UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

Utilizarea rețelelor neuronale convoluționale în aprofundarea jocului de șah

Conducător științific
Lect. Dr. Coroiu Adriana Mihaela

Absolvent
Turturică Dorin

2023

ABSTRACT

The bachelor thesis presents a chess web application aimed to offer an enriching, personalized gaming experience, promoting accelerated learning in an interactive environment. The project incorporates a Convolutional Neural Network (CNN) model, functioning as both a competitive adversary and insightful mentor. This model synergizes a Minimax algorithm with Alpha-Beta Pruning for state generation and evaluation, along with a CNN model for positional score estimation, mirroring high-performing chess agents like Stockfish.

Moreover, the application allows users to login/register via email or Google. It offers versatile gaming modes: singleplayer with an adjustable AI bot, multiplayer with a chosen friend or random opponent, and an AI-assisted game analysis function. Also, the thesis explores the integration of webcam utilization in chess matches, ensuring fair play and promoting real-time, immersive interaction between opponents. Thus, it showcases the convergence of deep learning and interactive gaming to enhance the chess-playing experience.

I declare that this thesis does not contain any plagiarism, and all bibliographic sources (including data, figures and tables) have been used respecting the Romanian legislation and the international conventions corresponding to copyright, and are cited in the text of the thesis and in the Reference section. I understand plagiarism constitutes an offence punishable by law.

Cuprins

Capitolul 1

Introducere

Șahul, unul dintre cele mai vechi și prestigioase jocuri strategice din lume, are o istorie profundă care traversează mai multe milenii. Forma actuală a șahului a fost stabilită în jurul secolului al XV-lea, după o evoluție îndelungată.

Jocul se remarcă prin complexitatea sa, punând la încercare logica, strategia și previziunea jucătorilor. Șahul a generat o bogată literatură de strategii și analize, contribuind semnificativ la dezvoltarea inteligenței artificiale. Profunzimea jocului a motivat mulți, inclusiv noi, a folosi șahul drept un domeniu de testare pentru algoritmii de căutare și planificare, iar în ultimul deceniu a reprezentat un cămin pentru metodele de învățare automată profundă (deep learning).

Lucrarea de licență propusă abordează problema dezvoltării unei aplicații web pentru șah care oferă o experiență de joc dezvoltatoare și personalizată, accelerând învățarea, într-un mod autentic și interactiv.

Un element central al acestei experiențe este dezvoltarea și implementarea unui model de rețea neuronală convoluțională (CNN) pentru a oferi un adversar și un îndrumător inteligent în aplicația de șah. Structura agenților prezintă similitudini cu cei mai performanți agenți disponibili pe piață, precum Stockfish. Această arhitectură este obținută prin combinarea unui algoritm Minimax cu Alfa-Beta Pruning, utilizat în generarea și evaluarea stărilor, împreună cu un model de rețea neuronală convoluțională pentru a estima scorul poziției. Modelul CNN-ul oferă o metodă avansată de procesare a tablei de șah, pe de o parte permițând o analiză cu scop mentorial de învățare ca analizator, și un joc strategic provocator ca adversar.

Celălalt aspect cheie abordat în această teză este posibilitatea pentru jucători de a folosi o cameră web în timpul partidelor de șah. În primul rând, utilizarea camerei web permite participanților încredere că adversarii lor nu folosesc ajutorul nepermis al unui program sau al unei terțe persoane. În al doilea rând, oferă persoanelor o interacțiune sentimentală, văzându-și adversarii în timp real, contribuind la crearea unei conexiuni de joc plăcute și imersive.

Astfel ne propunem să atingem mai multe obiective, printre care se includ:

- Simplificarea procesului de înregistrare și acces: Aplicația oferă posibilitatea de a se înregistra și a se autentifica cu ușurință prin intermediul Email-ului sau a contului Google. Acest lucru asigură accesul rapid și facil al utilizatorilor la funcționalitățile aplicației.

- Oferirea unei experiențe de joc Singleplayer personalizate: utilizatorii pot alege să joace împotriva unui bot cu nivel de dificultate selectabil. Aceasta permite un joc flexibil, adaptabil la nivelul de competență al fiecărui jucător, promovând învățarea și îmbunătățirea abilităților de șah.

- Crearea unei experiențe interactive Multiplayer: Aplicația oferă posibilitatea de a juca împotriva altor jucători, fie aleatori, fie prieteni, menținând o conexiune imersivă prin intermediul chatului și al camerei web. Aceasta promovează socializarea și competiția autentică, îmbunătățind experiența de joc.

- Promovarea învățării: Aplicația include o funcție de analiză a tablei de joc, oferind utilizatorilor posibilitatea de a învăța de la cel mai bun (în acest caz, modelul AI Stockfish). Aceasta facilitează înțelegerea strategiilor de șah și îmbunătățește abilitățile jucătorilor.

Capitolele lucrării se împart în:

2. "Aspecte teoretice privind Rețelele Neuronale". Explică conceptele fundamentale legate de rețelele neuronale și rolul lor în șah. În cadrul acestuia, se discută despre aplicabilitatea rețelelor neuronale în șah, procesul de convoluție și filtrare, arhitectura specifică a rețelelor neuronale pentru șah și diferitele tipuri de învățare utilizate.

3. "Aspecte teoretice despre șah". Se concentrează pe teoria specifică a șahului, inclusiv evaluarea tablei de joc, reprezentarea acesteia, precum și prezentarea agentului Stockfish. De asemenea, se discută modul în care se realizează căutarea mișcării cu ajutorul algoritmului Minimax îmbunătățit cu Alfa-Beta Pruning și seturile de date folosite pentru antrenarea.

4. "Realizarea aplicației". Detaliază procesul de construire a aplicației. Se discută despre arhitectura aplicației, tehnologiile utilizate, diagramele de clase ale aplicației, precum și despre integrarea Stockfish în aplicație.

5. "Utilizarea aplicației". Descrie experiența utilizatorilor cu aplicația. Include o diagramă de cazuri și detaliază procesul de înregistrare și autentificare, precum și navigarea și funcționalitățile paginii principale.

Pentru realizarea testării și validării aplicației cu utilizatori reali, programul a fost utilizat de 12 prieteni. Aceștia au apreciat în mod deosebit aplicația de șah dezvoltată, explorând funcțiile singleplayer, multiplayer și de analiză a tablei. Totuși, aspectul cel mai apreciat a fost interactivitatea în timp real, o caracteristică distinctivă care transformă profund experiența de joc și care diferențiază această aplicație de alternativele existente.

Capitolul 2

Aspecte teoretice privind Rețelele Neuronale

Acest capitol descrie principiile esențiale privind rețelele neuronale și importanța lor în jocul de șah. Aceasta include o discuție despre cum sunt utilizate rețelele neuronale în șah, metoda de convoluție și filtrare, structura particulară a rețelelor neuronale pentru șah și diversele metode de învățare aplicate.

2.1 Rețelele neuronale și aplicațiile lor în șah

Rețelele neuronale au generat o revoluție în lumea șahului, modificând semnificativ modul în care programele de joc sunt dezvoltate și îmbunătățind procesul de învățare automată. Aceste rețele neuronale, care sunt modele matematice ce imită funcționarea creierului uman, pot fi antrenate pentru a recunoaște tipare complexe și pentru a face predicții bazate pe datele de intrare.

Rețeaua Neurală Convoluțională are un strat de intrare, un strat de ieșire, multe straturi ascunse și milioane de parametri care au capacitatea de a învăța obiecte și modele complexe. Ea eșantionează intrarea dată prin procese de convoluție și împărțire, este supusă unei funcții de activare(unde toate acestea sunt straturile ascunse care sunt parțial conectate), iar la capăt se află stratul complet conectat care duce la stratul de ieșire. Rezultatul păstrează forma originală similară cu dimensiunile imaginii de intrare. Vezi figura 2.1.

2.2 Convoluție & Filtru

Convoluția implică următoarele caracteristici importante:

1. Convulație & Filtru.

2. Conectivitate locală.
3. Partajarea parametrilor.

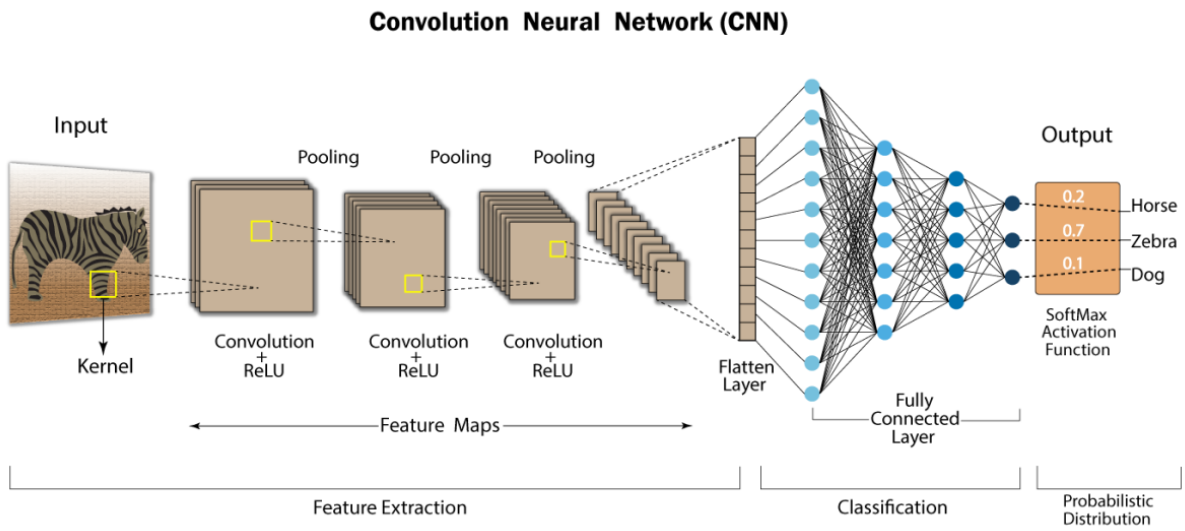


Figura 2.1: Ciclul de dezvoltare al sistemelor bazate pe componente adaptat modelului cascadă.

2.2.1 Convulație și Filtru

Rețelele convoluționale sunt o arhitectură de rețele neuronale puternică pentru jocul de șah. În această arhitectură, se utilizează filtre convoluționale pentru a extrage caracteristici specifice din pozițiile de șah, cum ar fi locația pieselor, structura pionilor sau poziția regelui, mai detaliat prezentate în capitolul 3.3. Aceste caracteristici sunt apoi folosite pentru a dezvolta strategii de joc mai eficiente.

Intrarea unei rețele neuronale convoluționale (CNN) este de obicei un tablou tridimensional de pixeli în care prima dimensiune reprezintă lățimea imaginii, a doua dimensiune reprezintă înălțimea și a treia dimensiune reprezintă adâncimea, explicit prezentat în [Sab17]. Această ultimă dimensiune corespunde numărului de canale pe care imaginea le are, adică imaginile binarizate au un singur canal, deoarece pixelii pot fi doar negri sau albi (0 sau 1). Pe de altă parte, imaginile colorate, care sunt de obicei reprezentate în spațiul de culoare RGB, au 3 canale, în care fiecare canal reprezintă o culoare primară, în care stocăm o valoare între 0-255 reprezentând intensitatea uneia dintre culori.

Cum se poate vedea în figura 2.1, O rețea neuronală convoluțională (CNN) este compusă din una sau mai multe straturi convoluționale, urmate de unul sau mai multe straturi complet conectate (FCs), asemănătoare cu structura rețelei neuronale.

Designul structural al CNN utilizează structura bidimensională a imaginii ca intrări pentru a realiza conexiunea locală, ponderarea, iar apoi împerecherea, dotând CNN cu caracteristici invariante la translație. Comparativ cu rețelele neuronale cu straturi similare, CNN-urile au mai puțini parametri și conexiuni și, prin urmare, sunt mai ușor de instruit. CNN-urile constau din multe straturi convoluționale și de împerechere, iar în final, un strat complet conectat. Un strat convoluțional adoptă o imagine ca intrare și este format dintr-un număr de filtre diferite, în general 3×3, (numite nuclee de convoluție) pentru a efectua operațiuni de convoluție și a produce caracteristici diferite (Ecuația (2.1)).

Principiul convoluțional utilizează o fereastră de dimensiuni reduse pentru a aluneca de la stânga la dreapta și de sus în jos pentru a obține caracteristicile locale din imagine ca intrările stratului următor. Această fereastră alunecătoare este numită nucleu de convoluție sau filtru. Matricea formată prin alunecarea și calculul pe imagine se numește caracteristică de convoluție sau hartă de caracteristici. Harta de caracteristici este outputul către stratul următor prin unitatea liniară rectificată (ReLU) pentru funcția de activare. Este un tip de pooling, deoarece dimensiunea datelor va fi redusă, deci numărul de parametri și calcule sunt reduse, ceea ce accelerează operația sistemului, reduce posibilitatea de suprapotrivire și are efect de anti-interferență. După eșantionare, ieșirile sunt introduse în stratul complet conectat ([KJK16]). Stratul complet conectat este o rețea neuronală generală pentru clasificare. Stratul de conexiune este, de asemenea, cel mai ușor mod de a învăța o combinație non-liniară a caracteristicilor din stratul de convoluție și stratul de împerechere anterior. Aplatizăm harta de caracteristici în stratul complet conectat și actualizăm greutățile în rețeaua neuronală prin backpropagation. Mai multe detalii despre reprezentarea tablei se poate găsi la capitolul 3.2.

$$\text{Input Matrix x Filter Matrix = Feature Map.} \tag{2.1}$$

De exemplu, putem antrena o rețea convoluțională pentru a identifica anumite tipare de poziții care duc la o victorie sau o înfrângere, sau pentru a identifica poziții critice din care să începem să ne dezvoltăm strategia. Rețelele convoluționale sunt, de asemenea, capabile să identifice configurații complexe de piese de șah, ceea ce poate fi util în dezvoltarea unei strategii de joc mai complexe.

În plus, rețelele convoluționale pot fi utilizate pentru a analiza și a învăța din jocurile anterioare. Acest lucru poate fi deosebit de util în cazul jucătorilor profesioniști, care pot utiliza această tehnică pentru a identifica modele și strategii care le pot îmbunătăți jocul.

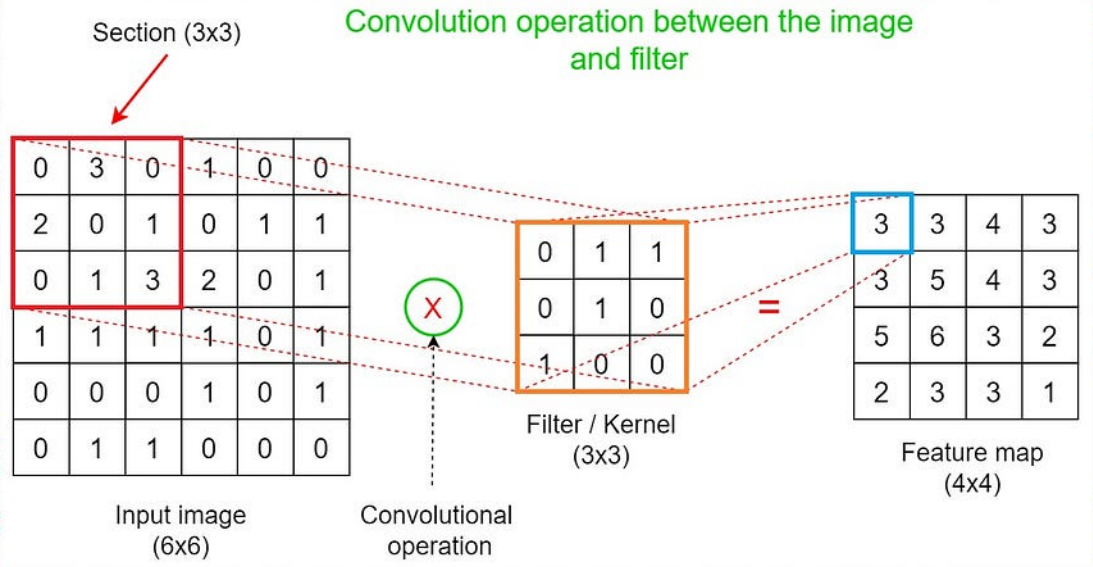


Figura 2.2: În această imagine se aplică matricei de intrare de 6x6, un filtru de 3x3, pentru a obține o matrice de caracteristici de 4x4. Precum în ecuația (2.1).

2.2.2 Conectivitate locală

Precum este prezentat în [Swa20], fiecare neuron este conectat doar la o submulțime a imaginii de intrare (spre deosebire de o rețea neuronală în care toți neuronii sunt complet conectați). Într-o rețea neuronală convoluțională (CNN), se alege o anumită dimensiune de filtru, care se deplasează peste aceste submulțimi de date de intrare. În CNN, există mai multe filtre, iar fiecare filtru se deplasează pe întreaga imagine și învață diferite porțiuni ale imaginii de intrare.

2.2.3 Partajarea parametrilor

Este partajarea greutăților de către toți neuronii dintr-un anumit tablou de caracteristici. Toți aceștia partajează aceeași cantitate de greutate, motiv pentru care se numește partajarea parametrilor.

2.3 Arhitectura rețelelor neuronale pentru șah:

Arhitectura principală pentru rețeaua noastră neuronală va fi convoluțională. Această rețea prezintă mai mulți termeni, printre care:

1. Feed-forward.
2. Neliniaritate.
3. Padding și Stride.
4. Pooling.

5. Regresie.

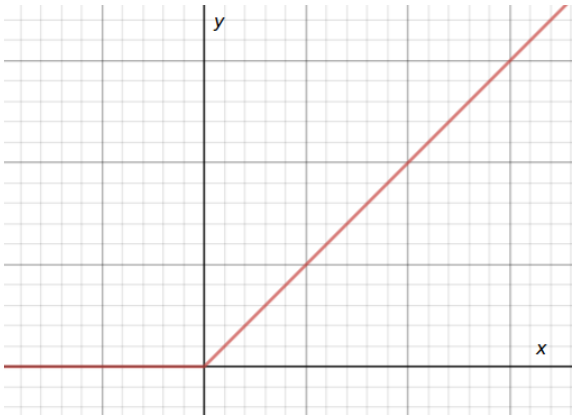
Acești termeni reprezintă elemente fundamentale ale arhitecturii convoluționale și vor fi discutați în continuare.

2.3.1 Feed-forward

Într-un CNN (Convolutional Neural Network) de tip feed-forward, neuronii sunt organizați în straturi, unde fiecare strat preia intrări numai de la cel precedent. Neuronii din primul strat primesc date ca intrare, iar neuronii din ultimul strat produc rezultatul programului. Straturile intermediare sunt denumite ascunse și sunt adesea dens conectate, adică primesc intrări ponderate de la toți nodurile din stratul anterior. După cum s-a descris, deoarece rezultatul fiecărui neuron este pur și simplu o sumă ponderată a stratului anterior, întreaga rețea ar putea fi înlocuită cu un singur strat. Din acest motiv, rezultatul unui neuron este trecut printr-o funcție de activare.

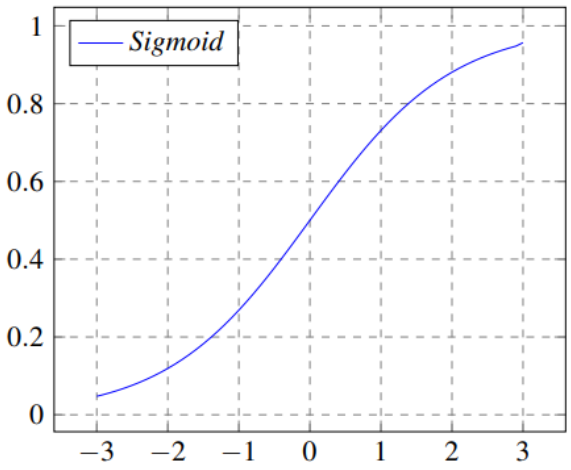
2.3.2 Neliniaritate

Aceasta introduce non-liniaritate și permite rezolvarea problemelor care nu sunt separabile liniar. Există multe tipuri de funcții de activare, deoarece singurul lucru cerut pentru a introduce non-liniaritate este ca ele să nu fie liniare. Pentru CNN, una dintre cele mai comune este ReLU (vezi figura 2.3) - scurt pentru unitate liniară rectificată - care, în ciuda simplității sale, este suficientă pentru a rupe non-liniaritatea. o altă opțiune este funcția de activare sigmoid (vezi figura 2.4).



$ReLU(x) = \max(0, x)$
De exemplu: Într-o matrice dată (M),
M = [[-3, 19, 5], [7, -6, 12], [4, -8, 17]]
ReLU o transformă astfel:
[[0, 19, 5], [7, 0, 12], [4, 0, 17]].

Figura 2.3: Funcție de activare ReLU



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

De exemplu: Într-o matrice dată (M),
M = [[-3, 19, 5], [7, -6, 12], [4, -8, 17]]
Sigmoid o transformă astfel:
[[0.05, 1.00, 0.99], [1.00, 0.00, 1.00],
[0.98, 0.00, 1.00]].

Figura 2.4: Funcție de activare σ

2.3.3 Padding și Stride

Padding-ul și Stride-ul influențează modul în care se efectuează operația de convoluție. Padding-ul și Stride-ul pot fi folosite pentru a modifica dimensiunile (înălțimea și lățimea) vectorilor de intrare/ieșire fie prin creștere, fie prin scădere.

Padding-ul permite mai mult spațiu pentru ca kernel-ul să acopere imaginea și este precis pentru analiza imaginilor. Datorită padding-ului, informațiile de pe marginile imaginilor sunt, de asemenea, păstrate similar cu cele din centrul imaginii. Vezi figura 2.5.

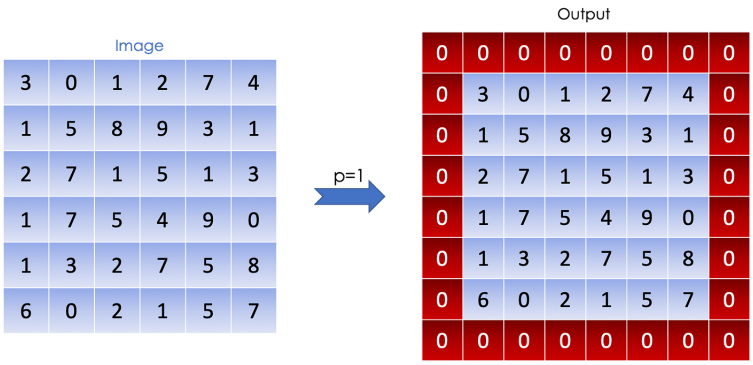


Figura 2.5: p = dimensiunea padding, în cazul nostru bordează matricea cu 1 strat.

Stride-ul controlează modul în care filtrul se convoluează peste intrare, adică numărul de pixeli cu care se deplasează în matricea de intrare. Dacă stride-ul este setat la 1, filtrul se deplasează cu 1 pixel la fiecare pas, iar dacă stride-ul este 2, filtrul se deplasează cu 2 pixeli la fiecare pas. Cu cât este mai mare valoarea stride-ului, cu atât va fi mai mică ieșirea rezultată. Vezi figura 2.6.

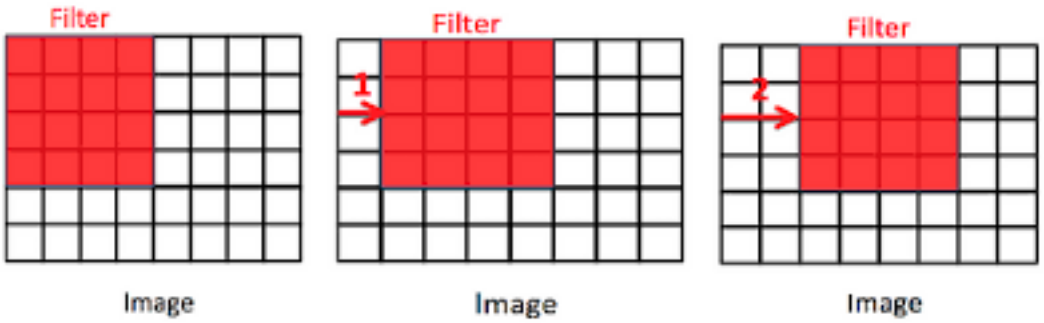


Figura 2.6: Imaginea din mijloc are $stride$ = 1, iar cea din dreapta $stride$ = 2.

2.3.4 Pooling

Pe lângă convoluții în sine, operațiile de pooling alcătuiesc un alt element de bază important în rețelele CNN. Operațiile de pooling reduc dimensiunea hărților de caracteristici prin utilizarea unei funcții pentru a rezuma subregiunile, cum ar fi calcularea mediei sau a valorii maxime. Acest fenomen poate fi văzut în detaliu aici [DV16].

Poolingul funcționează prin deplasarea unei ferestre peste intrare și transmiterea conținutului ferestrei către o funcție de pooling. Într-un fel, poolingul funcționează foarte asemănător cu o convoluție, dar înlocuiește combinația liniară descrisă de nucleul convoluțional cu o altă funcție. Figura 2.7 oferă un exemplu pentru poolingul mediu, iar ilustrația 2.8 face același lucru pentru poolingul maxim. Următoarele proprietăți afectează dimensiunea rezultatului (oj) unui strat de pooling de-a lungul axei j:

- i_j: Dimensiunea intrării de-a lungul axei j.
- k_j: Dimensiunea ferestrei de pooling de-a lungul axei j.
- s_j: Pasul (distanța între două poziții consecutive ale ferestrei de pooling) de-a lungul axei j.

2.3.5 Regresie

Învățarea în sine se face prin minimizarea a ceea ce este cunoscut sub numele de funcție de pierdere. Funcția de pierdere poate fi, de exemplu, eroarea pătrată medie (MSE) sau eroarea absolută medie (MAE) a predicțiilor, în cazul regresiei. Pentru a minimiza această cantitate, eroarea este permisă să se propage înapoi, realizând un fel de coborâre a gradientului pe spațiul problemei, utilizând derivata fiecărui strat, așa cum este prezentat în [Jan17].

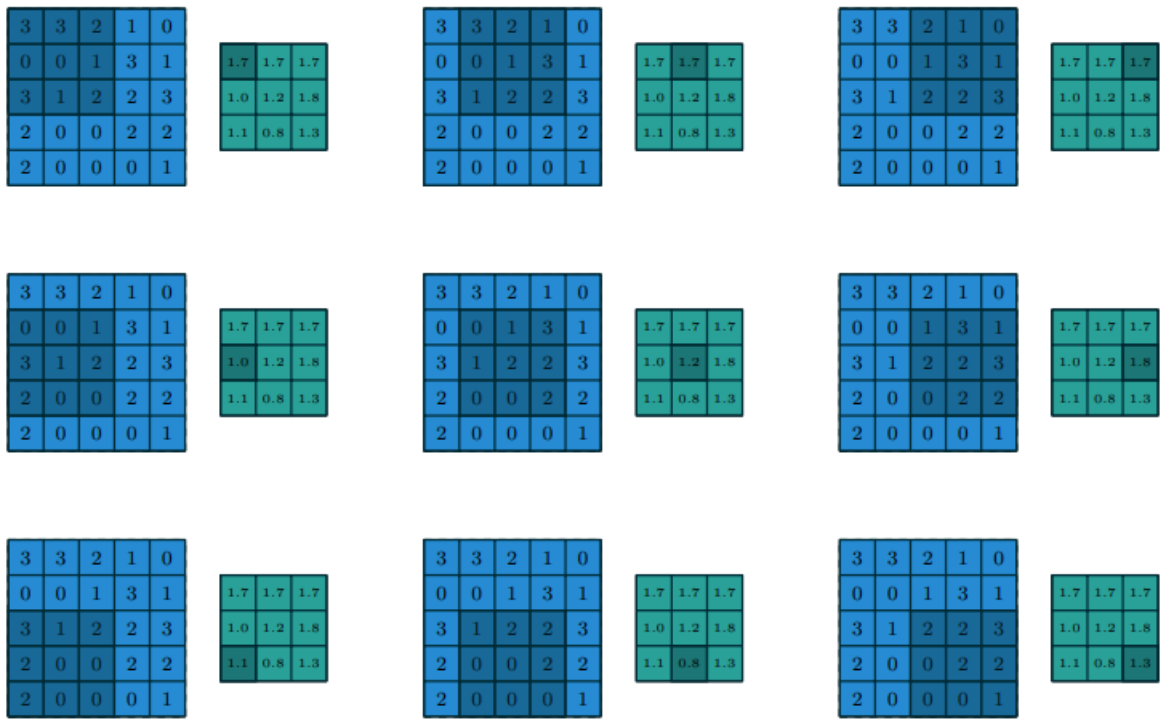


Figura 2.7: Calcularea valorilor de ieșire ale unei operații de pooling mediu de 3 × 3 pe o intrare de 5 × 5 folosind pași de 1 × 1.

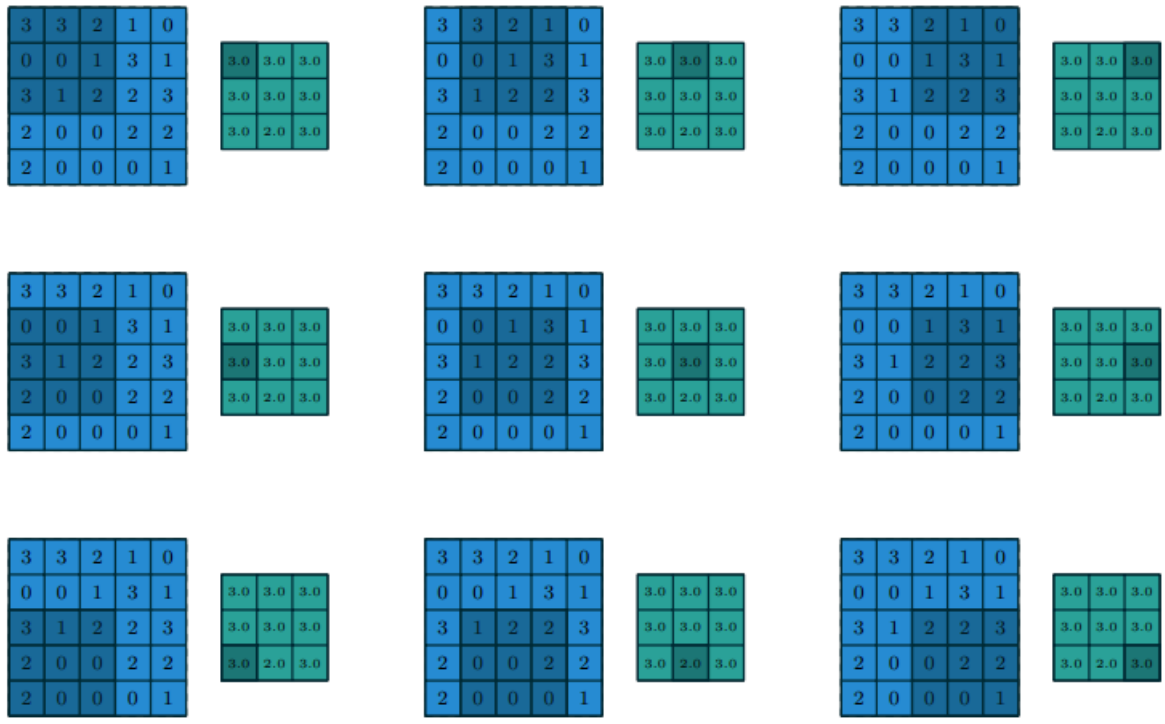


Figura 2.8: Calcularea valorilor de ieșire ale unei operații de pooling maxim de 3 × 3 pe o intrare de 5 × 5 folosind pași de 1 × 1.

$$\text{MSE} = \frac{1}{n}\sum(Y_i - \hat{Y}_i)^2 \qquad (2.2) \qquad\qquad \text{MAE} = \frac{1}{n}\sum|Y_i - \hat{Y}_i| \qquad (2.3)$$

Figura 2.9: Unde n = numărul de observații, Y_i = valoarea reală, iar $\hat{Y}_i$ = valoarea prezisă.

2.4 Tipuri de învățare:

2.4.1 Supervizată

Învățarea supervizată este o metodă de antrenare a modelelor de inteligență artificială care utilizează un set de date etichetate pentru a învăța să prezică o ieșire (rezultatul) pe baza unei intrări (date de intrare). Aceasta metoda o folosim și noi.

Regresia este o tehnică prin care încercăm să găsim o "regulă" matematică care să ne ajute să prezicem o anumită valoare pe baza altor valori pe care le avem. De exemplu, să spunem că avem date despre câți bani cheltuiește o persoană pe mâncare în funcție de venitul său lunar. Utilizând regresia, putem găsi o formulă matematică sau o funcție care să ne permită să estimăm cât de mult va cheltui o persoană pe mâncare pe baza venitului său.

Scopul este să găsim această formulă sau funcție astfel încât estimările noastre să fie cât mai apropiate de realitate. Regresia se bazează pe observațiile pe care le avem și încearcă să găsească o relație între variabila pe care vrem să o prezicem (numită variabilă dependentă) și alte variabile pe care le avem (numite variabile independente sau factori).

Regresia liniară este o metodă simplă de regresie care presupune că există o relație liniară între variabila independentă și variabila dependentă. Formula pentru regresia liniară este:

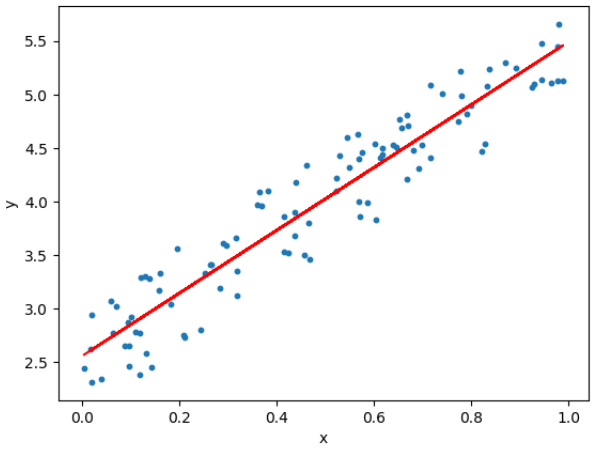


Figura 2.10: Regresie linera

$$y = b_0 + b_1 x$$

Figura 2.11: Unde y - variabila dependentă, x - variabila independentă, b_0 - interceptul (valoarea lui y când x = 0) și b_1 este coeficientul de regresie (pentru a vedea cât de mult crește sau scade y pentru fiecare unitate de creștere a lui x).

De exemplu, dacă încercăm să prezicem prețul unei case în funcție de dimensiunea acesteia, regresia liniară ne-ar ajuta să găsim o relație între dimensiunea casei și prețul acesteia.

Regresia polinomială este o altă metodă de regresie care poate fi utilizată atunci când relația între variabila independentă și variabila dependentă nu este liniară. De exemplu, dacă există o relație curbată între dimensiunea casei și prețul acesteia, regresia polinomială poate fi utilizată pentru a găsi o relație matematică între cele două variabile.

Formula pentru regresia polinomială este:

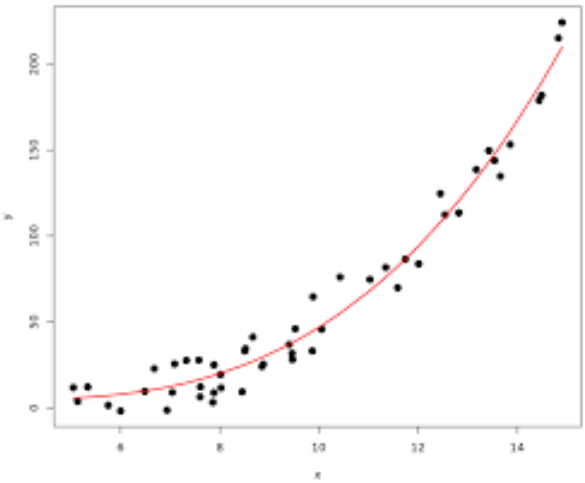


Figura 2.12: Regresie polinomială

$$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n$$

Figura 2.13: Unde y - variabila dependentă, x - variabila independentă, b_0, b_1, b_2, ..., b_n - coeficienții, n - gradul polinomului.

De exemplu, dacă există o relație curbată între dimensiunea casei și prețul acesteia, regresia polinomială ne-ar ajuta să găsim un polinom care să se potrivească cel mai bine cu datele noastre și să ne ajute să prezicem prețul casei în funcție de dimensiunea acesteia.

2.4.2 Nesupervizată

Învățarea nesupervizată este o metodă de antrenare a modelelor de inteligență artificială care nu utilizează date etichetate, ci învață să găsească structuri și modele ascunse în datele de intrare. În cazul jocului de șah, învățarea nesupervizată poate fi utilizată pentru a identifica modele de joc sau configurații de piese care apar frecvent și care pot influența strategiile de joc.

2.4.3 Prin întărire

Învățarea prin întărire este o metodă de antrenare a modelelor de inteligență artificială care se bazează pe interacțiunea modelului cu un mediu și pe primirea de

recompense sau pedepse în funcție de acțiunile întreprinse. În cazul jocului de șah, învățarea prin întărire poate implica antrenarea unui agent de joc pentru a lua decizii în funcție de pozițiile de șah și de starea jocului, și pentru a primi recompense sau pedepse în funcție de rezultatul final al partidelor. Agentul de joc își îmbunătățește strategiile pe măsură ce interacționează cu mediul și își maximizează recompensele pe termen lung. Această abordare poate fi utilizată pentru a dezvolta programe de joc de șah care pot învăța și se pot adapta pe parcursul partidelor, îmbunătățindu-și performanța și strategiile în timp.

Capitolul 3

Aspecte teoretice despre șah

În acest capitol vom prezenta principiile particulare ale șahului, cum ar fi: evaluarea și reprezentarea tablei de joc, introducerea în agentul Stockfish, modul în care se efectuează căutarea mișcărilor prin intermediul algoritmului Minimax (îmbunătățit cu Alfa-Beta Pruning) și seturile de date utilizate. Pentru un start foarte accelerat de informație, site-ul [che18] oferă date valoroase despre începerea unui program de șah, iar pentru partea concretă a unui model python de CNN, videoclipul acesta [Sec21] explică pașii ușor și descriptiv.

3.1 Evaluarea tablei de joc

Înțelegerea precisă a unei poziții este un element cheie în jocul de șah. Spre deosebire de ceea ce cred majoritatea oamenilor, jucătorii de șah cu rating înalt nu se diferențiază de cei cu rating mai mic în capacitatea lor de a calcula multe mutări înainte. Dimpotrivă, ceea ce face ca maeștrii internaționali de șah să fie atât de puternici este abilitatea lor de a înțelege foarte rapid cu ce situație se confruntă pe tablă. În funcție de aceste evaluări, ei decid ce linii de șah să calculeze și câte poziții înainte trebuie să verifice înainte de a se angaja într-o mutare efectivă.

Este posibil să identificăm un compromis între numărul de stări viitoare ale tablei care trebuie explorate și precizia evaluării unei stări curente a tablei. De fapt, dacă evaluarea unei anumite poziții de șah este foarte precisă, nu este nevoie să explorăm un set mare de stări viitoare ale tablei. Un caz foarte simplu este prezentat în Figura 3.1, unde este rândul negrului să facă o mutare.

După cum se poate observa din figura 3.1, Alb amenință Regina neagră cu calul din c3. Chiar și un jucător amator știe că Regina este piesa cea mai valoroasă de pe tabla de șah și că este piesa care, după Rege, merită cea mai mare atenție. Acest lucru face procesul de evaluare al poziției prezentate foarte ușor: Negrul trebuie să mute Regina pe o căsuță în care nu va mai fi amenințată de piesele Albului. Pe lângă

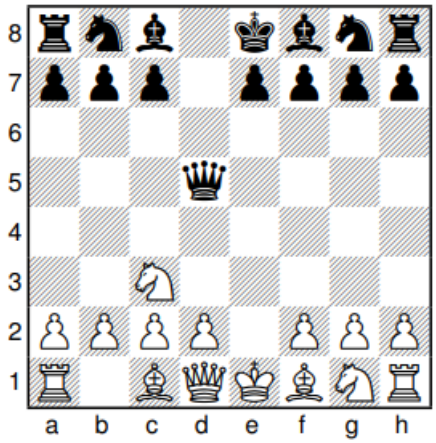


Figura 3.1: Exemplu de poziție care nu necesită multa analiză pentru a fi evaluată în mod precis.

faptul că este foarte ușor de evaluat, evaluarea poziției în sine este și foarte precisă, deoarece Negrul nu trebuie să investească timp în calcularea unor linii de șah lungi și complicate pentru a înțelege că, dacă nu își mută Regina pe o căsuță sigură, jocul va fi pierdut în curând.

Maeștrii internaționali de șah sunt foarte buni în evaluarea unui set mult mai mare de poziții de șah care sunt, de obicei, mai complicate decât cea prezentată anterior, dar în cea mai mare parte se bazează doar pe previziuni pentru a verifica dacă evaluările lor inițiale, care se bazează pe intuiție, sunt corecte. Prin aceasta, ei se asigură că minimizează șansele de a face o mutare care duce la pierdere.

Scopul principal al acestui lucru este de a învăța un sistem să evalueze pozițiile de șah foarte precis, cu ajutorul explorării stărilor viitoare ale tablei care folosesc algoritmi de previziune. Pentru a face acest lucru, modelăm această metodă de antrenament ca o sarcină de clasificare și ca o sarcină de regresie. În ambele cazuri, diferite arhitecturi de rețele neurale artificiale (ANN) trebuie să fie capabile să evalueze pozițiile de șah care au fost punctate de Stockfish.

3.2 Reprezentarea tablei

Un aspect foarte important asupra acurateței învățării algoritmului nostru CNN este influența tipului de reprezentare vectorială a tablei de șah. Tipul de reprezentare ales este reprezentarea prin bitmap, prezentată în Figura 3.2. În această metodă, pentru fiecare dintre cele 2 culori (alb, negru) și fiecare dintre cele 6 piese (pion, cal, nebun, tură, regină, rege) există o matrice, deci 2 x 6 = 12 matrici. O abordare diferită ar fi să adăugăm și toate căsuțele atacate de un jucător, adăugând încă 2 matrici, așa cum este prezentat în articolul [Pur21], dar am ales o abordare mai simplistă. Matricea noastra este populată cu 1 dacă piesa respectivă este prezentă pe pătra-

tul corespunzător, altfel conține 0. Această hartă de caracteristici este reconfigurată într-un tensor cu dimensiunea de 8x8x12, unde primele două dimensiuni se referă la mărimea tablei de șah, iar ultima dimensiune reprezintă numărul de straturi de caracteristici ale datelor de intrare. O astfel de reprezentare a tablei conferă vectorului o lungime totală de 768, reflectând poziția completă a tablei de șah.

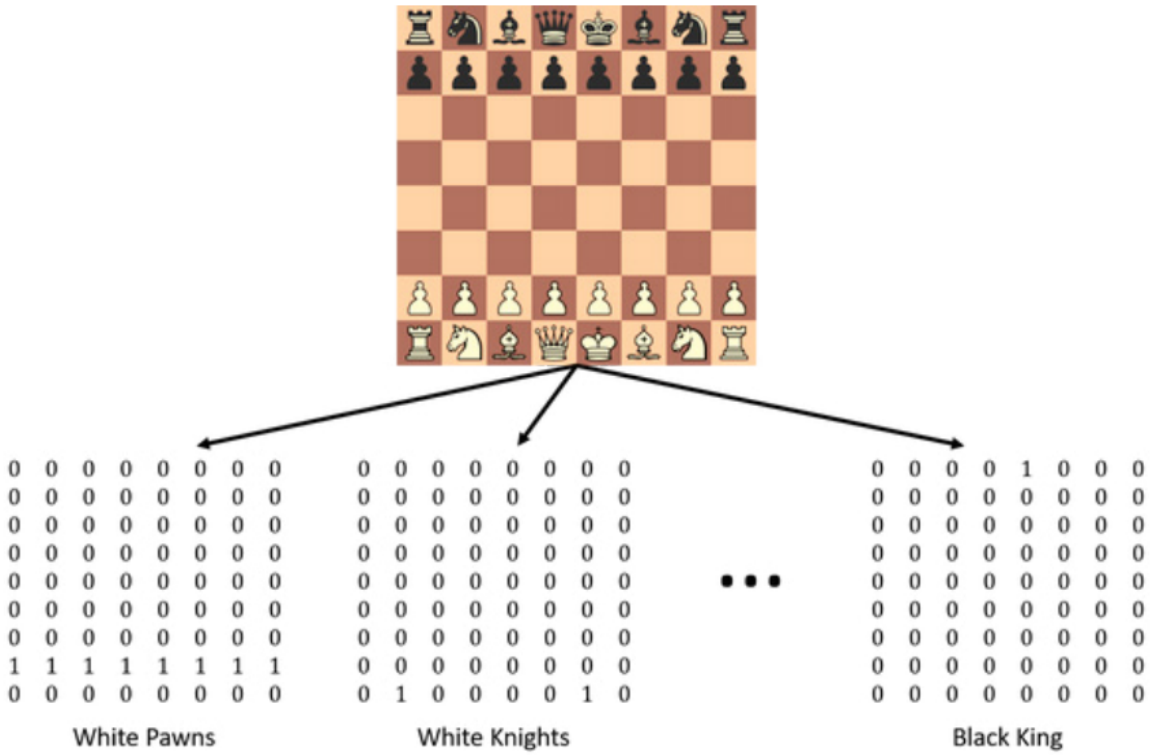


Figura 3.2: Reprezentare bitmap. Se poate observa că pentru un tip de piesă și culoare există o matrice, care conține 1 dacă pe poziția respectivă este piesa de acea culoare.

3.3 Stockfish

Lansat sub licența GPL, Stockfish este unul dintre cele mai puternice motoare de șah open source din lume. Conform Listei Internaționale de Clasament a Șahului pe Calculator (CCRL), ocupă primul loc în lista motoarelor de calculator care joacă șah, sau cum spune Chess.com: "În prezent, cel mai puternic motor de șah disponibil publicului" ([Che21]). Stockfish evaluează pozițiile de șah pe baza unei combinații de cinci caracteristici principale diferite și un algoritm de previziune. Cele mai importante caracteristici sunt:

1. Echilibrul de material: aceasta este probabil cea mai intuitivă și ușor de înțeles caracteristică a listei. De fapt, în majoritatea timpului, pozițiile egale prezintă

aceeași cantitate exactă de piese pe tablă. Pe de altă parte, dacă un jucător are mai multe piese decât adversarul, foarte probabil are un avantaj care face posibilă câștigarea jocului. Se poate observa în Figura 3.3 ca materialul este cel mai important.

2. Structura pionilor: în ciuda a ceea ce cred majoritatea jucătorilor naivi, pionii sunt piese foarte puternice pe tablă. Importanța lor crește odată cu trecerea timpului până când se ajunge la finalul jocului, unde, împreună cu regele, pot decide rezultatul final al unui joc. Stockfish oferă evaluări mai mici dacă pionii sunt dublați, neapărați, nu controlează pătratele centrale ale tablei sau au șanse mici de a fi promovați.
3. Amplasarea pieselor: poziția pieselor în raport cu numărul de pătrate pe care le controlează este un concept foarte important, în special în mijlocul jocului. Cazurile care cresc șansele de câștig sunt, de exemplu, nebunii care controlează diagonale mari pe tablă, caii care acoperă cele mai centrale pătrate și turnurile care atacă rândurile apropiate de regele advers.
4. Pionii promovați: pionii au capacitatea de a fi promovați la piese de valoare superioară dacă ajung pe partea opusă a tablei și, ca urmare, pot duce la poziții câștigătoare. Pionii care nu au pioni adversari care să-i împiedice să avanseze către a opta linie îmbunătățesc scorul de evaluare al lui Stockfish, deoarece au șanse mai mari de a fi promovați. Aceste șanse devin chiar mai mari dacă regele advers este foarte departe.
5. Siguranța regelui: deoarece obiectivul principal al șahului este de a da șah-mat regelui advers, este foarte important ca această piesă în special să fie cât mai sigură posibil. Stockfish acordă prioritate rocării și tuturor pieselor care împiedică adversarul să atace direct regele.

Figura 3.4 reprezintă 4 poziții diferite de șah în care caracteristicile motorului Stockfish au un impact semnificativ asupra evaluării. Excludem prima caracteristică legată de echilibrul material, deoarece este foarte intuitivă și ușor de înțeles.

Cele 5 caracteristici prezentate anterior sunt cele mai importante. Modelul agentului Stockfish este mai complex și dificil de digerat, dar aceste 2 articole [Cha21a], [Cha21b], fac o treaba minunată în a diseca informația. Cu toate acestea, șahul poate deveni incredibil de complex, iar o evaluare mai precisă poate fi obținută numai prin utilizarea algoritmului de previziune. De fapt, este posibil să aveți un rege foarte nesigur și, în același timp, să amenințați mat datorită unei mobilități ridicate particulare a pieselor. Pentru a evalua aceste condiții particulare foarte precis, Stockfish folosește algoritmul de previziune cunoscut sub numele de tăiere $\alpha - \beta$. Bazat pe simpla regulă MinMax, acesta poate explora aproximativ 30 de noduri adâncime într-un minut, în arborele de mutări posibile și respinge cele care, pe baza unei

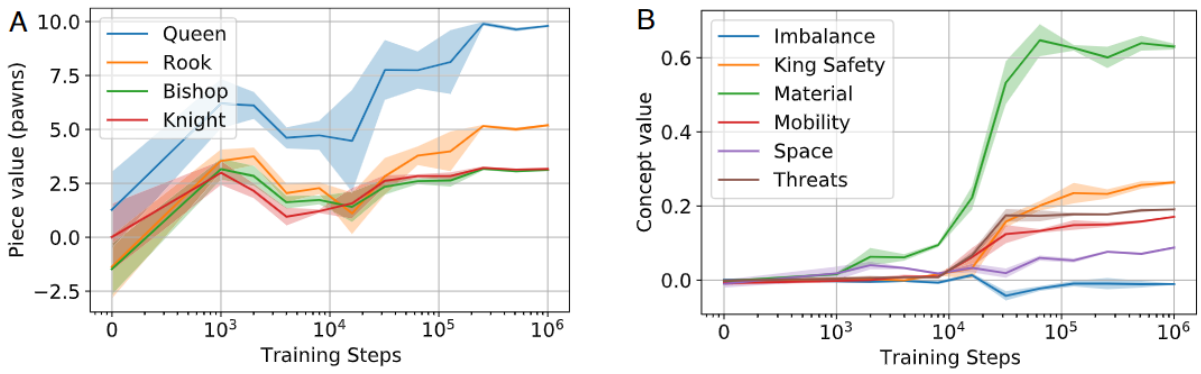


Figura 3.3: Regresia valorilor în funcție de conceptele definite de oameni în timp, conform [MKT+22]. (A) Greutățile valorilor pieselor tind către valori apropiate de cele folosite în analiza convențională. (B) Materialul prezice valoarea în stadiile inițiale ale antrenamentului, iar concepte mai subtile, precum mobilitatea și siguranța regelui, apar mai târziu.

mișcări de contracarare, duc la poziții de pierdere. Vom explica acum în detaliu cum funcționează acest algoritm specific.

3.4 Căutarea Mișcării

Indiferent de ceea ce cred oamenii, șahul rămâne încă un joc nerezolvat. Un joc este definit ca fiind rezolvat dacă, având o situație legală pe tablă, este posibil să se prezică dacă jocul se va încheia cu o victorie, o remiză sau o înfrângere, presupunând că ambii jucători vor juca secvența optimă de mutări. În prezent, indiferent de câtă putere de calcul este folosită, este încă imposibil să prezicem acest rezultat. Este adevărat că, de exemplu, Alb are o ușoară avantaj după ce a făcut prima mutare în joc, dar dacă aceasta este suficientă pentru a câștiga întregul joc este încă necunoscut. Pe de altă parte, un exemplu de joc rezolvat este versiunea în limba engleză a jocului de dame, în 2007 s-a dovedit de fapt că, dacă ambii jucători joacă optim, toate jocurile se vor încheia cu o remiză ([SBB+07]). De asemenea, merită menționat că șahul jucat pe un tablă n × n este chiar o problemă EXPTIME-dură, ceea ce pune constrângeri serioase în domeniul programării șahului.

Având aceasta în vedere, se dovedește a fi puțin neinteresant să explorezi algoritmi care permit căutarea tot mai adâncă în timpul procesului de căutare a mutării, deoarece indiferent de adâncimea acestei căutări, va fi totuși imposibil să se ajungă la un joc optim. Este mult mai provocator să înțelegem ce tipuri de configurații ale tablei merită explorate în profunzime și care nu merită să fie analizate.

Procedura de previzualizare poate fi formalizată astfel: notăm cu S toate pozițiile posibile ale tablei în joc și cu $t = 1, 2, \ldots$ turele în care s-a făcut o mutare pe tabla de joc. La fiecare tur t există o poziție de tablă corespunzătoare $x_t \in S$, de unde

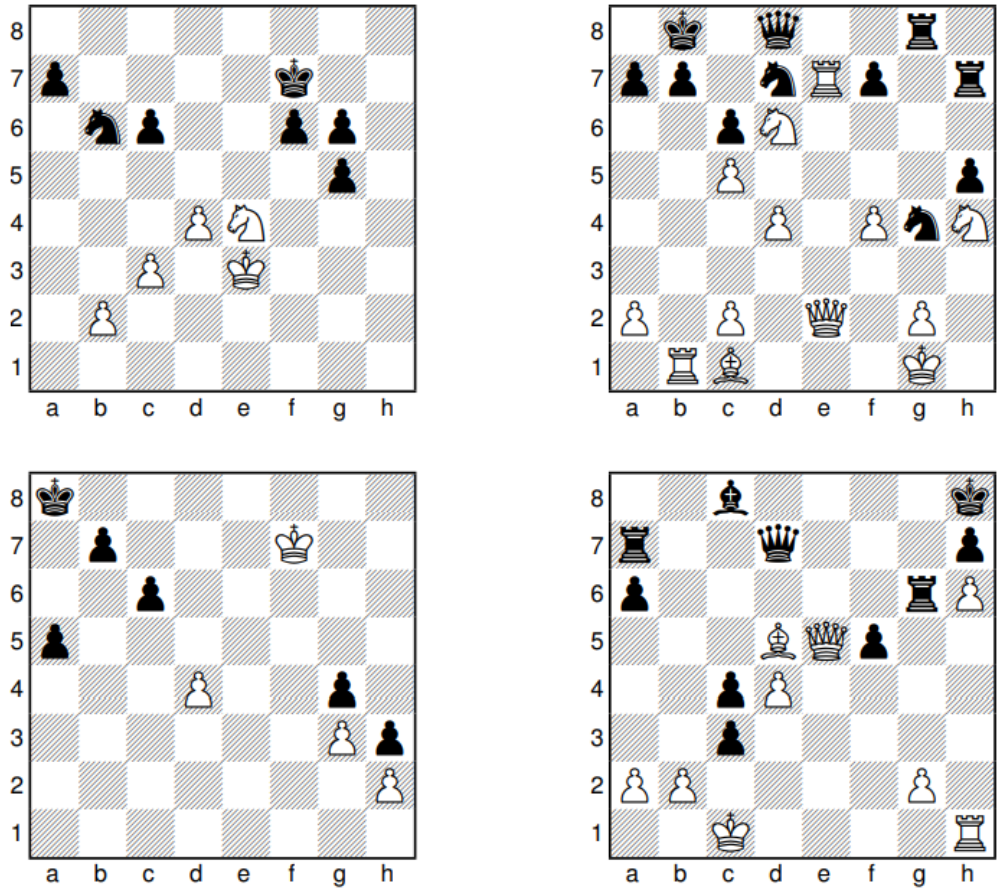


Figura 3.4: De la stânga sus la dreapta jos, sunt prezentate ultimele 4 caracteristici importante ale motorului Stockfish. Prima poziție reprezintă o structură slabă a pionilor pentru jucătorul Negru, care are atât un pion izolat, cât și un pion dublat. A doua poziție evidențiază modul în care piesele albe sunt plasate pe tabla de joc și cum atacă zona apropiată de regele Negru. În cea de-a treia poziție, prezentăm un exemplu de pion trecut în a5, care în curând va promova în regină. Poziția finală reprezintă un atac puternic din partea jucătorului Alb, care verifică regele Negru foarte nesigur cu regina sa.

jucătorul poate alege o mutare $m \in M_t$ care duce la o nouă stare a tablei, x_{t+1}. Ideea principală este de a găsi secvența de mutări care maximizează șansele de a câștiga jocul. În acest studiu, ne propunem să antrenăm o rețea neuronală artificială capabilă să includă această procedură în funcția sa de evaluare, cu ajutorul arborelui de mutări posibile.

3.4.1 Căutarea MinMax și tăierea Alpha-Beta

Șahul este definit ca un joc cu suma zero, ceea ce înseamnă că pierderea unui jucător este câștigul celuilalt jucător ([EWL99]). Ambii jucători își aleg mutările, $m \in M_t$, cu scopul de a-și maximiza șansele proprii de câștig. Făcând acest lucru, minimizează în același timp șansele de câștig ale adversarului lor. MinMax este un algoritm pen-

tru alegerea mulțimii de mutări $m \in M_t$ care duce la situația cea mai favorabilă pentru jucător. Acest lucru se realizează generând S pana la o adâncime d. Odată ce acest lucru s-a întâmplat, o funcție de evaluare este folosită pentru a determina valoarea fiecărei stări a tablei, adică o stare a tablei care aduce cel mai bun scor maxim pentru alb, respectiv cel mai bun scor minim pentru negru. Aceeași funcție utilitară este apoi aplicată în mod recursiv asupra stărilor tablelor x_{t-1} până când se ajunge în vârful arborelui. Odată ce o valoare $\forall x_t \in S$ a fost atribuită, este posibil să se aleagă secvența de mutări care, conform funcției de evaluare, duce la victorie. Teoretic, este posibil să se utilizeze MinMax în șah, însă din cauza complexității computaționale menționate anterior, aceasta nu este fezabilă. MinMax este de fapt un algoritm de căutare în adâncime care are o complexitate de $\mathcal{O}(b^d)$, unde b este factorul de ramificare, iar d corespunde adâncimii căutării.

Pentru a rezolva această problemă, algoritmul de tăiere Alpha-Beta poate fi utilizat. Propus de [KM75], această tehnică este capabilă să calculeze aceleași decizii ca MinMax, dar fără a examina fiecare nod în arborele de căutare. Acest lucru se realizează prin eliminarea $x_t \in S$ care nu sunt relevante pentru decizia finală de alegere a $m \in M_t$. Conform lui [Rus10], principiul general este următorul: să considerăm un nod aleatoriu n_t, la un anumit tur t în arborele de mutări posibile; dacă jucătorul de șah are deja posibilitatea de a ajunge la o m mai bună (cu $M_t \in n_t$), aflându-se la n_{t-1} sau chiar mai sus, este posibil să marcăm n ca nefiind demn de explorare. Ca urmare, n și descendenții săi pot fi eliminați din arborele de mutări. O reprezentare grafica se poate observa in figurile 3.5, 3.6, 3.7 , iar pseudo-codul acestui algoritm este prezentat la începutul pe următoarea pagină:

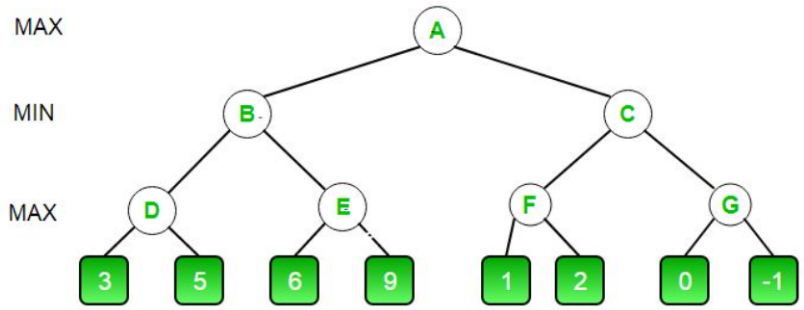


Figura 3.5: Apelul inițial începe de la A. Valoarea α aici este $-\infty$ și valoarea β este $+\infty$. Aceste valori sunt transmise către nodurile ulterioare. La A, maximizatorul trebuie să aleagă maximul dintre B și C, deci A apelează mai întâi B. În cazul în care explicația este prea grea de digerat, [Gee17] descrie în detaliu fiecare pas.

Algoritmul de tăiere α-β oferă o îmbunătățire semnificativă din perspectiva complexității computaționale în comparație cu MinMax. În cazul ideal, complexitatea sa este de fapt $\Theta(b^{d/2})$, ceea ce permite algoritmului să prevadă de două ori mai departe decât MinMax la același cost. Cu toate acestea, în ciuda acestei îmbunătățiri semnifica-

Algorithm 1 Minimax Algorithm

```
1:  function MINIMAX(node, depth, isMaximizingPlayer, α, β)
2:      if node is a leafnode then
3:          return value of the node
4:      if isMaximizingPlayer then
5:          bestVal ← −∞
6:          for all childnode do
7:              value ← Minimax(node, depth + 1, false, α, β)
8:              bestVal ← max(bestVal, value)
9:              α ← max(α, bestVal)
10:             if β ≤ α then
11:                 break
12:         return bestVal
13:     else
14:         bestVal ← +∞
15:         for all childnode do
16:             value ← Minimax(node, depth + 1, true, α, β)
17:             bestVal ← min(bestVal, value)
18:             β ← min(β, bestVal)
19:             if β ≤ α then
20:                 break
21:         return bestVal
```

tive, căutarea arborelui α-β nu este suficientă pentru a rezolva jocul de șah. Cu toate acestea, este foarte potrivită pentru programarea șahului. Trucul constă în marcarea $\forall x_t = d \in S$ ca stări terminale și aplicarea regulii MinMax asupra acestor stări specifice ale tablei pentru a limita adâncimea căutării. Cu cât este mai mare valoarea lui d, cu atât căutarea în arbore va fi mai solicitantă din punct de vedere computațional.

3.5 Seturi de date

Acest capitol este esențial pentru înțelegerea modului în care modelul nostru a fost antrenat și îmbunătățit pentru a juca șah.

Primul set de date pe care l-am utilizat provine din partidele profesionale de șah jucate de către maeștrii grandmasteri începând cu anul 1995, și se poate găsi aici [Kag23]. Acest set de date constituie baza pentru înțelegerea și învățarea modului în care jucătorii profesioniști abordează jocul de șah. Fiecare mișcare, strategie și tactică utilizată de acești experți este capturată și folosită pentru a instrui modelul nostru CNN. Învățând de la cei mai buni, modelul este capabil să recreeze și să simuleze strategiile de joc ale jucătorilor de șah de top.

Al doilea set de date este [Dat17]. Acesta e format dintr-un fișier CSV care conține pozițiile tablei de șah codificate prin notația FEN (Forsyth-Edwards Notation) și o valoare de evaluare asociată. FEN este un standard pentru reprezenta-

rea pozițiilor de șah, permițând modelului să înțeleagă structura curentă a jocului. Valoarea de evaluare furnizează o măsură a avantajului sau dezavantajului unui jucător într-o anumită poziție, ajutând modelul să învețe nu doar care sunt mișcările posibile, ci și calitatea lor strategică.

Combinând aceste două seturi de date, modelul nostru CNN este capabil să înțeleagă atât complexitatea strategică a jocului de șah, cât și valoarea individuală a fiecărei mișcări în contextul unei anumite poziții pe tabla de șah.

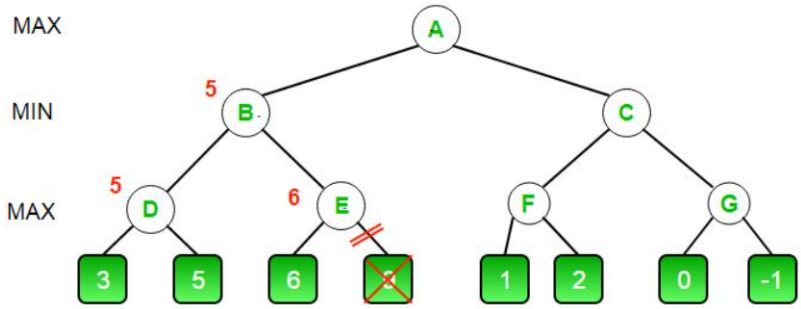


Figura 3.6: E se uită la copilul său din stânga, care este 6. La E, α = max($-\infty$, 6), adică 6. Aici condiția devine adevărată. β este 5 și α este 6. Deci $\beta \leq \alpha$ este adevărat. Prin urmare, se oprește și E returnează 6 la B. Observați cum nu contează care este valoarea copilului din dreapta lui E. Ar putea fi $+\infty$ sau $-\infty$, tot nu ar conta. Nici măcar nu a trebuit să ne uităm la el, pentru că minimizatorului i s-a garantat o valoare de 5 sau mai mică. Deci, de îndată ce maximizatorul a văzut 6, el știa că minimizatorul nu va veni niciodată în această direcție, deoarece poate obține un 5 pe partea stângă a lui B. Astfel, nu a trebuit să ne uităm la acel 9 și am economisit timp de calcul.

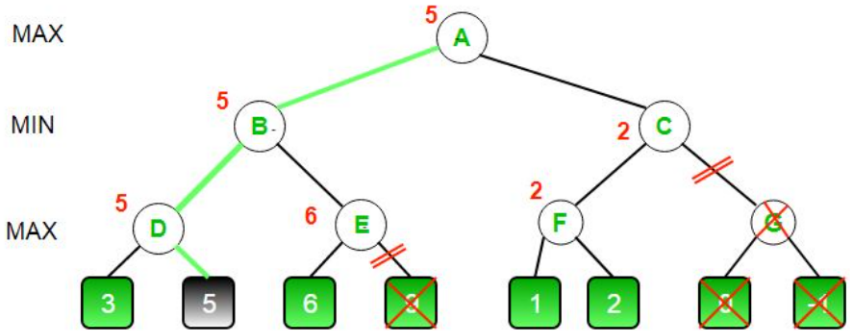


Figura 3.7: B returnează 5 lui A. La A, α = max($-\infty$, 5), adică 5. Acum, maximizatorului i se garantează o valoare de 5 sau mai mare. A apelează acum C pentru a vedea dacă poate obține o valoare mai mare decât 5. F se uită la copilul său din dreapta, care este 2. Prin urmare, cea mai bună valoare a acestui nod este 2. α rămâne tot 5. F returnează o valoare de 2 lui C. La C, β = min($+\infty$, 2). Condiția $\beta \leq \alpha$ devine adevărată deoarece β = 2 și α = 5. Deci se întrerupe și nu este necesar să se calculeze întregul subarbore al lui G.

Capitolul 4

Realizarea aplicației

Acest capitol explică în detaliu procesul de creare a aplicației. Se discută despre arhitectura aplicației, tehnologiile utilizate, diagramele de clase ale aplicației, precum și despre integrarea Stockfish în aplicație.

4.1 Arhitectura aplicației

Arhitectura aplicației este gândită într-un mod eficient, respectând principiile SOLID care stau la baza dezvoltării software de calitate. Separarea frontendului de backend asigură o claritate și o modularitate sporită a codului, făcând aplicația mai ușor de dezvoltat, de testat și de întreținut. Acest lucru este esențial în dezvoltarea unei aplicații scalabile și durabile. În plus, separarea componentelor în funcție de logica necesară respectă principiul responsabilității unice, fiecare componentă având un scop clar și bine definit.

Organizarea fișierelor și a directoarelor respectă convențiile de numire, facilitând navigarea și înțelegerea structurii proiectului. Denumirile respectă convenția directoarelor începând cu literă mică și fișiere cu literă mare, cum se poate observa în figura 4.1. Faptul că fișierele pot conține mai multe componente cu logică similară îmbunătățește coeziunea și reduce cuplarea, în conformitate cu principiile SOLID.

Utilizarea TypeScript este o alegere excelentă pentru menținerea codului sustenabil și scalabil. Acesta oferă un nivel suplimentar de securitate în timpul dezvoltării, permițând detectarea mai multor tipuri de erori înainte de a rula codul.

Pentru funcționalitățile de chat și cameră, am ales să utilizăm Socket.IO și WebRTC Peer. Socket.IO permite comunicarea în timp real între client și server, fiind ideal pentru funcționalitatea de chat. Pe de altă parte, WebRTC Peer permite transmiterea video și audio direct între utilizatori, fără a necesita un server intermediar, ceea ce face ca streamingul video să fie mai eficient și mai rapid.

Starea fiecărui jucător în joc este gestionată de Redux Store, care stochează informații

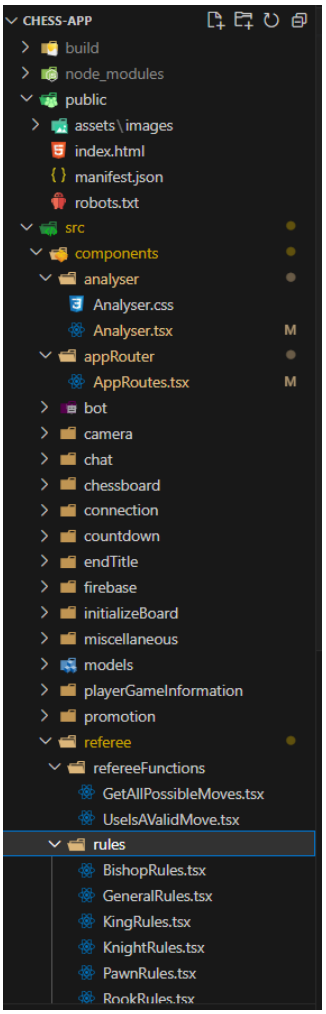


Figura 4.1: Structura fișierelor și directoarelor aplicației.

```tsx
export interface PlayerState {
  inCheck: boolean;
  kingMoved: boolean;
  leftRookMoved: boolean;
  rightRookMoved: boolean;
  kingPosition: SerializedPosition;
  knightCount: number;
  lightBishopCount: number;
  piecesCount: number;
}

const initialState: PlayerState = {
  inCheck: false,
  kingMoved: false,
  leftRookMoved: false,
  rightRookMoved: false,
  kingPosition: serializePosition(new Position(0, 4)),
  knightCount: 2,
  lightBishopCount: 1,
  piecesCount: 16,
};

export const whitePlayerSlice = createSlice({
  name: "whitePlayer",
  initialState,
  reducers: {
    setWhiteInCheck: (state, action: PayloadAction<boolean>) => {
      state.inCheck = action.payload;
    },
    setWhiteKingMoved: (state, action: PayloadAction<boolean>) => {
      state.kingMoved = action.payload;
    },
    setWhiteLeftRookMoved: (state, action: PayloadAction<boolean>) => {
      state.leftRookMoved = action.payload;
    },
    setWhiteRightRookMoved: (state, action: PayloadAction<boolean>) => {
      state.rightRookMoved = action.payload;
    },
    setWhiteKingPosition: (
      state,
      action: PayloadAction<SerializedPosition>
    ) => {
      state.kingPosition = action.payload;
```

Figura 4.2: Fișierul whitePlayer-Slice și state-ul.

precum dacă jucătorul este în șah (inCheck), dacă regele s-a mutat (kingMoved, ajută la deteminarea valabilității castelului) sau turnurile (leftRookMoved, rightRookMoved), poziția regelui (kingPosition), numărul total de piese (piecesCount, pentru a determina daca e egalitate) numărul de cai (knightCount), numărul de nebuni de pe pătrățele albe (lightBishopCount) și temporizatorul (timer). Acest lucru se poate vedea în whitePlayerSlice din figura 4.2 Această structură asigură o gestionare eficientă și flexibilă a stării jocului.

4.2 Tehnologii utilizate

În acest capitol vom prezenta tehnologiile folosite în dezvoltarea aplicației de șah.

4.2.1 React

React este o bibliotecă JavaScript care se concentrează pe construirea interfețelor de utilizator dinamice și interactive. Ceea ce face React special este abordarea sa de componentizare, care împarte interfața de utilizator în componente independente și reutilizabile, ceea ce face ca codul să fie mai ușor de gestionat și de dezvoltat în timp.

O altă caracteristică importantă a React este Virtual DOM, care este o reprezentare în memorie a arborelui DOM (Document Object Model). Pentru a înțelege mai bine vizitați [Fac23]. Aceasta face posibilă actualizarea rapidă a interfeței de utilizator, fără a fi nevoie să se actualizeze întreaga pagină web. În plus, React oferă o mare flexibilitate și scalabilitate, făcându-l potrivit pentru proiecte de diferite dimensiuni și complexități.

Una dintre principalele motive pentru care am ales React pentru aplicația noastră web de șah este faptul că este bine adaptată pentru aplicații cu interfață de utilizator complexă, cum este dezvoltarea unei aplicații de șah. Poate fi considerata astfel, deoarece implică o interfață de utilizator cu o mulțime de interacțiuni, animații și stări diferite ale jocului. React este potrivit pentru acest tip de aplicație datorită performanței sale bune și a abilității de a actualiza rapid interfața de utilizator în timp real, rezultând într-o interacțiune mai ușoara a utilizatorului.

În plus, React are o comunitate mare și activă, cu o mulțime de resurse și instrumente disponibile, ceea ce face procesul de dezvoltare mai ușor și mai eficient. În concluzie, alegerea React pentru aplicația noastră web de șah este o decizie excelentă, datorită performanței bune, flexibilității și scalabilității sale, precum și comunității și resurselor disponibile.

4.2.2 Node.js

Am ales să folosim Node.js ca platformă de backend pentru aplicația noastră de șah deoarece aceasta oferă o performanță ridicată și o scalabilitate excelentă. De asemenea, am ales Node.js datorită faptului că aceasta este construită pe motorul JavaScript, astfel că am putut utiliza aceeași limbajă de programare atât pentru partea de frontend, cât și pentru cea de backend.

Node.js este că este foarte bine integrată cu React, astfel, am putut crea o aplicație coerentă și bine structurată, care oferă o experiență de utilizare plăcută și fluidă, vezi [Nod23]. În plus, Node.js oferă un set de biblioteci și unelte puternice pentru a construi aplicații web, ceea ce ne-a permis să implementăm funcționalitățile complexe ale aplicației noastre de șah într-un mod eficient și ușor de întreținut.

4.2.3 Firebase

Am ales platforma Firebase pentru înregistrarea în aplicație și gestionarea bazei mele de date, din mai multe motive fundamentale. În primul rând, Firebase oferă o soluție cuprinzătoare pentru gestionarea autentificării utilizatorilor, permițând implementarea ușoară a funcționalităților de înregistrare și autentificare prin intermediul API-ului lor intuitiv. Aceasta a facilitat dezvoltarea rapidă și eficientă a componentei de autentificare, economisind timp și resurse de dezvoltare.

În al doilea rând, Firebase furnizează o bază de date în timp real, care se potrivește perfect nevoilor unei aplicații de șah. Aceasta permite actualizarea instantanee a informațiilor și sincronizarea în timp real între toți utilizatorii, ceea ce este esențial într-un mediu de joc interactiv. Baza de date în timp real oferă, de asemenea, un nivel ridicat de scalabilitate, permițând gestionarea fără probleme a creșterii numărului de utilizatori și a volumului de date.

În plus, Firebase oferă securitate puternică și confidențialitate a datelor, ceea ce este un aspect crucial pentru orice aplicație care implică autentificare și stocare a informațiilor personale ale utilizatorilor. Beneficiind de autentificare cu factori multipli și de măsuri avansate de securitate, precum autentificarea tokenilor și verificarea IP-urilor, Firebase asigură protecția și integritatea datelor utilizatorilor.

De asemenea, Firebase vine cu o gamă largă de instrumente și servicii suplimentare, cum ar fi analiticele și notificările push, care pot îmbunătăți experiența utilizatorului și funcționalitățile aplicației mele de șah.

În concluzie, alegerea Firebase pentru înregistrarea utilizatorilor și baza de date în cadrul aplicației mele de șah a fost justificată de gama sa completă de funcționalități, de securitatea și confidențialitatea oferite, precum și de abilitatea de a gestiona în mod eficient și scalabil autentificarea utilizatorilor și volumul de date. Mai multe informații se pot găsi la [Fir23].

4.2.4 Tensorflow

Tensorflow este o bibliotecă de învățare automată open-source, dezvoltată de Google, care oferă un ecosistem complet de instrumente, biblioteci și resurse comunitare. A fost ales pentru dezvoltarea modelului CNN datorită unei serii de avantaje semnificative pe care le oferă.

- Flexibilitate: Tensorflow permite definirea și instruirea unei varietăți largi de modele de învățare automată, inclusiv CNN. Oferă o gamă largă de operații și funcții care sunt esențiale pentru crearea și instruirea acestor modele.
- Scalabilitate: Tensorflow este capabil să lucreze cu o gamă largă de configurații

hardware, inclusiv GPU-uri și TPU-uri, ceea ce îl face o opțiune excelentă pentru procesarea eficientă a datelor în rețelele neuronale convoluționale.

- Comunitatea largă: Întrucât este o platformă open-source, Tensorflow are o comunitate largă de dezvoltatori și utilizatori care împărtășesc cod, idei și cele mai bune practici. Acest lucru îmbunătățește suportul și permite rezolvarea rapidă a problemelor.
- Instrumente de vizualizare: Tensorflow vine cu TensorBoard, un instrument de vizualizare care ajută la înțelegerea, depistarea și optimizarea proceselor de învățare.

În concluzie, utilizarea Tensorflow pentru CNN în aplicația noastră web de șah a permis o flexibilitate crescută, scalabilitate, suportul unei comunități largi și instrumente de vizualizare eficiente, facilitând astfel dezvoltarea, instruirea și optimizarea modelului. Poți afla mai multe aici [Ten23].

4.3 Diagramele de clase ale aplicației

4.3.1 Diagrama de clase frontend

Diagrama de clase a interfeței grafice a aplicației poate fi vizualizată în figura 4.3. Acesta este nucleul construcției, ea oferind o privire detaliată asupra modului în care elementele frontend sunt interconectate.

Index este fișierul principal al aplicației, acesta reprezintă punctul de pornire al programului. În acest fișier, se randează App, care include un wrapper cu Redux store. Redux este utilizat pentru a păstra codul structurat și ordonat, acesta fiind un principiu esențial al programării React eficiente și durabile.

Redux store este partea centrală a aplicației, având diverse slice-uri precum:

- WhitePlayer.
- BlackPlayer.
- MyGameSlice.
- AnalyseSlice.
- PageSlice.

WhitePlayerSlice și BlackPlayerSlice au o structură similară și conțin informații esențiale despre starea jocului: tura jucătorului, tura stânga mutată, tura dreapta

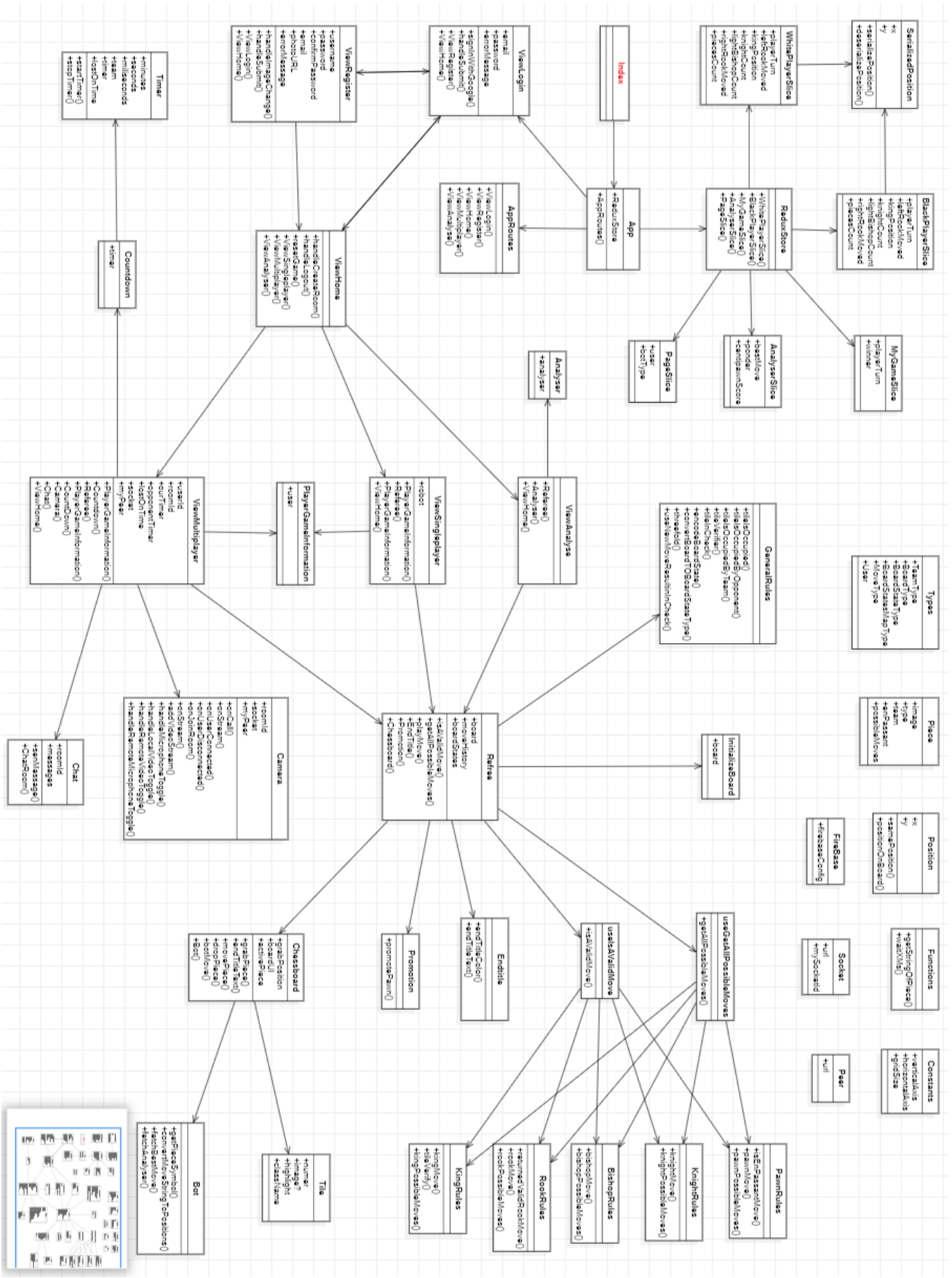


Figura 4.3: Diagrama de clase a frontend-ului.

mutată, poziția regelui, calul mutat, contorul pentru nebunul alb, și contorul pieselor. Pentru a stoca o Poziție (clasă proprie) în Redux, este necesară serializarea ei, în acest sens, am creat o clasă exclusivă pentru serializare si deserializare.

MyGameSlice este un alt segment important al stării de joc, păstrând tura jucătorului și identitatea câștigătorului. PageSlice se ocupă cu gestionarea datelor utilizatorului curent conectat și a tipului de bot selectat pentru jocul în modul singleplayer. Acesta permite personalizarea și adaptarea experienței de joc pentru fiecare utilizator în parte. În final, AnalyserSlice contine informații cruciale legate de învatarea jocului: cea mai bună mutare, mutarea ponder și avantajul în scor. Acesta este esențial pentru a permite analiza strategică și planificarea mișcărilor viitoare.

App generează componenta AppRoutes, în care sunt definite toate rutele aplicatiei:

1. ViewLogin.
2. ViewRegister.
3. ViewHome.
4. ViewSingleplayer.
5. ViewMultiplayer.
6. ViewAnalyse.

Acesta este nodul central al navigării în aplicație, asigurând o tranzitare fluidă între diferitele segmente ale acesteia.

Pagina de deschidere a aplicației este reprezentată de ViewLogin, unde utilizatorul se poate întâlni cu două forme de autentificare: una prin introducerea manuală a adresei de e-mail și a parolei, și una prin autentificare cu Google. În cazul în care apar erori de autentificare, acestea sunt afișate în mod clar utilizatorului prin intermediul câmpului errorMessage, facilitând astfel înțelegerea problemei și corectarea acesteia.

În situația în care utilizatorul nu deține deja un cont, are opțiunea de a-și crea unul prin intermediul înregistrării cu Google sau accesând ViewRegister. Aici, el este invitat să introducă datele necesare înregistrării, și anume:

- Numele de utilizator.
- Parola (și confirmarea acesteia).
- Adresa de e-mail.
- Imagine opțională.

Ca și în cazul autentificării, orice erori apărute sunt comunicate clar utilizatorului.

Odată autentificat, utilizatorul este redirecționat către ViewHome. Acesta reprezintă o pagină de start, de unde acesta poate alege una dintre cele trei View-uri sau se poate deconecta. În cazul în care alege să joace cu un prieten, este disponibilă funcția handleCreateRoom(), care creează o cameră de joc dedicată celor doi jucători. O altă funcție esențială este resetGame(), care are rolul de a reseta toate datele din Redux.

Primul View, dedicat jocului în modul singleplayer, include un adversar AI, fie personalizat (robot de convoluție), fie unul predefinit (stockfish). Utilizatorul are libertatea de a-și alege dificultatea jocului. În cadrul acestei pagini, sunt afișate datele despre utilizator (nume, imagine), un arbitru, precum și opțiunea de a ieși din meci, care va rezulta într-o pierdere instantă.

Arbitrul nu doar că arbitrează jocul, dar îndeplinește și rolul de tablă de șah. El gestionează tabla de joc, ține evidența istoricului de mutări și a stărilor tablei (util pentru verificarea regulii ThreeFoldRepetition). De asemenea, arbitrul are capacitatea de a valida dacă o mutare este corectă, de a genera toate pozițiile posibile pentru o anumită piesă sau jucător, și de a efectua o mutare. Când un jucător câștigă, se afișează un dialog (modalul EndTitle) care afișează cuvintele:

- Win. → Verde.
- Lose. → Rosu.
- Draw. → Galben.

în funcție de rezultatul meciului. În situația în care un pion ajunge pe ultimul rank, este afișat modalul Promotion, care permite utilizatorului să aleagă piesa în care dorește să promoveze pionul.

În arhitectura aplicației există mai multe fișiere destinate regulilor de joc, printre acestea regăsindu-se fișierul GeneralRules. Acesta cuprinde o suită de funcții care au responsabilitatea de a verifica diverse stări ale jocului: dacă anumite căsuțe sunt ocupate sau atacate, dacă o anumită mutare rezultă în șah, precum și funcții ce se ocupă de conversia tablei de joc pentru a facilita gestionarea acesteia. De asemenea, există reguli separate ce permit verificarea validității unei mutări de:

- Pion.
- Cal.
- Nebun.
- Tură.

- Regină.
- Rege.

sau obținerea tuturor pozițiilor posibile pentru o anumită piesă sau jucător.

Tabla de șah, elementul central al jocului, dispune de trei funcții principale: grabPiece(), care se activează când o piesă este ridicată, movePiece(), care coordonează deplasarea piesei pe tabla de joc, și dropPiece(), care se activează atunci când piesa este lăsată pe o căsuță, fixându-se automat în centrul acesteia. În acest context, tabla de șah găzduiește și algoritmul bot-ului nostru, care își poate executa mutarea odată ce aceasta a fost decisă. Bot-ul dispune de mai multe funcții ajutătoare pentru conversie și pentru comunicarea cu backend-ul, astfel încât să poată determina cea mai bună mutare în contextul jocului sau analizei meciului. Tabla de șah generează 64 de căsuțe (Tile-uri), fiecare căsuță conținând indexul său, imaginea piesei (dacă există una plasată pe căsuță), precum și o funcție de evidențiere atunci când o piesă este apăsată (pentru a vizualiza toate mutările posibile). În funcție de poziția sa, fiecare căsuță este colorată în mod distinct.

ViewMultiplayer este un alt element important al aplicației, care include un userId pentru inițierea serverului WebRTC necesar comunicării cu camera, un roomId pentru jocul cu un prieten (dacă este cazul), două cronometre (timere) pentru fiecare jucător, funcția lostOnTime care se activează la sfârșitul timpului alocat meciului, precum și elementele de socket și peer ale WebRTC. Similar cu ViewSingleplayer, sunt afișate informațiile despre jucători și arbitrul cu tabla, dar aici se adaugă și două cronometre, o funcție de videochat și un chat. Cronometrele pot fi personalizate la începutul meciului și sunt afișate în componenta Countdown.

În situația în care utilizatorii optează pentru funcționalitatea de Camera, conexiunea se realizează prin intermediul librăriei socket.io, care operează pe protocolul TCP. După stabilirea conexiunii, transmisiunea video și audio se realizează live, prin protocolul UDP. Utilizatorul are la dispoziție mai multe opțiuni de personalizare a experienței, precum posibilitatea de a se pune pe mute, de a-și ascunde camera, de a dezactiva camera adversarului sau de a-l pune pe mute pe adversar. Fiecare dintre aceste opțiuni dispune de un buton cu handler propriu. În manieră similară funcționalității de Camera, modulul de Chat dispune de un roomId propriu și de o listă cu mesajele din sesiunea curentă.

Ultimul view este cel de Analyse. Asemănător celorlalte view-uri, acesta conține un arbitru și o tablă de șah, dar în plus dispune și de o componentă specială care cuprinde un Analizor. Acesta din urmă comunică prin intermediul unui API cu serverul backend pentru a obține bestMove, ponderMove și CentipawnScore.

Întrucât menținerea unui cod curat și ușor de înțeles este o prioritate, s-a creat un fișier separat care conține definițiile pentru tipuri, printre care:

- TeamType.
- BoardType.
- BoardStateType.
- BoardStatesMapType.
- MoveType.
- User.

Clasa Piece include:

- Imaginea piesei.
- Tipul piesei.
- Echipa piesei.
- Indicator dacă piesa poate fi capturată prin en-passant
- Lista cu toate mutările posibile.

Clasa Position deține două coordonate, x și y, și dispune de funcții care pot verifica dacă o poziție este identică cu poziția curentă sau dacă piesa se află pe tablă. Fișierul Functions cuprinde funcții ajutătoare, precum getStringOfPiece() sau waitXMs(). Aplicația dispune și de un fișier de Constante, unde sunt definite valori pentru verticalAxis, horizontalAxis și gridSize. Fișierele Firebase, Socket și Peer, sunt folosite pentru configurare.

4.3.2 Diagrama de clase backend

Diagrama de clasă a backendului, ilustrată în figura 4.4, reprezintă arhitectura logică internă a aplicației noastre web pentru șah.

Punctul de inițiere al aplicației este componenta App, care include setările pentru firebase(storageProxy), identificatorul socket-ului și componenta Api. Componenta App este responsabilă pentru adăugarea tuturor middleware-urilor esențiale necesare pentru funcționarea corectă a aplicației. Aici se includ:

- Cors().
- Express.json().
- Express.urlencoded().
- BodyParser.urlencoded().

- BodyParser.json().
- StorageProxy().

După configurarea acestora, aplicația începe să ruleze pe portul 3001.

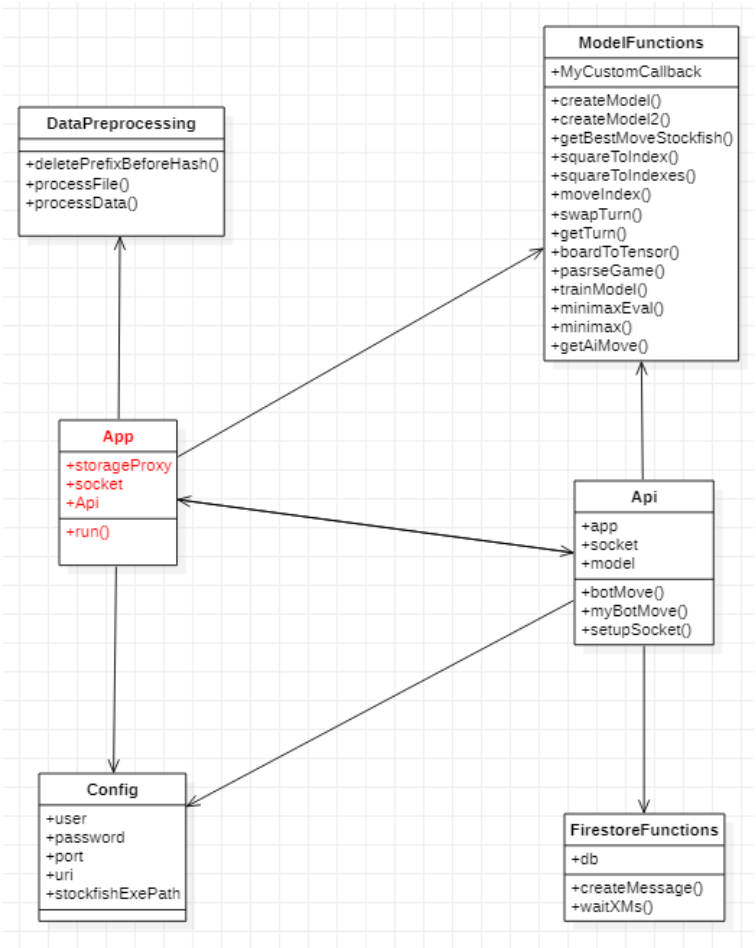


Figura 4.4: Diagrama de clase a backend-ului.

Fișierul Config găzduiește constantele care sunt vitale pentru operațiile aplicației, inclusiv:

- Datele utilizatorului.
- Parola.
- Portul.
- URI-ul.
- Calea de acces către executabilul Stockfish - stockfishExePath.

Aceste constante sunt distribuite în întreaga aplicație pentru a asigura o funcționare coerentă și eficientă.

Componenta Api este un alt element esențial din diagrama noastră de clase. Ea găzduiește instanța aplicației, identificatorul socket-ului și modelul bot-ului TensorFlow. De asemenea, Api este responsabil pentru gestionarea metodelor de API REST.

Prima metodă, botMove(), creează un proces care utilizează Stockfish bazat pe FEN-ul jocului curent. Această metodă returnează bestMove (cea mai bună mișcare), ponderMove (mișcarea ponderată) și centipawnScore (scorul centipawn) frontend-ului.

A doua metodă, myBotMove(), se bazează pe librăria TensorFlow și implementează un model de rețea neuronală convoluțională (CNN) care joacă șah, returnând frontend-ului cea mai bună mișcare calculată.

Ultima metodă, setupSocket(), asigură:

- Menținerea conexiunii socket.
- Intrarea în camera de joc.
- Furnizarea fluxului video.
- Răspunsul la cererea video.
- Notificarea automată a unui mesaj nou sau a situației în care un utilizator scrie.

Componenta ModelFunctions reprezintă o complexitate crescută datorită responsabilităților bazate pe inteligență artificială. Aceasta este instruită pentru crearea și antrenarea modelelor cu ajutorul mai multor modalități și a multiplelor baze de date.

Unul dintre metodele de învățare implementate implică selectarea unei poziții aleatorii și utilizarea funcției getBestMoveStockfish(). Această funcție extrage cea mai bună mutare și o folosește pentru antrenarea modelului, împreună cu scorul asociat mutării respective.

Alternativ, o altă metodă mai eficientă de antrenare implică utilizarea unei baze de date care conține deja pozițiile și scorurile centipawn deja calculate. Această metodă permite o optimizare semnificativă a timpului de antrenare, permițând modelului să acceseze direct informații preprocesate și să învețe din acestea.

În procesul de antrenare a modelului, este esențial să convertim tabla de șah într-un format pe care TensorFlow îl poate înțelege. Pentru aceasta, am implementat funcția boardToTensor(), care transformă tabla într-un tensor de 8x8x12. Detalii suplimentare despre modul în care tabla este reprezentată se pot găsi în capitolul 3.2.

Funcția trainModel() este cea care se ocupă efectiv de antrenarea modelului. Aceasta parcurge baza de date, procesează informațiile și începe antrenarea. În acest proces, sunt utilizate atât datele preprocesate, cât și cele rezultate în urma funcției

getBestMoveStockfish(), contribuind astfel la dezvoltarea unui model robust și eficient.

Pe măsură ce avansăm prin ciclurile de antrenare ale modelului, ajungem la un punct în care, la un număr predefinit de iterații, antrenarea intră în stadiul de învățare. Aici, putem seta o serie de parametri importanți, cum ar fi numărul de elemente dintr-un batch și numărul de epoci pentru antrenare. La finalizarea acestui proces, rezultatele învățării sunt păstrate într-un fișier pentru referință și utilizare ulterioară.

După ce procesul de învățare este complet, putem apela funcția getAiMove() pentru a utiliza modelul și a obține cea mai bună mișcare în orice situație de joc. Pentru a realiza acest lucru, getAiMove() se bazează pe un algoritm Minmax îmbunătățit cu Alfa Beta Pruning, pentru a determina cea mai optimă mișcare în timp record. Pentru detalii suplimentare despre această metodă, consultați capitolul teoretic 3.4.1.

Componenta ModelFunctions include și o serie de funcții auxiliare care sunt esențiale în funcționarea sistemului nostru. Acestea includ:

- SquareToIndex().
- SquareToIndexes().
- MoveIndex().
- SwapTurn().
- GetTurn().
- ParseGame().
- Minmax().
- MinmaxEval().

Fișierul DataPreprocessing joacă un rol crucial în prelucrarea și prepararea datelor pentru antrenarea modelului. Aceasta include trei funcții principale: deletePrefixBeforeHash(), proccessFile(), processData().

Pentru configurarea și utilizarea Firebase, ne bazăm pe setul de funcții definite în FirebaseFunctions.

În final, backendul are și responsabilitatea de a iniția serverul Peer serverless, asigurând astfel comunicarea fluidă între utilizatori și îmbunătățind performanța aplicației.

4.4 Integrarea Stockfish în aplicație

Implementarea Stockfish CLI în aplicația mea a reprezentat o alegere esențială pentru a oferi o experiența de șah de înaltă calitate. Stockfish este recunoscut ca fiind cel mai puternic motor de șah din lume, iar faptul că este open source îmi oferă acces liber la un instrument de înaltă performanță, dar și posibilitatea de a-l personaliza conform nevoilor aplicației mele. Una dintre cele mai importante caracteristici ale Stockfish este capacitatea de a seta parametrii care influențează dificultatea botului. Acest lucru îi permite jucătorului să se confrunte cu o varietate de nivele de dificultate, oferindu-i posibilitatea de a juca singleplayer la diferite rating-uri.

În ceea ce privește arhitectura tehnologică, am optat pentru o abordare bazată pe request-uri POST pentru a obține cea mai bună mișcare de la server. După ce serverul calculează și returnează mișcarea, aceasta este trimisă către frontend-ul aplicației mele, unde este analizată și executată. Acest flux de lucru asigură un răspuns rapid și eficient, oferind în același timp o experiență de joc fluidă și plăcută pentru utilizator.

Capitolul 5

Utilizarea aplicației

În acest capitol se ilustrează experiența utilizatorilor cu aplicația. Se prezintă o diagramă de cazuri de utilizare și explică în amănunt procedura de înregistrare și autentificare, precum și modul de navigare în aplicație și funcționalitățile paginii principale.

5.1 Diagrama de cazuri

Diagrama de cazuri de utilizare, prezentată în figura 5.1, prezintă trei actori principali, și anume: Persoana, Utilizatorul și Sistemul de Gestionare a Bazelor de Date (SGBD) Firebase.

Primul actor, Persoana, reprezintă orice vizitator care poate accesa pagina web a aplicației. Pentru a beneficia de funcționalitățile complete ale aplicației, persoana trebuie să se autentifice. Autentificarea este un proces necesar pentru a asigura o experiență personalizată și sigură pentru fiecare utilizator. De asemenea, aceasta facilitează gestionarea mai eficientă a interacțiunilor utilizatorilor în cadrul aplicației.

Odată ce procesul de autentificare a fost completat cu succes, Persoana devine un Utilizator autentificat. Acesta are acces la meniul principal al aplicației, un hub central care oferă multiple opțiuni de interacțiune. Utilizatorul este prezentat cu patru posibilități distincte:

1. Joaca în mod singleplayer, unde se poate antrena împotriva unui adversar controlat de AI.
2. Joaca în mod multiplayer, ce permite interacțiunea cu alți utilizatori în timp real.
3. Joaca împotriva unui prieten, o opțiune ce permite jocul direct cu un utilizator cunoscut.

4. Analizează tabla, o caracteristică educațională ce permite analizarea jocurilor anterioare și strategiilor de șah.

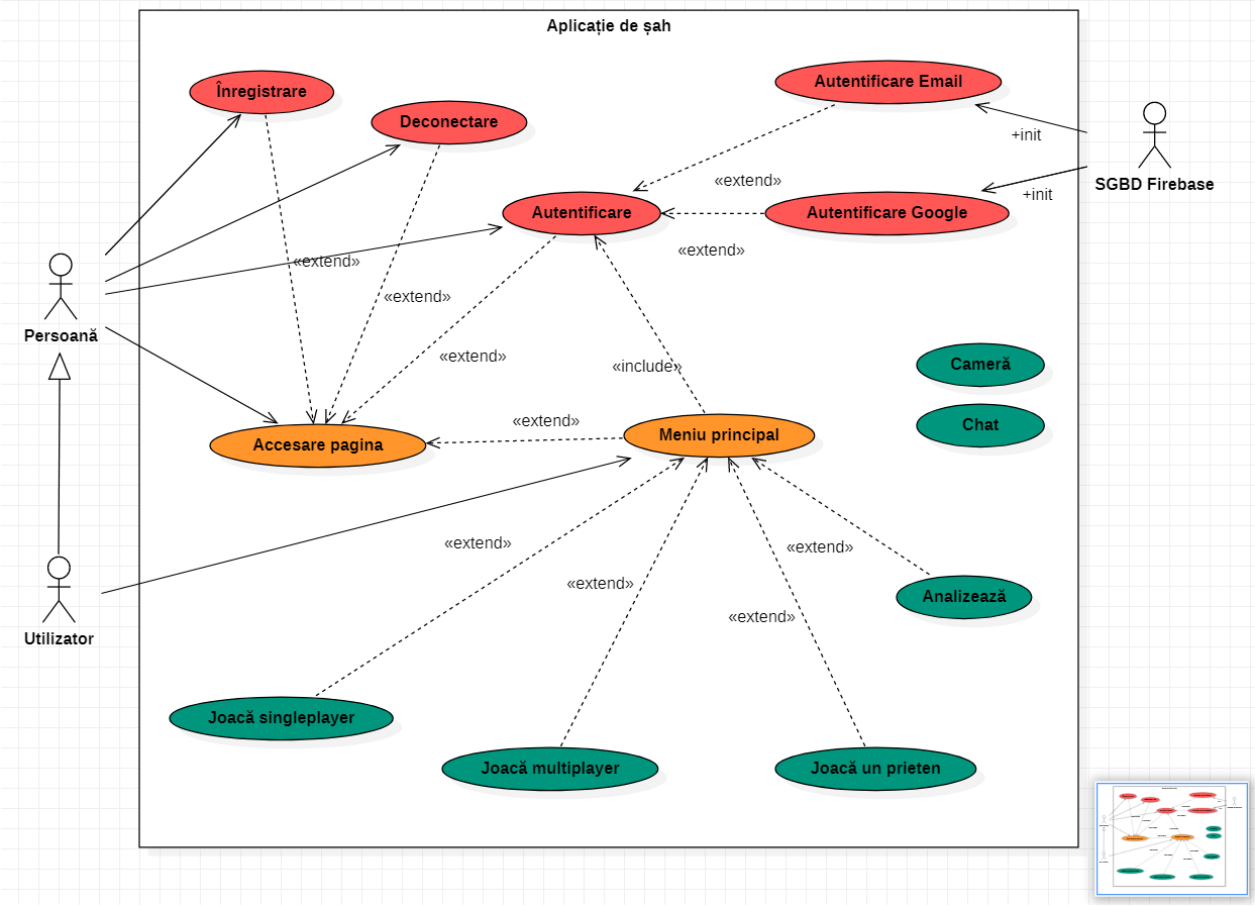


Figura 5.1: Diagrama de utilizare a aplicației

În modurile de joc multiplayer și cu un prieten, există posibilitatea comunicării prin intermediul funcționalităților video, vocale și de chat, sporind astfel imersiunea și interacțiunea dintre utilizatori.

Al treilea actor, SGBD Firebase, joacă un rol esențial în gestionarea și asigurarea securității datelor utilizatorilor. Acest sistem de gestionare a bazelor de date este integrat cu procesul de autentificare, fie că este vorba de autentificare prin email sau prin Google. SGBD Firebase asigură stocarea și gestionarea eficientă a datelor, garantând în același timp securitatea și confidențialitatea informațiilor utilizatorilor.

5.2 Înregistrare

Pentru a utiliza aplicația de șah, trebuie să creezi un cont. Procesul de înregistrare este simplu și rapid. În primul rând, accesează pagina de înregistrare a aplicației,

detaliat în figura 5.2. Acolo vei găsi un formular care conține câmpurile obligatorii precum:

- Nume utilizator.
- Parola.
- Confirmare parolă.
- Email.
- Opțional, poți să-ți adaugi o imagine de profil din calculatorul tău, deschizând windows explorer, cum se poate vedea în imaginea 5.3.

După completarea câmpurilor, apasă butonul "Register" pentru a-ți crea contul. Dacă utilizatorul introduce date invalide sau apar erori în timpul procesului de înregistrare, va primi un mesaj de eroare care te va informa despre problema întâmpinată. Dacă totul decurge bine, vei fi redirecționat către pagina principală a aplicației.

Pentru a ușura procesul de înregistrare, am implementat și opțiunea de înregistrare utilizând contul Google. Dacă dorești să te înregistrezi folosind Google, pur și simplu apasă butonul "Sign in with Google" de pe pagina de înregistrare. Vei fi redirecționat către pagina de autentificare Google, unde poți selecta unul dintre conturile Google existente sau poți crea unul nou. Asemenea ca la înregistrarea prin email, după autentificare vei fi redirecționat la pagina principală.

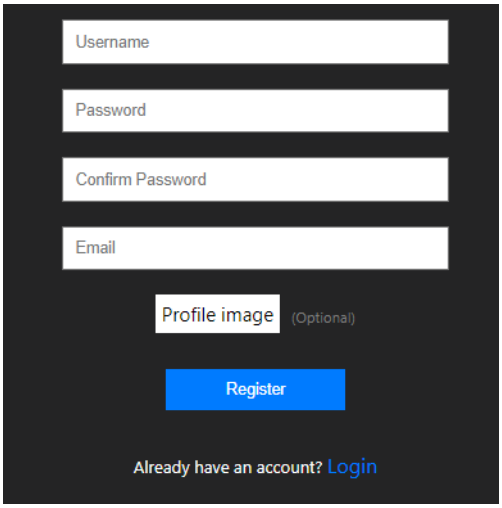


Figura 5.2: Vedere Register cu email.

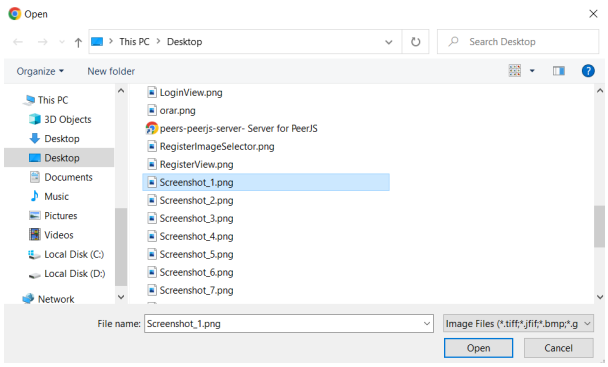


Figura 5.3: Vederea opțiunii de alegere imagine de profil.

5.3 Autentificare

Pentru a vă autentifica în aplicația de șah, completați adresa de email și parola asociată contului dvs. și apăsați butonul "Login", așa cum se poate vedea in imaginea

5.4.

Dacă aveți deja un cont de utilizator înregistrat și introduceți corect informațiile de autentificare, veți fi redirecționat către pagina principală a aplicației. În cazul în care informațiile de autentificare introduse sunt incorecte sau apare o eroare în timpul procesului de autentificare, veți primi un mesaj care vă va informa despre problema întâmpinată.

De asemenea, precum la înregistrare, puteți utiliza și opțiunea de autentificare prin intermediul contului Google, vizibil în figura 5.5, care prezintă aceeași pași ca și la înregistrare.

Dacă nu aveți încă un cont de utilizator, vă recomandăm să vă înregistrați prin intermediul opțiunii "Register".

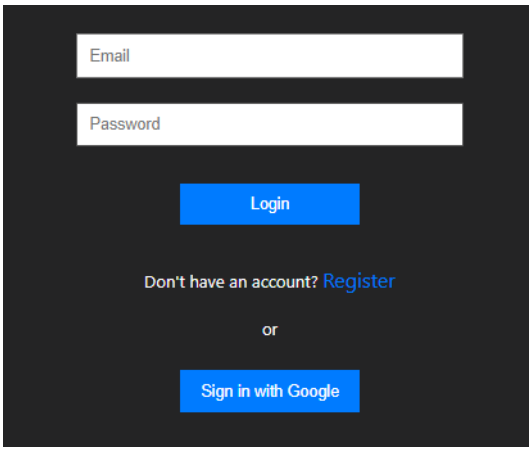


Figura 5.4: Vedere Login cu înregistrare email si Google.

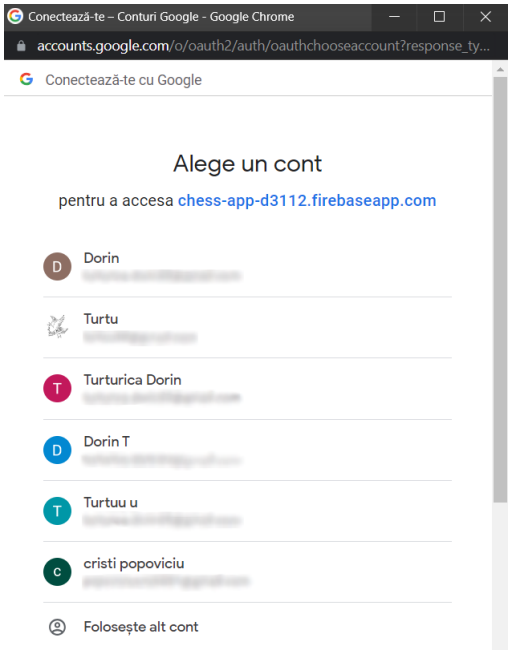


Figura 5.5: Autentificare/Înregistrare cu Google.

5.4 Pagina principală

După autentificarea cu succes, în partea dreaptă-sus a paginii se va afișa numele și imaginea utilizatorului pentru a confirma că utilizatorul s-a autentificat cu succes în aplicație, vizibil în figura 5.7. Cum este vizibil în figura 5.6, utilizatorul poate alege dintre:

1. Singleplayer.
2. Multiplayer.

3. Play a friend.
4. Analyse.

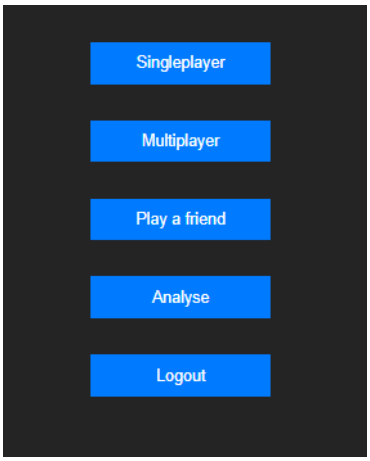


Figura 5.6: Vedere Meniului Principal.

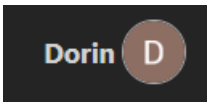


Figura 5.7: În coltul dreapta sus, date despre utilizatorul logat.

5.4.1 Singleplayer

Secțiunea de joc în modul singleplayer reprezintă o funcționalitate cheie a aplicației web, oferind o platformă de interacțiune directă dintre utilizator și inteligența artificială. În această configurație, jucătorul uman are oportunitatea de a confrunta un bot controlat de calculator, care imită strategii sofisticate de joc.

Utilizatorul are la dispoziție un set de opțiuni pentru a particulariza nivelul de dificultate al bot-ului, astfel încât să fie în măsură să-și adapteze provocarea la nivelul său de abilitate sau preferințe. Acest element de customizare sporește atractivitatea și accesibilitatea platformei pentru o gamă largă de jucători, de la începători la avansați.

Pentru a îmbogăți experiența de joc, utilizatorul are opțiunea de a alege între două tipuri de bot: unul bazat pe o rețea neuronală convoluțională (CNN), pe care am proiectat-o și instruit-o personal, și celălalt reprezentat de renumitul motor de șah, Stockfish. Fiecare dintre acestea vine cu setări distincte care permit controlul fin asupra nivelului de dificultate.

În cadrul mecanicii de joc, utilizatorii se confruntă într-o succesiune alternantă de mutări, în conformitate cu regulile standard de șah. Aplicația oferă ajutor vizual în alegerea mutărilor prin evidențierea tuturor opțiunilor legale disponibile după selectarea unei piese, incluzând:

- Mutările de bază.
- Rocada.

- EnPassant.
- Capturările.

Vezi figura 5.8 Această caracteristică funcționează în tandem cu mecanismul care restrânge opțiunile de mutare atunci când jucătorul este în șah, în conformitate cu regulile jocului.

În momentul încheierii partidei, utilizatorul este informat printr-un mesaj modal despre rezultatul jocului, fie că acesta a câștigat, a pierdut sau a remizat.

Figura 5.8: Vederea Singleplayer împotriva unui bot. După ce apăsăm regina putem vedea toate casuțele legale în care se poate deplasa. Piesele pe care le poate lua sunt plasate într-o căsuță pătratică neagră.

5.4.2 Multiplayer

Această funcționalitate oferă utilizatorului posibilitatea de a juca șah în mod multiplayer. Caracteristicile esențiale precum tabla de joc și respectarea regulilor de șah sunt la fel ca și în modulul singleplayer. Un punct crucial de diferențiere este însă adversarul, care, în acest context, este un om real.

O componentă cheie a experienței multiplayer este introducerea unui cronometru, cum se poate observa în imaginea 5.9. Setat înainte de începerea meciului. În

momentul în care timpul unui jucător expiră, acesta pierde partida, rezultatul fiind comunicat printr-un mesaj modal, la fel ca în modul singleplayer.

Pentru a îmbogăți și mai mult interacțiunea în timpul jocului, aplicația oferă o funcționalitate de video chat. Aceasta permite jucătorilor să se vadă reciproc în timp real, pentru a ne asigura ca oponentul nu trișează sau pentru a interacționa social cu inamicul, contribuind astfel la autenticitatea și intensitatea experienței. Camerele video sunt poziționate strategic în colțul din dreapta sus al ecranului, pentru a nu a deranja partida. Cea a adversarului e afișată predominant, pentru a distinge mai ușor mișcările sale. Camera jucătorului în sine este de asemenea disponibilă, permițându-i să își verifice propriul cadru de câte ori dorește.

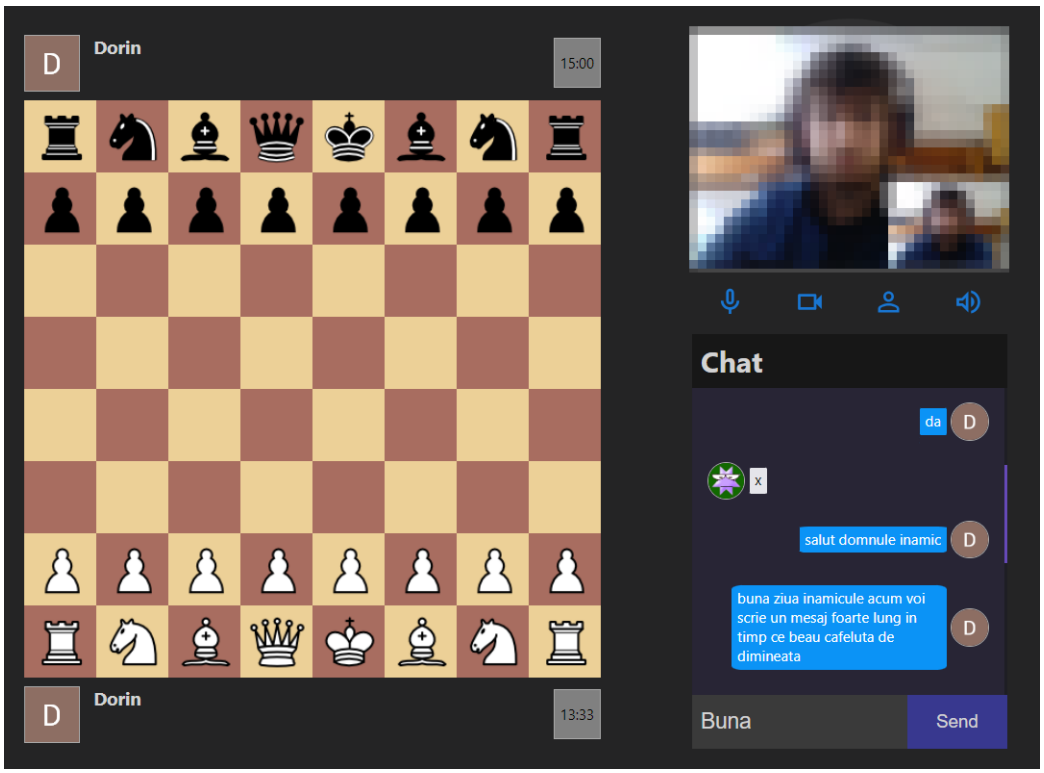


Figura 5.9: Vederea Multiplayer împotriva unui jucător aleator sau prieten. Se poate ca exista două camere, cea mare a inamicului, iar cea mică a noastră. Am blurat fața din motive de securitate. În cazul de față am folosit aceeași cameră a mea pentru exemplificare. Chatul se poate observa în partea din dreapta jos, populat cu mesaje.

Sub secțiunea de video se găsesc patru butoane de control:

1. Oprirea microfonului propriu.
2. Oprirea propriei camere.
3. Dezactivarea camerei adversarului.
4. Oprire sonor primit de inamic.

Acestea sunt însoțite de iconițe explicative, care indică starea pornită/oprită.

Ultima caracteristică notabilă a modului multiplayer este chatul. Am plasat aceasta sub butoanele de control, oferind un flux în timp real al comunicării dintre jucători. După introducerea unui mesaj, chatul se actualizează automat pentru amândoi, asigurând astfel un dialog fluent și o experiență multiplayer plăcută.

5.4.3 Play a friend

Această funcționalitate este identica cu cea a jocului multiplayer simplu, dar aici persoana poate să joacă cu un prieten. Un link unic pentru cameră este generat automat, iar utilizatorul poate trimite acesta prietenului său pentru a se alătura jocului.

5.4.4 Analyse

Pagina de analiză a aplicației noastre reprezintă o resursă esențială pentru învățarea și îmbunătățirea abilităților de joc la șah.

În primul rând, utilizatorul beneficiază de un sistem de recomandare a mișcărilor. Acesta furnizează cea mai bună mișcare posibilă în situația dată, calculată pe baza algoritmului implementat în motorul de joc Stockfish (vezi figura 5.10). Acest lucru poate ajuta jucătorii să își îmbunătățească abilitățile strategice, să învețe despre tactici avansate și să evite erorile comune.

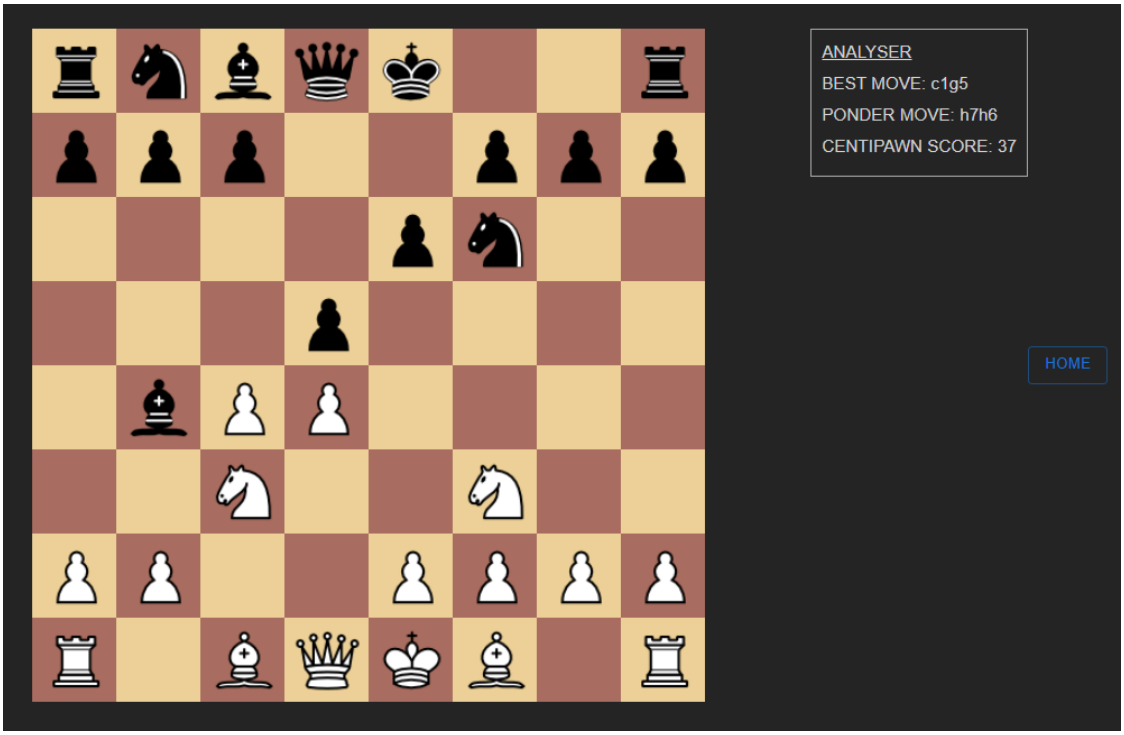


Figura 5.10: Vederea Analyse, împreuna cu analizatorul de pe dreapta.

Pe lângă recomandarea celei mai bune mișcări, utilizatorul primește și informații despre posibila reacție a adversarului, sub forma unei mutări ponderate. Astfel individul poate deprinde tactici anticipative.

Pentru a oferi o înțelegere mai profundă a situației de joc, sistemul furnizează și un scor centipawn, precum [Har23] sugerează la pasul 2. Acesta este un sistem de evaluare care atribuie o valoare numerică fiecărei piese de șah:

1. 100 - Pion.
2. 300 - Cal și nebun.
3. 500 - Tură.
4. 900 - Regină.
5. 9000 - Rege.

Aceste valori pot fi ajustate în funcție de alte aspecte ale jocului, după cum este detaliat în secțiunea 3.3. Scorul centipawn oferă o modalitate obiectivă de a evalua poziția pe tablă și de a cuantifica avantajul fiecărui jucător.

Capitolul 6

Concluzii și direcții viitoare de lucru

În lucrarea de licență prezentată ne-am propus să contribuim la dezvoltarea și îmbunătățirea experienței de joc de șah prin intermediul unei aplicații web interactive și personalizate. Acest obiectiv a fost realizat combinând tehnologiile web moderne cu principiile de inteligență artificială avansată.

Un aspect esențial al aplicației noastre este integrarea unui model de rețea neuronală convoluțională (CNN), care a fost antrenat și optimizat pentru a juca șah. Prin intermediul acestui model, am putut oferi utilizatorilor noștri un adversar și mentor inteligent, capabil să simuleze strategiile de joc ale grandmasterilor de șah. Acest lucru nu doar că îmbunătățește experiența de joc a utilizatorilor noștri, dar ajută și la accelerarea învățării șahului.

În cadrul fiecărui capitol am dovedit următoarele:

2. "Aspecte teoretice privind Rețelele Neuronale". Am prezentat principiile fundamentale ale rețelelor neuronale și aplicabilitatea lor în șah, explorând procesele de convoluție, filtrare și diferite tipuri de învățare automată.

3. "Aspecte teoretice despre șah". S-a focalizat pe analiza teoriei șahului, cu discuții privind evaluarea tablei de joc, reprezentarea acesteia și integrarea algoritmului Minimax îmbunătățit cu Alfa-Beta Pruning.

4. "Realizarea aplicației". Am detaliat procesul de dezvoltare a aplicației, acoperind arhitectura, tehnologiile utilizate, diagramele de clase și integrarea agentului Stockfish.

5. "Utilizarea aplicației". Am descris experiența utilizatorului cu aplicația, detaliind procesul de înregistrare și autentificare, navigarea prin aplicație și principalele funcționalități ale platformei.

În privința direcțiilor de viitor, există potențial exponențial de a extinde funcționalitățile aplicației, datorită codului scris de mână(folosind doar librării esențiale proiectării

ordonate), a arhitecturii structurate și a codului curat. Datorită faptului că am scris tabla și regulile manual, vom avea posibilitatea de a dezvolta moduri noi de joc din codul nostru existent, foarte ușor. De asemenea, putem introduce un sistem de clasificare pentru jucătorii multiplayer sau putem dezvolta un mod de instruire asistată de AI care să furnizeze sfaturi și strategii în timp real, pe măsură ce utilizatorii joacă.

În concluzie, această lucrare a atins obiectivele propuse, oferind un model convingător pentru dezvoltarea ulterioară a aplicațiilor de șah bazate pe AI.

În mod evident, domeniul șahului și inteligenței artificiale continuă să fie o zonă plină de oportunități pentru cercetare și inovație, iar această lucrare reprezintă un pas înainte în acest sens.

Bibliografie

[Cha21a] Antoine Champion. Dissecting stockfish part 1: In-depth look at a chess engine, Feb 2021.

[Cha21b] Antoine Champion. Dissecting stockfish part 2: In-depth look at a chess engine, May 2021.

[che18] Getting started programming a chess ai, Dec 2018.

[Che21] Chess.com. Top chess engines, 2021.

[Dat17] Datasnaek. Chess game dataset, 2017.

[DV16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv:1603.07285*, 2016.

[EWL99] Jens Eisert, Martin Wilkens, and Maciej Lewenstein. Quantum games and quantum strategies. *Phys. Rev. Lett.*, 83:3077–3080, Oct 1999.

[Fac23] Inc. Facebook. Introduction to react, 2023.

[Fir23] Firebase. Community | introduction to firebase, 2023.

[Gee17] GeeksforGeeks. Minimax algorithm in game theory | set 4 (alpha-beta pruning). 2017.

[Har23] Lauri Hartikka. A step-by-step guide to building a simple chess ai. 2023.

[Jan17] Katarzyna Janocha. On loss functions for deep neural networks in classification. *ar5iv preprint ar5iv:1702.05659*, 2017.

[Kag23] Kaggle. 3.5 million chess games, 2023.

[KJK16] Shaharyar Kamal, Ahmad Jalal, and Daijin Kim. Depth images-based human detection, tracking and activity recognition using spatiotemporal features and modified hmm. *Journal of Electrical Engineering & Technology*, 11(6):1857–1862, 2016.

[KM75] Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.

[MKT+22] Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119, 2022.

[Nod23] Node.js. Introduction to node.js, 2023.

[Pur21] Nihal Puram. Training a chess ai using tensorflow, 2021.

[Rus10] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[Sab17] Matthia Sabatelli. *Learning to Play Chess with Minimal Lookahead and Deep Value Neural Networks*. PhD thesis, Faculty of Science and Engineering, 2017.

[SBB+07] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844), 2007.

[Sec21] Digital Secrets. Creating a chess ai with tensorflow, 2021.

[Swa20] K E Swapna. Convolution neural network deep learning, 8 2020.

[Ten23] TensorFlow. Machine learning education | tensorflow, 2023.