostri volti e i vostri sorrisi: se ho vissuto tre anni di gioia e spensieratezza lo devo a voi tutti. Voglio ringraziarvi.

Ringrazio mia madre Carmela, mia sorella Adelaide e Pasquale per il loro supporto. Il più importante dei ringraziamenti va, infine, a mio nonno Francesco e a mia nonna Lucia. Senza di loro, niente di tutto ciò sarebbe stato possibile.