

Lucrare de licență
Chess Snapshot - Recunoașterea stării în jocul de
șah

Autor: Tiberiu-Ioan Boșcan
Coordonator științific: Lect. Dr. Ioana Cristina Plajer

2024

Rezumat

Chess Snapshot este un proiect de viziune computerizată care își propune să automatizeze recunoașterea stării jocului de șah. Acest proiect explorează utilizarea tehniciilor de prelucrare a imaginilor și de învățare automată pentru a analiza pozițiile de pe tabla de șah din imagini și a identifica starea curentă a jocului. Prin folosirea tehniciilor moderne în domeniul viziunii computerizate, Chess Snapshot își propune să ofere o unealtă convenabilă pentru pasionații de șah pentru a analiza și a înregistra cu exactitate partidele desfășurate pe tablă.

Cuprins

1 Introducere	4
1.1 Scopul lucrării și motivația	4
1.2 Tehnici folosite pentru recunoașterea stării jocului	5
1.2.1 Detectia tablei de șah	5
1.2.2 Detectia pieselor	5
1.2.3 Implementari recente	5
1.3 Contribuții personale	6
1.3.1 Optimizarea eficienței algoritmilor din implementările anterioare	6
1.3.2 Utilizarea de unelte informaticice moderne	7
1.3.3 Îmbogățirea setului de date pentru antrenarea modelului de detecție a pieselor de șah	7
1.3.4 Dezvoltarea unei aplicații multi-platformă	7
1.4 Cerințe și specificații	8
1.4.1 Cerințe	8
1.4.2 Tratarea cerințelor	8
1.5 Prezentare succintă a aplicației	10
2 Tehnologii utilizate	11
2.1 Limbaje de programare	12
2.1.1 Python	12
2.1.2 Dart	12
2.2 Roboflow	13
2.3 Jupyter Notebook	14
2.4 OpenCV	14
2.5 Tensorflow & Keras	14
2.6 YOLOv8	15
2.7 Flask	16
2.8 Flutter	16
2.9 Alte instrumente	17
2.9.1 Matplotlib	17
2.9.2 Scikit-learn	18
2.9.3 GitHub	18
3 Arhitectură	20
3.1 Arhitectura detectorului stării jocului de șah	20
3.2 Arhitectura aplicației	21
4 Algoritmi de procesare de imagini folosiți	23
4.1 Egalizarea adaptivă a histogramelor (CLAHE)	23
4.2 Binarizarea folosind algoritmul Otsu	24
4.3 Filtrarea bilaterală	25

4.4	Detectarea marginilor folosind operatorul Canny	25
4.5	Detectarea liniilor folosind transformarea Hough	26
4.6	Transformarea perspectivă	27
5	Detectia tablei și identificarea stării jocului	29
5.1	Detectia tablei	29
5.1.1	Detectia liniilor drepte	30
5.1.2	Detectia punctelor de intersecție	34
5.1.3	Căutarea poziției tablei de șah	37
5.2	Detectia pieselor de șah	38
5.2.1	Set de date	38
5.2.2	Adnotare	38
5.2.3	Antrenare	39
5.3	Detectia stării jocului de șah	39
5.3.1	Utilizarea detectoarelor tablei și pieselor de șah	40
5.3.2	Postprocesare și validare	40
5.3.3	Generarea notării FEN	40
6	Aplicația tip multi-platformă	42
6.1	Aplicabilitatea aplicației Chess Snapshot ca soluție multi-platformă	42
6.2	Back-end-ul aplicației	43
6.2.1	Detectia stării jocului de șah	43
6.2.2	Determinarea unei mutări optime pentru o stare de joc	44
6.2.3	Publicarea API-ului pe internet	44
6.3	Front-end-ul aplicației	45
6.3.1	Pagina Detect	45
6.3.2	Bara de acțiuni a paginii Detect	45
6.3.3	Lista de funcționalități a paginii Detect	46
6.3.4	Tabla de șah editabilă	47
6.3.5	Pagina Play	48
6.3.6	Bara de acțiuni a paginii Play	49
6.3.7	Lista de funcționalități a paginii Play	50
6.3.8	Tabla de șah pentru jocul local	51
7	Testare	53
7.1	Testarea detectiei punctelor de intersecție	53
7.1.1	Rezultatele funcției de activare pe parcursul epocilor	53
7.1.2	Punte de intersecție filtrate prin această detectie pe imagini cu table de șah	54
7.2	Testarea detectiei tablei de șah	55
7.3	Testarea detectiei pieselor de șah	56
7.3.1	Matricea de confuzie	56
7.3.2	Curba Precision-Recall	56
7.3.3	Predictii vizuale ale pieselor	57
8	Concluzii și rezultate	58
8.1	Rezultatele testării	58
8.2	Concluzii	58

9 Dezvoltări viitoare	59
9.1 Recunoașterea în timp real	59
9.2 Redimensionarea imaginilor de înaltă calitate	59
9.3 Antrenarea pe un set de date extins pentru detectorul de piese de șah	59
9.4 Utilizarea rețelelor neuronale generative	59
9.5 Îmbunătățirea detectării colțurilor și marginilor	60
9.6 Extinderea funcționalităților de analiză a jocului	60
9.7 Jocul în rețea	60
10 Bibliografie	62
11 Anexă	63
11.1 Anexă A: Imagini de test pentru detectarea tablei de șah	63
11.2 Anexă B: Imagini de test pentru detectarea pieselor	70

Capitolul 1

Introducere

Pentru a înțelege mai bine proiectul Chess Snapshot și motivația sa, este important să ne îndreptăm atenția către viziunea computerizată, un domeniu dinamic al inteligenței artificiale ce se ocupă cu interpretarea și înțelegerea informațiilor vizuale de către computere.

Viziunea computerizată, cunoscută și sub numele de *computer vision*, reprezintă o disciplină în cadrul inteligenței artificiale care se concentrează pe dezvoltarea algoritmilor și a tehnologiilor ce permit computerelor să înțeleagă și să interpreteze informațiile vizuale, precum imagini și videoclipuri. Această disciplină a tehnologiei este utilizată într-o varietate de domenii, inclusiv în jocurile de strategie, cum ar fi șahul, pentru a oferi soluții și asistență în interpretarea și analiza informațiilor vizuale.

1.1 Scopul lucrării și motivația

Scopul proiectului Chess Snapshot este să utilizeze avansurile în domeniul viziunii computerizate pentru automatizarea procesului de recunoaștere și înregistrare a stării jocului de șah. Această aplicație oferă o soluție practică și convenabilă pasionaților de șah, eliminând necesitatea introducerii manuale a mutărilor și permitând înregistrarea rapidă și precisă a jocului desfășurat pe tabla reală.

Motivația din spatele acestei aplicații derivă din dorința de a simplifica și eficientiza procesul de înregistrare a partidelor de șah, oferind jucătorilor o modalitate mai accesibilă și mai precisă de a păstra înregistrările acestor partide.

Înregistrarea mutărilor în timpul unei partide de șah poate încetini ritmul de joc. Seturile de șah specializate care înregistrează automat mutările sunt costisitoare și nu sunt accesibile pentru jucătorii amatori. În era actuală, în care analizele și stocarea partidelor de șah se fac predominant pe computere, capacitatea de a introduce partide într-un format inteligibil pentru calculator printr-o simplă fotografie a tablei reale de șah devine din ce în ce mai relevantă.

În plus, turneele de șah desfășurate pe table reale sunt adesea transmise în timp real pe internet, oferind spectatorilor posibilitatea de a urmări jocurile pe o tablă virtuală. Această practică a devenit tot mai populară în comunitatea șahistă și necesită o metodă eficientă de înregistrare a mutărilor de pe tabla reală și de transpunere a lor în format digital.

Acest proiect propune o abordare care utilizează o singură imagine a tablei de șah, făcând cel mai probabil de pe un telefon mobil, pentru a genera o reprezentare computerizată a poziției pieselor.

Astfel, Chess Snapshot facilitează procesul de înregistrare și analiză a partidelor și contribuie la îmbunătățirea experienței de joc a pasionaților de șah, care nu dispun de

echipamente sofisticate sau de mai multe camere pentru înregistrare.

1.2 Tehnici folosite pentru recunoașterea stării jocului

1.2.1 Detectia tablei de șah

În domeniul recunoașterii tablei de șah, există soluții notabile bazate pe metode de detectare a colțurilor, precum soluția ChessVision implementată de Jialin Ding[5] și cea dezvoltată de Craig Belshe[1]. Totuși, există o tendință de a favoriza soluțiile bazate pe detectarea liniilor și marginilor. Abordările care detectează direct colțurile tablei de șah funcționează bine pentru aplicațiile generale de urmărire a jocurilor, unde tabla poate fi identificată înainte de începerea jocului, când nu există piese pe tablă. Cu toate acestea, detectarea colțurilor eșuează în cazul ocluziilor și a zgromotului de fundal.

Abordările bazate pe linii utilizează detectia contururilor în imaginea de intrare pentru a identifica marginile tablei de șah. Cunoștințele de domeniu, precum faptul că tabla poate fi identificată prin 18 linii în total și că orientările a jumătate dintre aceste linii vor fi ortogonale față de celelalte, fac ca aceste abordări bazate pe linii să fie mai robuste la zgromot. Prin urmare, reprezintă o tehnică mai populară. Această tehnică a fost implementată și explicată de Maciej A. Czyzewski, Artur Laskowski și Szymon Wasik[4].

1.2.2 Detectia pieselor

Este important de remarcat, că majoritatea modelelor pentru recunoașterea pieselor de șah sunt antrenate folosind seturi de date specializate ce includ imagini ale tablei de șah, ale pieselor și fotografii din diverse perspective, inclusiv vederi de sus sau de la nivelul ochilor. Aceste seturi de date sunt esențiale pentru antrenarea modelelor în vederea recunoașterii eficiente a pieselor de șah în diferite condiții. Astfel, modelele sunt capabile să înțeleagă și să identifice piesele din diverse unghiuri și să se adapteze la variabilitatea mediilor reale în care sunt utilizate.

Craig Belshe a implementat o soluție pentru recunoașterea pieselor de șah utilizând tehnici de viziune computerizată și învățare automată în proiectul său „Chess Piece Detection”[1]. Acest proiect folosește un model antrenat folosind YOLO (You Only Look Once), un tip de rețea neuronală convezională utilizată pentru detectarea obiectelor. Sistemul a fost antrenat pe un set de date specializat cu piese de șah pentru a identifica și clasifica tipul pieselor pe tablă. Acest tip de modele au demonstrat o eficacitate ridicată în identificarea și recunoașterea pieselor de șah pe tablă.

1.2.3 Implementări recente

Implementarea pentru detectarea pieselor de șah propusă de Craig Belshe[1] se bazează pe mai multe etape. În primul rând, este folosit algoritmul Harris, o metodă de detectare a colțurilor, pentru a identifica marginile tablei de șah dintr-o imagine. Apoi, prin aplicarea unei transformări de perspectivă, se corectează distorsiunile pentru a obține o imagine cu tabla de șah într-o poziție standardizată.

Următoarea etapă implică identificarea grilei de pătrate de șah. Imaginea transformată geometric este împărțită în 8x8 pătrate de mărimi egale. Dacă detectia colțurilor este una precisă, atunci grila coincide cu tabla de șah.

În final, se detectează piesele de șah propriu-zise. Pentru a localiza fiecare piesă în cadrul grilei, se folosește poziția punctului de jos al centrului piesei (centrul marginii inferioare a cadrului de detectie). Acest punct este calculat ca fiind centrat între colțurile stânga-jos și dreapta-jos ale piesei. Prin utilizarea acestui punct, se asigură că fiecare piesă este asociată cu pătratul corespunzător din grilă în imaginea originală.

În implementarea propusă de Jialin Ding[5] pentru detectarea pieselor de șah, procesul începe cu recunoașterea tablei de șah. Utilizatorului îi este prezentată imaginea și este solicitat să selecteze manual cele patru colțuri ale tablei, într-o anumită ordine, de exemplu, în sensul acelor de ceasornic, începând din colțul din stânga sus.

Pentru antrenarea clasificatorilor pieselor de șah, s-au utilizat imagini dintr-o bază de date. Aceste imagini de antrenare au fost selectate pentru fiecare clasă de piese și au avut un raport de aspect de 1:2. Clasificatorul pentru pion a folosit casete de delimitare cu un raport de aspect de 1:1, în timp ce clasificatorul pentru nebun a utilizat casete de delimitare cu un raport de aspect de 1:1.5. Diferența în imaginile rezultate ale casetelor de delimitare este ilustrată prin imagini eșantion.

Pentru fiecare pătrat de pe tabla de șah, fiecare dintre cele șapte clasificatoare este rulat pe o fereastră cu raportul de aspect corespunzător clasei respective de piese. Acest lucru permite detectarea și clasificarea pieselor individuale pe baza aspectului lor în raport cu pătratul de șah.

Implementarea lui Maciej A. Czyzewski et al.[4] reprezintă sursa principală de inspirație pentru Chess Snapshot. Această implementare se bazează pe o abordare bazată pe linii pentru detectarea tablei de șah și pentru transformarea geometrică a acesteia. Metoda utilizează o rețea neuronală convoluțională (CNN) pentru detectarea punctelor de intersecție, identificând astfel, împreună cu detecția marginilor, colțurile tablei de șah indiferent de orientare și distanță în imagine. Această abordare are avantajul de a funcționa mai bine în condiții de iluminare variate și evită problemele întâmpinate în abordările bazate pe detectarea colțurilor.

Pentru detectarea pieselor autorii au folosit de o metodă similară celei descrise de Jialin Ding[5], care a obținut o acuratețe de aproximativ 90% pentru identificarea pieselor individuale de șah.

1.3 Contribuții personale

1.3.1 Optimizarea eficienței algoritmilor din implementările anterioare

Pentru a îmbunătăți metodele de preprocesare a imaginilor și de detectare a liniilor și punctelor de intersecție am adus câteva modificări semnificative. Am optat pentru utilizarea unui filtru bilateral în locul filtrului Gaussian pentru reducerea zgomotului din imagine. Acest lucru permite păstrarea mai bună a marginilor din imagine, contribuind la o detecție mai precisă a liniilor și punctelor.

Prin introducerea unor noi metode de filtrare și gruparea liniilor și punctelor de intersecție detectate, am optimizat timpul de execuție și am eliminat zgomotul din datele detectate. Astfel, am redus timpul necesar pentru analiza imaginilor, fără a compromite precizia acesteia.

Am implementat mai multe ajustări minore care au avut un impact semnificativ asupra eficienței algoritmului. În ceea ce privește clasificarea punctelor de intersecție, am optat pentru o abordare mai rapidă și mai eficientă prin utilizarea directă a funcției `model(X)` în locul metodei `model.predict(X)`. Această modificare a dus la o reducere semnificativă a timpului necesar pentru clasificare, îmbunătățind astfel performanța

generală a detecției.

1.3.2 Utilizarea de unelte informaticice moderne

Am utilizat tehnologiile cu versiuni mai noi și mai performante în comparație cu proiectele prezentate anterior. Astfel, am adoptat Python 3.10, OpenCV 4.8, TensorFlow 2 și YOLOv8, beneficiind de optimizările și îmbunătățirile aduse de aceste versiuni.

Figura 1.1 prezintă comparații între diferite variante sau implementări ale algoritmului YOLO (You Only Look Once), o tehnică populară de învățare profundă utilizată pentru detecția obiectelor în imagini și videoclipuri.

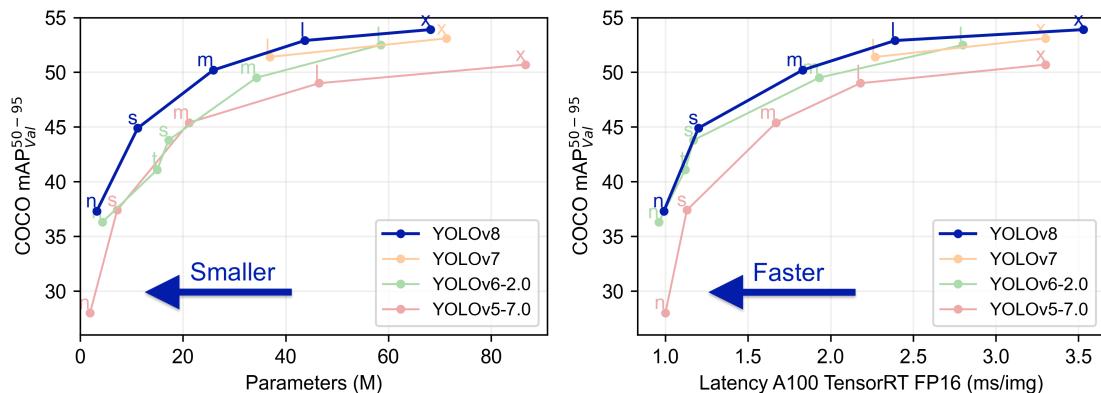


Figura 1.1: Comparație între performanța variantelor algoritmului YOLO (You Only Look Once) în detecția obiectelor. (<https://github.com/ultralytics/ultralytics>) [12]

1.3.3 Îmbogățirea setului de date pentru antrenarea modelului de detecție a pieselor de șah

Pentru a obține o detecție mai precisă a pieselor de șah, am creat un set de date extins, care include imagini din diverse perspective ale pieselor, orientări ale camerei, niveluri variate de luminozitate și fundaluri diferite. Această diversitate în setul de date a permis o antrenare a modelului pentru a fi mai robust și mai precis în detectarea pieselor în diverse condiții.

1.3.4 Dezvoltarea unei aplicații multi-platformă

În ceea ce privește interfața utilizatorului, am dezvoltat o aplicație cross-platform, cu accent pe platformele mobile Android și iOS, dar care poate fi utilizată și pe Windows. Această abordare permite o experiență consistentă și accesibilă utilizatorilor de pe diverse dispozitive.

Aceste îmbunătățiri și actualizări aduc Chess Snapshot la un nou nivel de performanță și eficiență, oferind utilizatorilor o soluție precisă și ușor de utilizat pentru capturarea și analiza pozițiilor de șah.

1.4 Cerințe și specificații

1.4.1 Cerințe

Analiza utilizării proiectului propus a stat la baza determinării cerințelor și specificațiilor acestuia. Rolul aplicației dezvoltate este să permită identificarea rapidă și precisă a poziției pieselor pe tabla de șah, pentru a servi eficient utilizatorilor interesanți de transpunerea într-un timp scurt a stării jocului în mediul digital. De asemenea, este esențial ca sistemul să fie eficient în condiții de iluminare variabilă, inclusiv iluminare deficitară sau supra-iluminare, și să poată gestiona diverse orientări și fundaluri, chiar și atunci când în imagine sunt prezente și alte obiecte în afara tablei și a pieselor de șah. Aceste cerințe au fost considerate pentru a completa specificațiile următoare:

- **Identificare rapidă și precisă a poziției pieselor:** Aplicația trebuie să fie capabilă să detecteze și să identifice rapid și precis poziția fiecărei piese de șah de pe tablă.
- **Compatibilitate cu iluminare variabilă:** Sistemul trebuie să fie eficient în gestionarea diferitelor condiții de iluminare, inclusiv în medii slab iluminate sau supra-iluminate, pentru a asigura o funcționare consistentă în diverse medii.
- **Adaptabilitate la diferite orientări și fundaluri:** Aplicația trebuie să poată gestiona diverse orientări ale tablei de șah și diferite fundaluri, asigurând totodată o detecție precisă a pieselor de șah.
- **Folosirea de tehnologii recente și eficiente:** Implementarea aplicației trebuie să se bazeze pe tehnologii moderne și eficiente din domeniul vizionării computerizate și învățării automate, pentru a asigura performanță și precizie în detecție.
- **Interfață intuitivă:** Interfața utilizatorului trebuie să fie prietenoasă și intuitivă, astfel încât utilizatorii să poată interacționa cu aplicația fără dificultăți.
- **Asigurarea scalabilității:** Arhitectura aplicației trebuie să fie proiectată astfel încât să permită scalabilitatea, să poată gestiona eficient creșterea numărului de utilizatori și volumul de date procesate.

1.4.2 Tratarea cerințelor

Identificare rapidă și precisă a poziției pieselor

Pentru a garanta o detectare precisă și eficientă a tablei de șah, am aplicat diverse strategii de optimizare în algoritm. Aceste strategii se concentrează pe filtrarea liniilor și a punctelor de intersecție, precum și pe gruparea punctelor de intersecție apropiate.

Pentru îmbunătățirea detectării liniilor tablei de șah, am aplicat diverse filtre pentru a elimina zgomotul și pentru a accentua liniile relevante.

După detectarea punctelor de intersecție și filtrarea lor folosind o rețea neuronală convolutională (CNN) cu arhitectură descrisă în detaliu într-un capitol ulterior al lucrării, procesul de grupare a acestor puncte este realizat prin aplicarea algoritmului de grupare DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Acest algoritm este ales pentru capacitatea sa de a grupa punctele apropiate în funcție de densitatea acestora în spațiu, eliminând astfel redundanța și reducând complexitatea procesării ulterioare.

Compatibilitate cu o iluminare variabilă

Pentru a asigura compatibilitatea aplicației cu o iluminare variabilă, am utilizat algoritmului CLAHE (Contrast Limited Adaptive Histogram Equalization). Acest algoritm este o variantă îmbunătățită a echilibrării tradiționale a histogramei, variantă care este adaptivă la variațiile locale ale contrastului din imagine. În plus, am antrenat modelele de învățare automată pe seturi de date variate, care includ imagini în diferite condiții de iluminare. Acest lucru a permis obținerea robusteții necesare în detectarea și recunoașterea pieselor de șah în medii cu iluminare diverse.

Adaptabilitate la diferite orientări și fundaluri

Pentru a asigura adaptabilitatea aplicației la diferite orientări ale tablei de șah și fundaluri variate, am implementat tehnici avansate de preprocesare a imaginilor, precum și algoritmi robusti de detectare și recunoaștere.

Folosirea de tehnologii recente eficiente

Chess Snapshot este o aplicație intuitivă dezvoltată cu Flutter, care permite utilizatorilor să captureze imagini ale tablei lor de șah și să recunoască automat pozițiile pieselor și starea jocului.

Back-end-ul, creat cu Flask, gestionează procesarea imaginilor și comunicarea între front-end și modelele de învățare automată. Prin utilizarea detectorului tablei de joc și modelului antrenat folosind YOLOv8, Chess Snapshot realizează o recunoaștere rapidă și precisă a pieselor pe tabla de șah, oferind o experiență fluidă utilizatorilor.

Interfață intuitivă

Pentru a oferi utilizatorilor o experiență plăcută, am creat o interfață intuitivă. Interfața permite utilizatorilor să încarce rapid imaginile cu tabla de șah și să obțină rezultatul analizei. De asemenea, am inclus opțiuni pentru ajustarea poziției pieselor de pe tabla și pentru exportarea rezultatului în format FEN (Forsyth-Edwards Notation)[3].

Asigurarea scalabilității

Pentru a face Chess Snapshot accesibil pe o gamă largă de dispozitive, ne-am concentrat pe scalabilitate. Interfața este adaptabilă și poate fi utilizată pe diverse dispozitive, inclusiv computere desktop și dispozitive mobile. Am acordat o atenție deosebită extensibilității aplicației, astfel încât să fie ușor de adăugat noi funcționalități și de integrat îmbunătățiri tehnologice în viitor.

În ansamblu, Chess Snapshot reușește să ofere o soluție completă și convenabilă pentru pasionații de șah, facilitând monitorizarea și analiza stării jocului într-un mod simplu și accesibil.

1.5 Prezentare succintă a aplicației



Figura 1.2: Logo - Chess Snapshot

Chess Snapshot este o aplicație multi-platformă concepută pentru a funcționa atât pe dispozitive mobile cu Android, cât și pe calculatoare cu Windows, oferind o experiență uniformă pe ambele platforme. Utilizând tehnologii moderne, aplicația permite utilizatorilor să detecteze și să analizeze partide de șah pornind de la o simplă imagine.

Back-end-ul aplicației este construit pe baza Flask, un cadru web micro dezvoltat în Python, ales pentru simplitatea și flexibilitatea sa. Flask facilitează interacțiunea dintre detectorul de stare al jocului de șah și interfața front-end dezvoltată în Flutter, asigurând o comunicare rapidă și eficientă între componentele aplicației. Această arhitectură modulară și scalabilă permite gestionarea logicii aplicației și furnizarea serviciilor esențiale către utilizatori.

Chess Snapshot se concentrează pe analiza și jocul de șah, oferind două funcționalități principale prin intermediul serverului său și două pagini distincte pentru experiența utilizatorului.

Aplicația utilizează două endpoint-uri principale în cadrul API-ului REST oferit de Flask:

- **Detectia stării jocului de șah:** Endpoint-ul primește o imagine a tablei de șah și returnează starea curentă a jocului în format FEN (Forsyth-Edwards Notation).
- **Determinarea mutării optime:** Endpoint-ul analizează o poziție de șah dată în format FEN și returnează cea mai bună mutare posibilă utilizând motorul de șah Stockfish.

Aceste funcționalități sunt accesate printr-o interfață simplă și intuitivă, împărțită în două pagini principale:

- **Pagina Detect:** Aici, utilizatorii pot încărca imagini ale tablelor de șah pentru a detecta și afișa starea curentă a jocului. Ei pot modifica manual pozițiile detectate sau crea noi poziții, oferind astfel un mediu interactiv pentru interpretarea și editarea pozițiilor de șah.
- **Pagina Play:** Pe această pagină, utilizatorii pot juca partide de șah împotriva altor jucători sau împotriva calculatorului. Interfața oferă funcționalități esențiale pentru gestionarea jocului și analiza pozițiilor, inclusiv butoane pentru opțiuni de joc și o interfață interactivă pentru mutarea pieselor prin tragere și plasare (drag and drop).

În concluzie, Chess Snapshot este o aplicație puternică și versatilă, creată pentru a ajuta pasionații de șah să-și analizeze și să-și îmbunătățească jocul prin intermediul unor funcționalități avansate și a unei interfețe utilizator prietenoase.

Capitolul 2

Tehnologii utilizate

Chess Snapshot folosește tehnologii avansate de viziune computerizată și învățare automată pentru a recunoaște și analiza stările jocului de șah. Aceasta permite utilizatorilor să beneficieze de recomandări și analize avansate, îmbunătățindu-și astfel abilitățile și experiența de joc.

Capitolul următor detaliază tehnologiile cheie utilizate în procesul de dezvoltare a detectorului de poziție a pieselor pe tabla de șah și a aplicației asociate. Aceste tehnologii acoperă domenii precum viziunea computerizată, învățarea automată, prelucrarea imaginilor și dezvoltarea de aplicații cross-platform. O scurtă prezentare a fiecărei tehnologii și rolul său în proiect este oferită pentru a evidenția modul în care acestea contribuie la funcționarea și performanța aplicației.

Viziunea computerizată (Computer Vision): Pentru dezvoltarea aplicației s-au utilizat algoritmi și tehnici de viziune computerizată pentru detectarea și identificarea tablelor de șah în imagini. Pentru detectarea de contururi ale pătratelor tablei și identificarea colțurilor, am implementat algoritmi de transformare Hough. Acești algoritmi permit identificarea liniilor drepte dintr-o imagine, facilitând astfel recunoașterea marginilor tablei de șah.

Învățare automată (Machine Learning): Pentru detectarea și recunoașterea pieselor de șah pe tabla de joc, am utilizat rețeaua neuronală YOLOv8 (You Only Look Once version 8). Am antrenat această rețea neuronală pe un set de date personalizat, care include diferite poziții și orientări ale pieselor de șah, pentru a asigura o detectie precisă și robustă.

Biblioteci de procesare a imaginilor: Am utilizat bibliotecile **OpenCV** și **TensorFlow** pentru prelucrarea și analiza imaginilor, facilitând detectia și recunoașterea obiectelor. Am folosit TensorFlow pentru antrenarea unui model de rețea neuronală conlovuțională (CNN). Acest model CNN este responsabil pentru clasificarea punctelor de intersecție detectate, determinând dacă acestea corespund punctelor de intersecție ale pătratelor de pe tabla de șah sau nu.

Platforme de dezvoltare de aplicații Cross-platform: Pentru a dezvolta o aplicație cross-platform, am folosit platforma de dezvoltare Flutter. Acesta a permis crearea unei aplicații centrate pe dispozitive Android și iOS, dar care poate rula și pe sistemul de operare Windows, oferind astfel o experiență consistentă utilizatorilor indiferent de platforma pe care o folosesc.

Cadre de dezvoltare a aplicațiilor web: Pentru a facilita comunicarea între partea de front-end a aplicației, scrisă în Dart folosind Flutter, și algoritmul de detectare a stării jocului de șah scris în Python, am dezvoltat un server REST API folosind cadrul de dezvoltare Flask. Acest server a acționat drept intermediu pentru transmiterea datelor între cele două componente ale aplicației.

Aceste tehnologii sunt fundamentale pentru implementarea și funcționarea corectă a aplicației de recunoaștere a poziției pieselor pe tabla de șah și asigură performanță și fiabilitatea sistemului.

2.1 Limbaje de programare

2.1.1 Python

Python este un limbaj de programare interpretat, de nivel înalt, cu un accent puternic pe simplitate și claritate în scriere. Este cunoscut pentru sintaxa sa elegantă și ușor de înțeles, ceea ce îl face potrivit pentru o gamă largă de aplicații, de la dezvoltarea web la calcul științific și inteligență artificială.

Python este utilizat în diverse domenii, inclusiv:

- **Preprocesare și antrenare de modele:** Datorită numeroaselor bibliotecii de învățare automată și prelucrare a datelor, cum ar fi TensorFlow, PyTorch și scikit-learn, Python este alegerea principală pentru preprocesarea datelor și antrenarea modelelor de învățare automată.
- **Viziune computerizată:** Python este folosit pentru implementarea algoritmilor de viziune computerizată datorită bibliotecilor precum OpenCV și scikit-image. Acestea sunt esențiale pentru aplicații precum recunoașterea de obiecte, segmentarea imaginilor și alte sarcini de procesare a imaginilor.
- **API-uri REST:** Python este des folosit pentru dezvoltarea de API-uri REST datorită simplității sale și a cadrelor de dezvoltare robuste precum Flask și Django REST Framework.

2.1.2 Dart

Dart este un limbaj de programare modern dezvoltat de Google. Lansat inițial în 2011, Dart a devenit popular pentru versatilitatea sa într-o gamă largă de aplicații, inclusiv dezvoltarea web, aplicații mobile, programare pe partea serverului și crearea de instrumente de linie de comandă.

Unul dintre obiectivele cheie ale limbajului Dart este de a oferi o experiență de dezvoltare productivă și flexibilă, păstrând în același timp o performanță ridicată pe diferite platforme.

Dart este utilizat în special pentru:

- **Crearea interfețelor utilizator (UI)** Dart este folosit în principal pentru a dezvolta interfețe utilizator atrăgătoare și receptive. Framework-ul Flutter, care este construit în Dart, este larg recunoscut pentru crearea de aplicații mobile cu performanțe ridicate și aspect atrăgător.

- **Dezvoltarea aplicațiilor mobile.** Flutter este un cadru de dezvoltare pentru aplicații mobile open-source care utilizează Dart ca limbaj de programare principal. Aceasta permite dezvoltatorilor să creeze aplicații native pentru iOS și Android folosind o singură bază de cod.
- **Dezvoltarea aplicațiilor web.** Deși nu este la fel de răspândit în acest domeniu precum în dezvoltarea aplicațiilor mobile, Dart poate fi folosit și pentru dezvoltarea aplicațiilor web. Cu ajutorul cadrului AngularDart, dezvoltatorii pot construi aplicații web dinamice și performante.

Python este utilizat într-o gamă largă de aplicații, de la învățare automată la dezvoltarea web, în timp ce Dart este preferat pentru dezvoltarea de interfețe utilizator și aplicații mobile, în special prin intermediul framework-ului Flutter.

2.2 Roboflow

Roboflow este o platformă puternică dedicată adnotării de imagini și furnizării de seturi de date, dezvoltată pentru a facilita antrenarea modelelor de învățare automată. Această platformă oferă un set variat de instrumente și funcționalități care îmbunătățesc procesul de pregătire a datelor pentru proiectele de învățare automată.

Avantajele Roboflow sunt:

- **Adnotarea de imagini.** Una dintre funcționalitățile cheie ale Roboflow este capacitatea sa de a permite utilizatorilor să adnoteze imagini într-un mod plăcut. Prin intermediul unei interfețe intuitive, utilizatorii pot adăuga etichete și descrieri diverselor obiecte și regiuni din imagini, ușurând astfel înțelegerea și interpretarea acestora de către algoritmi de învățare automată.
- **Oferta de seturi de date.** Roboflow oferă o gamă largă de seturi de date pregătite pentru utilizare în proiecte de învățare automată. Aceste seturi de date sunt curate, etichetate și pregătite într-un format compatibil cu o varietate de cadre de lucru și biblioteci de învățare automată. Utilizatorii pot accesa aceste seturi de date direct prin platforma Roboflow, fără a fi necesară o pregătire suplimentară.
- **Preprocesarea imaginilor.** Înainte de a fi utilizate pentru antrenarea modelelor de învățare automată, imaginile trebuie să fie supuse unei preprocesări adecvate pentru a asigura o performanță optimă. Roboflow oferă instrumente și funcționalități pentru preprocesarea imaginilor, inclusiv redimensionarea, normalizarea și îmbunătățirea calității acestora, pentru a asigura coerența și calitatea datelor de intrare.
- **Augmentarea imaginilor.** Augmentarea imaginilor este o tehnică esențială în dezvoltarea modelelor de învățare automată, care constă în generarea de variante noi și diverse ale imaginilor existente prin aplicarea unei serii de transformări. Roboflow facilitează acest proces prin intermediul unor funcționalități avansate de augmentare a imaginilor, cum ar fi rotația, translația, și adăugarea de zgomot, ceea ce sporește diversitatea și robustețea setului de date utilizat pentru antrenare.

2.3 Jupyter Notebook

Jupyter Notebook este o aplicație web open-source care permite crearea și partajarea documentelor interactive care conțin cod, vizualizări și texte explicative. Este larg utilizat în domeniul științific și educațional pentru explorarea datelor, dezvoltarea și testarea codului, precum și pentru crearea de rapoarte și prezentări interactive.

Jupyter Notebook prezintă numeroase avantaje, printre care:

- **Mediu interactiv pentru explorarea datelor.** Jupyter Notebook oferă un mediu interactiv în care utilizatorii pot importa, manipula și vizualiza date într-un mod flexibil. Acest mediu permite utilizatorilor să experimenteze cu datele într-un mod interactiv, facilitând înțelegerea și analiza acestora.
- **Dezvoltare și testare incrementală a codului.** Jupyter Notebook permite utilizatorilor să scrie și să execute cod într-un mod incremental și interactiv. Acest lucru ușurează dezvoltarea și testarea algoritmilor și modelelor în diverse limbaje de programare, precum Python, R, sau Julia.
- **Integrarea codului și a textului explicativ.** Jupyter Notebook permite integrarea codului, vizualizărilor și textului explicativ într-un singur document interactiv. Acest lucru permite crearea de rapoarte și prezentări interactive, ce pot fi partajate cu ușurință.

2.4 OpenCV

OpenCV (*Open Source Computer Vision Library*) este o bibliotecă open-source specializată în prelucrarea imaginilor și a viziunii computerizate. Ea oferă o gamă largă de funcționalități și algoritmi pentru analiza și manipularea imaginilor și datelor video.

Principalele funcționalități ale bibliotecii includ:

- **Procesarea și analiza imaginilor.** OpenCV oferă o suită extinsă de funcții pentru lucrul cu imagini, inclusiv citirea, scrierea, manipularea și procesarea acestora. Este o bibliotecă utilizată într-o varietate de aplicații care implică procesarea și analiza imaginilor. Aceste aplicații includ corectarea distorsiunilor, filtrarea și îmbunătățirea calității imaginilor, detectarea marginilor și a contururilor, precum și multe altele. Funcționalitățile oferite de OpenCV sunt esențiale în domenii diverse, cum ar fi viziunea computerizată, prelucrarea imaginilor medicale și inspecția industrială.
- **Detectarea și recunoașterea obiectelor.** Biblioteca OpenCV furnizează algoritmi (funcții) pentru detectarea și recunoașterea obiectelor în imagini și videoclipuri. Aceste funcții sunt folosite în aplicații precum recunoașterea fețelor, detectarea de obiecte și urmărirea acestora în timp real.

2.5 Tensorflow & Keras

TensorFlow este o platformă open-source dezvoltată de Google pentru construirea și antrenarea modelelor de învățare automată. Acesta oferă o infrastructură puternică

și flexibilă pentru dezvoltarea și implementarea modelelor de învățare automată într-o varietate de aplicații.

Keras este o bibliotecă open-source construită pe baza TensorFlow, care oferă un API simplificat și ușor de utilizat pentru construirea și antrenarea rapidă a modelelor de învățare automată. Aceasta facilitează crearea rapidă de prototipuri și experimentarea cu diferite arhitecturi de rețele neuronale.

Principalele caracteristici și utilizări ale acestora includ:

- **Construirea și antrenarea modelelor de învățare automată.** TensorFlow și Keras sunt folosite pentru construirea și antrenarea modelelor de învățare automată, inclusiv rețele neuronale convoționale (CNN), rețele neuronale recurente (RNN) și multe altele. Aceste modele pot fi utilizate într-o gamă largă de aplicații, cum ar fi clasificarea imaginilor, analiza textului și recunoașterea vocală.
- **Implementarea și evaluarea modelelor.** Există câteva instrumente esențiale pentru implementarea și evaluarea modelelor de învățare automată, acoperind aspecte precum optimizarea, funcțiile de pierdere și metricile de performanță. Aceste resurse permit dezvoltarea și evaluarea fără dificultăți a unor modele complexe.
- **Scalabilitate și performanță.** TensorFlow este proiectat pentru a fi scalabil și eficient din punct de vedere al performanței, permitând antrenarea modelelor pe seturi de date mari și utilizarea eficientă a resurselor hardware, precum GPU-uri și TPU-uri.

2.6 YOLOv8

YOLOv8 este o variantă avansată a arhitecturii YOLO (You Only Look Once), care este o metodă populară și eficientă pentru detectarea obiectelor în imagini și clipuri video. Dezvoltat pe baza versiunilor anterioare ale YOLO, versiunea 8 aduce îmbunătățiri semnificative în performanță și acuratețe.

Principalele caracteristici și avantaje ale YOLOv8 includ:

- **Performanță de vârf.** YOLOv8 utilizează tehnici de ultimă generație pentru a atinge o performanță de vârf în detecția obiectelor, ceea ce înseamnă că poate detecta și clasifica obiecte în timp real în imagini și clipuri video.
- **Arhitectură eficientă.** Arhitectura YOLOv8 este optimizată pentru a fi rapidă și eficientă din punct de vedere al resurselor, ceea ce o face potrivită pentru utilizare în aplicații cu restricții de resurse, cum ar fi sistemele încorporate sau aplicațiile mobile.
- **Capacitate de generalizare.** Datorită unui proces de antrenament îmbunătățit și a unei arhitecturi robuste, YOLOv8 este capabil să generalizeze bine în diverse scenarii și în diverse condiții de iluminare, ceea ce îl face potrivit pentru utilizare într-o varietate de aplicații.
- **Suport pentru multiple clase de obiecte.** YOLOv8 poate detecta și clasifica multiple clase de obiecte în aceeași imagine sau clip video, inclusiv obiecte comune precum mașini, persoane, animale, mobilier etc.

- **Proiect open-source cu comunitate activă.** YOLOv8 este un proiect open-source, deci codul său este disponibil public și poate fi modificat și extins de către dezvoltatori din întreaga lume. Această natură deschisă favorizează inovarea și dezvoltarea continuă a algoritmului.

2.7 Flask

Flask este un cadru de dezvoltare web Python ușor de utilizat, care permite dezvoltatorilor să creeze rapid aplicații web scalabile și flexibile. Este considerat unul dintre cele mai populare și mai utilizate cadre web în ecosistemul Python.

Principalele caracteristici și avantaje ale Flask includ:

- **Ușurință învățării și utilizării.** Flask este renumit pentru eleganța sa și pentru accesibilitatea care îl caracterizează, făcându-l o alegere ideală pentru cei care doresc să înceapă rapid și să învețe într-un mod plăcut. API-ul său simplu și intuitiv face ca dezvoltarea aplicațiilor web să fie rapidă și plăcută, fiind potrivit chiar și pentru programatori începători.
- **Flexibilitate și extensibilitate.** Flask este extrem de flexibil și permite dezvoltatorilor să aleagă și să folosească doar componentele necesare pentru aplicația lor, fără a impune o structură rigidă. De asemenea, are o arhitectură modulară, ceea ce îl face ușor de extins cu funcționalități suplimentare prin intermediul unor pachete și extensiilor.
- **Minimalism și performanță.** Flask pune accent pe minimalism și simplitate, ceea ce duce la performanțe ridicate și la un timp de răspuns scăzut al aplicației. Datorită acestui fapt, Flask este o alegere excelentă pentru dezvoltarea de aplicații web care necesită un timp de încărcare rapid și o experiență fluidă pentru utilizatori.
- **Suport pentru testare unitară și integrare simplă.** Flask vine cu un set robust de instrumente pentru testarea automatizată, permitând dezvoltatorilor să creeze și să ruleze teste unitare și de integrare pentru aplicațiile lor. Această capacitate facilitează dezvoltarea și menținerea codului, asigurând calitatea și fiabilitatea aplicației.
- **Comunitate activă și documentație detaliată.** Flask are o comunitate activă de dezvoltatori și utilizatori, care oferă suport și resurse pentru a ajuta la rezolvarea problemelor și la învățarea continuă. De asemenea, are o documentație bogată și detaliată, care oferă ghiduri și exemple pentru a facilita dezvoltarea.

2.8 Flutter

Flutter este un cadru de dezvoltare open-source creat de Google, special conceput pentru construirea interfețelor utilizator native pentru aplicații mobile, web și desktop, toate utilizând același cod sursă. Folosind limbajul de programare Dart, Flutter oferă un mediu de dezvoltare rapid și eficient pentru construirea aplicațiilor cu aspect și performanțe native pe mai multe platforme.

Principalele caracteristici și avantaje ale Flutter includ:

- **Interfețe utilizator native.** Flutter utilizează un set extensiv de widget-uri personalizabile pentru a construi interfețe utilizator native, ceea ce înseamnă că aplicațiile create cu Flutter se simt și se comportă ca aplicații native pe platforma țintă.
- **Unicode și consistență.** Flutter promovează un codebase unic pentru toate platformele, permitând dezvoltatorilor să scrie o singură dată și să ruleze aplicația pe mai multe platforme fără a necesita modificări majore. Acest lucru asigură o consistență în funcționalitate și aspect pe toate dispozitivele.
- **Performanță ridicată.** Flutter utilizează motorul grafic Skia pentru a desena interfețele utilizator, oferind performanțe de vârf și o experiență fluidă pentru utilizatori. Acest lucru se datorează în parte și faptului că Flutter rulează aplicația în mod nativ, fără a folosi un intermedian de tip WebView sau container.
- **Rapiditate în dezvoltare.** Flutter vine cu un set bogat de instrumente și biblioteci, care accelerează procesul de dezvoltare și facilitează implementarea funcționalităților complexe. Hot reload este o caracteristică deosebit de utilă, permitând dezvoltatorilor să vadă modificările în timp real în timpul dezvoltării.
- **Comunitate activă și suport extensiv.** Flutter beneficiază de o comunitate activă și dinamică de dezvoltatori, care oferă suport și resurse pentru a ajuta la rezolvarea problemelor și la învățarea continuă. De asemenea, are o documentație bogată și actualizată constant, care oferă ghiduri și exemple pentru dezvoltatori.

2.9 Alte instrumente

În dezvoltarea aplicației Chess Snapshot am utilizat biblioteci Python precum Matplotlib pentru vizualizarea datelor, Scikit-learn pentru sarcini de analiză de date și GitHub pentru gestionarea codului sursă.

2.9.1 Matplotlib

Matplotlib este o bibliotecă Python amplu utilizată pentru vizualizarea datelor. Aceasta permite creatorilor de aplicații să realizeze o varietate de vizualizări, de la simple grafice de linie până la diagrame complexe, pentru a interpreta și prezenta datele într-un mod intuitiv.

Principalele caracteristici și avantaje ale Matplotlib includ:

- **Versatilitate.** Matplotlib suportă diverse tipuri de grafice și vizualizări, inclusiv grafice de linie, bare, scatter, histograme și multe altele. Această versatilitate permite utilizatorilor să aleagă reprezentarea vizuală care se potrivește cel mai bine cu datele lor.
- **Control detaliat asupra vizualizărilor.** Utilizatorii au un control fin asupra tuturor aspectelor graficelor create cu Matplotlib, de la culori și stiluri de linii, până la personalizarea axelor și adăugarea de adnotări. Acest nivel de personalizare ajută la crearea unor vizualizări precise și informative.
- **Integrare cu alte biblioteci.** Matplotlib se integrează bine cu alte biblioteci populare din ecosistemul Python, cum ar fi NumPy și Pandas. Aceasta facilitează fluxurile de lucru complexe de analiză și vizualizare a datelor.

- **Comunitate activă și documentație detaliată.** Matplotlib beneficiază de o comunitate activă de utilizatori și dezvoltatori, care contribuie cu cod, tutoriale și soluții la probleme comune. De asemenea, are o documentație bogată și detaliată, care ajută utilizatorii să învețe să utilizeze biblioteca eficient.
- **Suport pentru generarea de imagini de înaltă calitate.** Matplotlib permite generarea de grafice și imagini de înaltă calitate, care pot fi utilizate în publicații științifice, prezentări și rapoarte profesionale. Aceasta asigură că vizualizările nu doar arată bine, ci și comunică clar informațiile.

2.9.2 Scikit-learn

Scikit-learn este o altă bibliotecă Python vitală, specializată în învățarea automată și data mining. Această bibliotecă oferă o gamă largă de algoritmi pentru clasificare, regresie, grupare și alte sarcini de analiză de date. În plus, scikit-learn include instrumente pentru evaluarea modelelor și selectarea caracteristicilor, facilitând dezvoltarea rapidă și eficientă a soluțiilor de învățare automată.

Principalele caracteristici și avantaje ale Scikit-learn includ:

- **Diversitatea algoritmilor.** Scikit-learn oferă o varietate largă de algoritmi de învățare automată pentru sarcini comune, cum ar fi clasificarea, regresia, gruparea și reducerea dimensionalității. Aceasta permite dezvoltatorilor să aleagă și să aplice algoritmul cel mai potrivit pentru datele și problemele lor.
- **API simplu și coerent.** Biblioteca are un API simplu și coerent, care face utilizarea algoritmilor de învățare automată intuitivă și ușor de învățat. Aceasta permite dezvoltarea rapidă și testarea modelelor.
- **Integrare cu alte instrumente.** Scikit-learn se integrează bine cu alte biblioteci de date și știință a datelor din ecosistemul Python, cum ar fi NumPy, Pandas și Matplotlib. Aceasta permite utilizatorilor să combine funcționalități și să creeze fluxuri de lucru eficiente și puternice.
- **Documentație detaliată și comunitate activă.** Scikit-learn are o documentație amplă, care include ghiduri, exemple și tutoriale pentru a ajuta utilizatorii să învețe și să aplice tehniciile de învățare automată. De asemenea, beneficiază de o comunitate activă care oferă suport și contribuie la dezvoltarea continuă a bibliotecii.
- **Performanță și scalabilitate.** Scikit-learn este optimizat pentru performanță și scalabilitate, permitându-i să gestioneze seturi de date mari și complexe. Aceasta asigură că modelele pot fi antrenate și testate eficient, chiar și pe volume mari de date.

2.9.3 GitHub

GitHub este o platformă de gestionare a codului sursă bazată pe sistemul de control al versiunilor Git. Este utilizată pentru colaborarea la proiecte de dezvoltare de software, permitând programatorilor să își încarce, să își revizuiască și să își gestioneze codul sursă în mod eficient.

Principalele caracteristici și avantaje ale GitHub includ:

- **Controlul versiunilor.** GitHub utilizează Git pentru controlul versiunilor, permitând dezvoltatorilor să urmărească modificările aduse codului, să revină la versiuni anterioare și să gestioneze ramurile de dezvoltare. Aceasta asigură un istoric clar și organizat al evoluției codului sursă.
- **Colaborare eficientă.** GitHub facilitează colaborarea între dezvoltatori prin funcționalități precum pull requests, code reviews și issues tracking. Aceste instrumente permit echipele să colaboreze eficient, să discute și să aducă îmbunătățiri codului într-un mod colaborativ.
- **Servicii și instrumente.** GitHub se integrează cu o varietate de alte servicii și instrumente, cum ar fi sisteme de integrare continuă (CI/CD), platforme de testare și servicii de cloud. Aceasta permite automatizarea fluxurilor de lucru și îmbunătățirea eficienței dezvoltării.
- **Hosting pentru proiecte open-source și private.** GitHub oferă găzduire pentru proiecte open-source, permitând dezvoltatorilor să își partajeze și să își promoveze munca. De asemenea, suportă proiecte private, asigurând confidențialitatea și securitatea codului sursă pentru proiecte comerciale sau personale.
- **Documentație și resurse educaționale.** GitHub oferă o varietate de resurse educaționale și documentație pentru a ajuta utilizatorii să învețe să utilizeze platforma și să îmbunătățească practicile de dezvoltare software. De asemenea, platforma găzduiește o mulțime de proiecte open-source care pot servi drept exemple sau puncte de pornire pentru dezvoltarea ulterioară.

Capitolul 3

Arhitectură

Acest capitol este dedicat arhitecturii detectorului stării jocului de șah oferă o perspectivă detaliată asupra modului în care sistemul este conceput și funcționează. Aceasta include două diagrame relevante: una pentru ilustrarea procesului de detectare și alta pentru a evidenția conexiunea acestuia cu aplicația de telefon.

3.1 Arhitectura detectorului stării jocului de șah

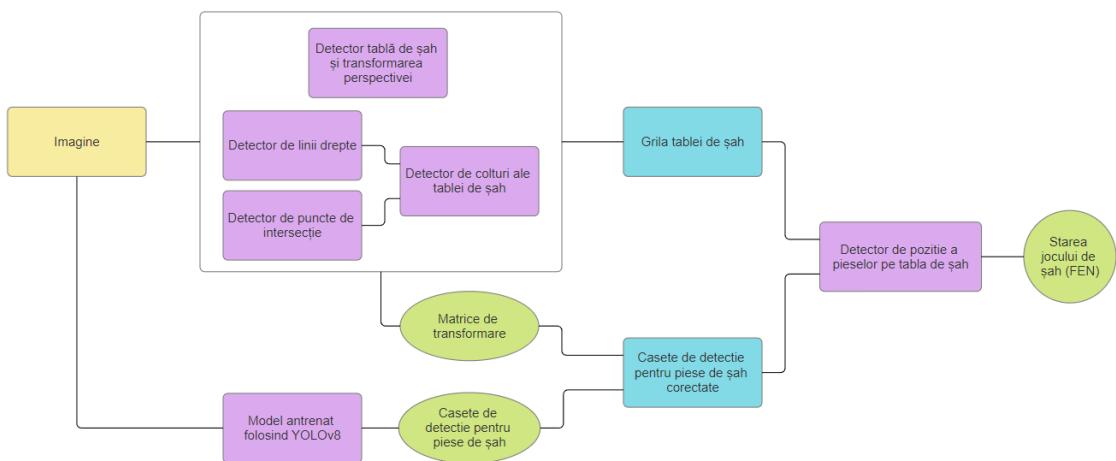


Figura 3.1: Diagrama arhitectura detectorului stării jocului de șah

Diagrama prezentată în Figura 3.1 ilustrează componentele principale și fluxurile de date în cadrul sistemului de recunoaștere. O explicație cuprinzătoare a fiecărui element reprezentat în diagramă este prezentată mai jos:

- **Imaginea de intrare** reprezintă imaginea originală în care se află tabla de șah, cu piesele dispuse pe ea.
- **Detectorul pentru tablă de șah și transformarea perspectivelor** primește imaginea de intrare și aplică mai multe transformări de perspectivă pentru a obține o imagine în care se găsește doar tabla de șah, văzută de sus și cu pătrate de dimensiuni aproape egale. De asemenea, generează matrici de transformare folosite pentru corectarea detectiilor pieselor de șah.

- **Detectorul de linii drepte** utilizează algoritmul Hough pentru detectarea liniilor drepte din imaginea de intrare. Aceste liniile sunt considerate a fi muchiile pătratelor tablei de șah, care sunt importante pentru a determina poziția și dimensiunea acesteia.
- **Detectorul de puncte de intersectare** identifică punctele de intersectare dintre pătratele tablei de șah filtrând intersecțiile dintre liniile drepte recunoscute anterior.
- **Detectorul de colțuri ale tablei de șah** identifică precis colțurile tablei folosind liniile drepte și punctele de intersectare găsite. Aceste colțuri sunt utilizate pentru a estima geometria tablei și pentru a corecta perspectiva imaginii.
- **Modelul antrenat folosind YOLOv8** este o rețea neuronală conoluțională (CNN) dedicată recunoașterii pieselor de șah într-o imagine originală. Acesta este capabil să identifice diverse tipuri de piese și să localizeze pozițiile lor pe tablă.
- **Grila tablei de șah.** Rezultatul detectării tablei de șah, este o imagine transformată și împărțită în 64 de pătrate, reprezentând grila de 8 rânduri și 8 coloane a tablei de șah.
- **Casete de detectie pentru piese de șah corectate** reprezintă pozițiile pieselor corectate în funcție de transformarea aplicată pe imaginea originală. Se folosesc matricile de transformare rezultate în urma detectării tablei de șah.
- **Detectorul de poziție a pieselor pe tabla de șah** determină starea actuală a jocului de șah, utilizând grila tablei de șah și informațiile despre pozițiile corectate ale pieselor, și o reprezintă sub formă de sir de caractere conform standardului FEN(Forsyth-Edwards Notation)[3], oferind astfel o descriere completă a poziției pieselor pe tablă.

3.2 Arhitectura aplicației

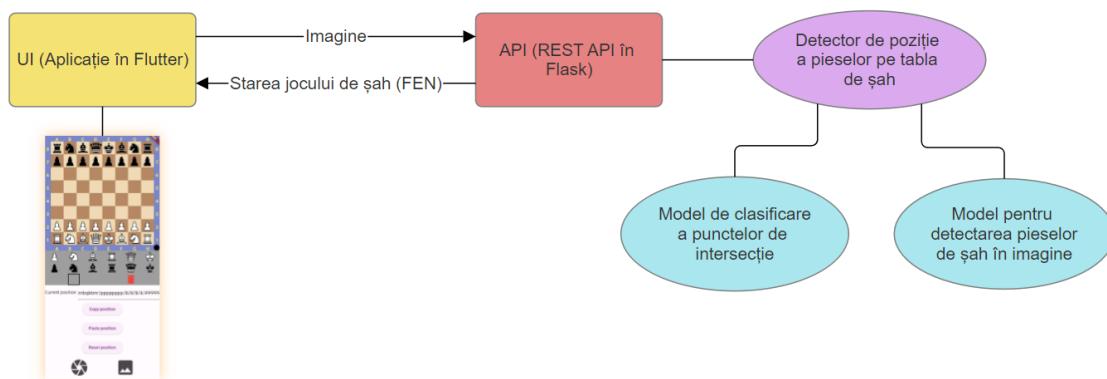


Figura 3.2: Diagrama reprezentând arhitectura aplicației

Arhitectura aplicației este construită pe modelul client-server, unde interacțiunea se realizează între un client, reprezentat de interfața utilizator (UI), și un server care găzduiește API-ul (Interfața de programare a aplicației). Serverul dispune de modulul Detector de poziție a pieselor pe tabla de șah, care a fost explicitat în detaliu în secțiunea anterioară.

UI (Aplicație în Flutter)

Interfața utilizator reprezintă partea client a aplicației și este dezvoltată folosind cadrul de dezvoltare Flutter pentru a oferi o experiență de utilizare interactivă și plăcută. Această interfață conține butoane pentru capturarea unei imagini cu camera dispozitivului sau selectarea unei imagini din fișierele de pe dispozitivul utilizatorului. De asemenea, include o zonă care poate fi editată pentru afișarea rezultatelor detecției stării jocului de șah și alte funcționalități cum ar fi copierea și editarea șirului de caractere FEN (Forsyth-Edwards Notation)[3].

API (REST API în Flask)

Serverul găzduiește un API REST care expune un endpoint accesibil clientului. Acest endpoint primește o imagine de intrare și, folosind detectorul de poziție a pieselor pe tabla de șah descris în secțiunea precedentă, generează și returnează un șir de caractere FEN care descrie poziția pieselor de șah detectate pe tablă.

Detector de poziție a pieselor pe tabla de șah

Acesta este detectorul principal al aplicației, scris în Python, care primește o imagine ca intrare și localizează pozițiile pieselor de șah pe tablă. Acesta folosește modele pentru detectarea și clasificarea punctelor de intersectare și pentru detectarea pieselor de șah în imagine.

Model de clasificare a punctelor de intersecție

Acesta este un model antrenat cu Keras și TensorFlow, care este utilizat pentru filtrarea punctelor de intersectare dintre marginile detectate. Acest model este un CNN (Rețea Neuronala Convolutională) care primește o imagine procesată de 21x21 pixeli și returnează o clasificare, indicând dacă punctul este o intersecție între colțurile a patru pătrate ale tablei de șah sau nu.

Model pentru detectarea pieselor de șah în imagine

Acest model, antrenat folosind YOLOv8 (You Only Look Once)[12], primește o imagine ca intrare și returnează o listă de cadre de delimitare pentru piesele de șah detectate.

Aceste componente lucrează împreună pentru a oferi funcționalitatea completă a aplicației, de la capturarea imaginilor până la interpretarea pozițiilor pieselor de șah și afișarea lor pe interfața utilizator.

Capitolul 4

Algoritmi de procesare de imagini folosiți

În dezvoltarea aplicației pentru detectarea și recunoașterea tablei de șah și a pieselor de șah, s-au folosit diversi algoritmi de procesare a imaginilor din biblioteca OpenCV. Acești algoritmi asigură precizia și eficiența în recunoașterea automată a elementelor de pe tabla de șah și sunt descriși în cele ce urmează.

4.1 Egalizarea adaptivă a histogramelor (CLAHE)

Egalizarea adaptivă a histogramelor folosind metoda CLAHE (Contrast Limited Adaptive Histogram Equalization) este o tehnică utilizată pentru îmbunătățirea contrastului și vizibilității detaliilor în imagini. În contrast cu egalizarea globală a histogramelor, care poate duce la o creștere exagerată a zgomotului și a contrastului în anumite regiuni ale imaginii, CLAHE aplică egalizarea histogramelor local, adaptând procesul pentru fiecare bucătă a imaginii.

Algoritmul CLAHE împarte imaginea în regiuni mai mici numite blocuri și aplică egalizarea histogramelor asupra fiecărui bloc. Pentru a limita amplificarea zgomotului în regiuni cu variații reduse ale intensității, CLAHE aplică un proces de limitare a contrastului.

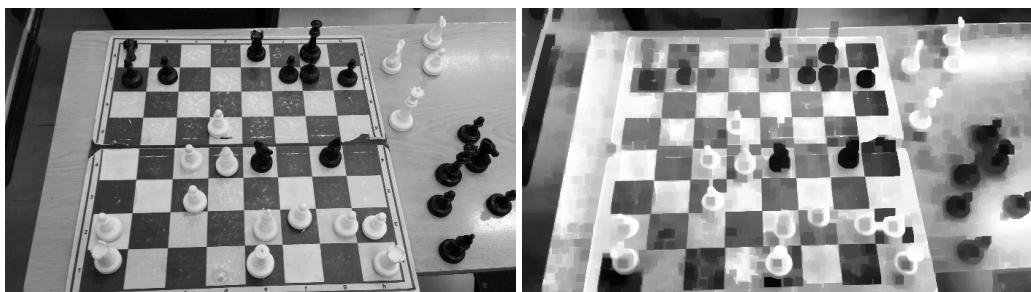


Figura 4.1: Imagine originală

Figura 4.2: Imagine rezultată

Figura 4.3: Imagine rezultată după aplicarea egalizării adaptive CLAHE cu parametrii $\text{limit}=3$, $\text{grid}=(6, 2)$ și $\text{iterations}=5$

Prin aplicarea CLAHE, se obține o imagine cu contrast îmbunătățit și detalii precum pătratele tablei de șah mai bine vizibile, pregătind astfel terenul pentru etapele ulterioare ale procesării imaginilor, cum ar fi detectia marginilor și identificarea liniilor.

Algoritmul CLAHE poate fi aplicat cu diferiți parametri asupra imaginii originale pentru a obține rezultate variate. Ajustarea parametrilor precum dimensiunea blocului și limita contrastului poate influența semnificativ aspectul final al imaginii prelucrate.

Funcția OpenCV corespunzătoare acestui algoritm este:

```
for _ in range(iterations):
    img = cv.createCLAHE(clipLimit=limit, tileSize=grid).apply(img)
```

Parametrii metodei reprezintă:

- **img:** Imaginea de intrare asupra căreia se aplică CLAHE.
- **clipLimit (limit):** Limita contrastului pentru normalizare. Aceasta restricționează extinderea histogramelor locale, limitând contrastul local.
- **tileGridSize (grid):** Dimensiunea blocului pentru care se realizează normalizarea. Imaginea este împărțită în blocuri, iar contrastul este egalizat separat în fiecare bloc.
- **iterations:** Numărul de iterații pentru aplicarea repetată a algoritmului CLAHE. Acest lucru poate fi util pentru obținerea unei îmbunătățiri suplimentare a calității imaginii prelucrate.

4.2 Binarizarea folosind algoritmul Otsu

Algoritmul Otsu este un algoritm de determinare a unui prag global optim pentru binarizarea imaginii. Acest prag se calculează automat, ținând cont de distribuția intensității pixelilor din imagine. Pragul optim este selectat astfel încât să maximizeze dispersia dintre clasele de pixeli (foreground și background). Funcția OpenCV corespunzătoare acestui algoritm este:

```
dimg = cv.threshold(dimg, 0, 255, cv.THRESH_OTSU) [1]
```

Parametrii algoritmului reprezintă:

- **dimg:** Imaginea de intrare care urmează să fie binarizată.
- **thresh (0):** Este valoarea pragului minim pentru binarizare. Orice valoare sub acest prag va fi considerată pixel negru.
- **maxval (255):** Este valoarea pragului maxim pentru binarizare. Orice valoare mai mare decât acest prag va fi considerată pixel alb.
- **type (cv.THRESH_OTSU):** Indicatorul pentru utilizarea algoritmului Otsu pentru determinarea pragului de binarizare.

Prin utilizarea algoritmului de binarizare Otsu, se obține o binarizare automată a imaginii, adaptată dinamic la distribuția intensității pixelilor, ceea ce duce la o pre-procesare eficientă și precisă în cadrul aplicației de recunoaștere a tablei și pieselor de șah.

4.3 Filtrarea bilaterală

Filtrarea bilaterală este o tehnică de filtrare a imaginilor care reduce zgomotul din imagine, păstrând în același timp marginile nealterate. Această filtrare combină două componente de filtrare: o componentă spațială, care ține cont de distanța dintre pixeli, și o componentă de intensitate, care ține cont de diferența de intensitate între pixeli. Un exemplu de utilizare al funcției OpenCV corespunzătoare asupra imaginii de intrare `img` este:

```
img = cv.bilateralFilter(img, 7, 75, 75)
```

În care parametrii reprezintă:

- **d (7):** Diametrul ferestrei de filtrare în jurul fiecărui pixel. Cu cât este mai mare această valoare, cu atât se iau în considerare mai mulți pixeli în procesul de filtrare, ceea ce poate duce la o estompare mai mare a marginilor.
- **sigmaColor (75):** Deviația standard a componentei spațiale. O valoare mai mare înseamnă că pixelii mai îndepărtați vor fi luati în considerare în filtrare.
- **sigmaSpace (75):** Deviația standard a componentei de intensitate. Cu cât este mai mare acest număr, cu atât pixelii cu intensități mai diferite vor fi considerați similari și vor fi mai puțin atenuați.

4.4 Detectarea marginilor folosind operatorul Canny

Operatorul Canny este utilizat pentru detectarea eficientă a marginilor în imagini. Acesta constă în mai mulți pași:

1. **Conversia la scală de gri.** Algoritmul Canny din OpenCV necesită o imagine în scala gri pentru a detecta marginile.
2. **Reducerea zgomotului.** Aplicarea unui filtru de reducere a zgomotului, cum ar fi filtrul Gaussian, pentru a îmbunătăți detectia marginilor.
3. **Calcularea gradientului.** Algoritmul calculează magnitudinea și direcția gradientului în fiecare punct al imaginii.
4. **Non-maximum suppression.** Presupune subțierea la grosimea de un pixel a marginilor detectate pe baza gradientului,
5. **Hysteresis thresholding.** Folosește două praguri pentru a identifica marginile finale, reducând astfel din fragmentarea contururilor și din zgomot. Pixelii cu valori de gradient mai mari decât pragul superior sunt considerați margini sigure, iar cei cu valori între cele două praguri sunt margini potențiale și vor fi considerați margini sigure, doar dacă sunt conectați la margini sigure.

Funcția corespunzătoare în OpenCV pentru algoritmul Canny se aplica unei imagini (`img`) cu următorii parametri:

```
cv.Canny(img, t1, t2)
```

- **t1** reprezintă pragul inferior pentru hysteresis thresholding. Orice margini cu valori de gradient mai mici decât acest prag sunt eliminate.
- **t2** reprezintă pragul superior pentru hysteresis thresholding. Orice margini cu valori de gradient mai mari decât acest prag sunt considerate sigure, iar cele între **t1** și **t2** sunt luate în considerare, doar dacă sunt conectate la margini sigure.

4.5 Detectarea liniilor folosind transformarea Hough

Transformarea Hough probabilistă este utilizată pentru detectarea liniilor drepte în imagini. Acest algoritm operează în spațiul parametric Hough, transformând ecuațiile liniilor într-o reprezentare spațială. Astfel, prin analiza acestei reprezentări, se identifică liniile în imagine. Mai exact, fiecare punct din imagine contribuie la o curbă în spațiul Hough care reprezintă toate liniile posibile care ar putea trece prin acel punct. Astfel, prin găsirea punctelor în care aceste curbe se intersectează, se identifică liniile din imagine.

Algoritmul transformării Hough liniare estimează cei doi parametri care definesc o linie dreaptă. Spațiul de transformare are două dimensiuni, iar fiecare punct din acest spațiu este utilizat ca un acumulator pentru a detecta sau identifica o linie descrisă de $\rho = x\cos\theta + y\sin\theta$. Fiecare punct de pe contururile detectate în imagine contribuie la acumulatoare.

Dimensiunea accumulatorului este egală cu numărul de parametri necunoscuți, adică doi, considerând valori cuantificate ale lui ρ și θ în perechea (ρ, θ) . Pentru fiecare pixel la (x, y) și vecinătatea sa, algoritmul transformării Hough determină dacă există suficiente dovezi ale unei liniilor drepte la acel pixel. Dacă da, va calcula parametrii (ρ, θ) ai aceleiai liniilor, apoi va căuta clasa accumulatorului în care parametrii se încadrează și va incrementa contorul acestei clase.

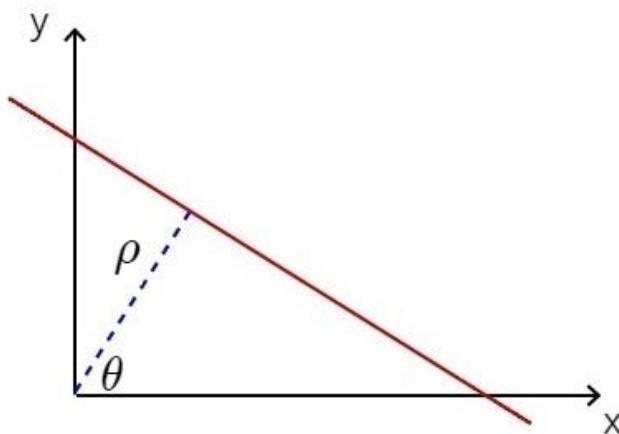


Figura 4.4: Linie detectată într-un sistem de coordonate cartezian (x, y) folosind transformarea Hough [6]

Un exemplu de aplicare al funcției corespunzătoare din OpenCV asupra unei imagini este:

```
lines = cv.HoughLinesP(img, rho=1, theta=np.pi/180,
threshold=threshold, minLineLength=minLineLength, maxLineGap=maxLineGap)
```

Parametrii funcției cv.HoughLinesP sunt:

- **img:** Imaginea de intrare asupra căreia se aplică detecția liniilor.
- **rho:** Rezoluția distanței ρ în pixeli. De obicei, este setată la 1 pixel.
- **theta:** Rezoluția unghiului θ în radiani. De obicei, este setată la $np.pi/180$ (un grad).
- **threshold:** Pragul minim pentru voturile necesare pentru a considera o linie dreaptă. Cu cât este mai mare acest parametru, cu atât sunt necesare mai multe voturi.
- **minLineLength:** Lungimea minimă a unei linii. Liniile mai scurte decât această valoare sunt respinse.
- **maxLineGap:** Distanța maximă între două puncte consecutive care pot fi considerate parte a aceleiași linii.

4.6 Transformarea perspectivă

Transformarea de perspectivă în procesarea imaginilor este o operațiune fundamentală folosită pentru a modifica sau a ajusta poziția relativă a obiectelor dintr-o imagine. Aceasta implică modificarea punctelor de vedere ale unei imagini, astfel încât obiectele să apară sub un unghi diferit sau să fie redimensionate pentru a părea că sunt văzute din altă perspectivă.

În esență, transformarea de perspectivă implică proiecția unei imagini 3D pe un plan 2D, ceea ce poate duce la distorsiuni sau deformări ale obiectelor, în funcție de unghiul de vedere și de poziționarea acestora în raport cu camera sau observatorul. Această tehnică este utilizată într-o varietate de aplicații, precum corectarea distorsiunilor în imagini, generarea de efecte artistice sau ajustarea imaginilor pentru analiza și procesarea ulterioară.

Pentru a realiza o transformare perspectivă, este necesar să stabilim patru puncte de referință corespunzătoare colțurilor unui obiect de interes din imaginea originală. Aceste puncte sunt folosite pentru a calcula o matrice de transformare, care va redimensiona și roti imaginea pentru a corecta distorsiunile de perspectivă. Astfel, prin aplicarea matricii de transformare, coordonatele fiecărui punct din imaginea originală (x, y) sunt transformate în coordonatele corespunzătoare din imaginea rezultată (x', y') , astfel încât obiectele să fie reprezentate conform noii perspective dorite.

Formula matematică pentru calculul transformării perspectivice este dată de:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Unde:

- (x, y) sunt coordonatele unui punct în imaginea originală.
- (x', y') sunt coordonatele transformate ale aceluiași punct în imaginea rezultată.
- Matricea de transformare $\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}$ este determinată de cele patru puncte de referință și reprezintă transformarea perspectivă aplicată imaginii.

Aplicațiile transformării perspective sunt diverse și includ:

- **Corectarea distorsiunilor de perspectivă:** În multe situații, fotografiile pot prezenta distorsiuni de perspectivă din cauza unghiului de fotografie sau a poziționării obiectului față de cameră. Transformarea perspectivă este utilizată pentru a corecta aceste distorsiuni, astfel încât obiectele să apară într-o perspectivă plană.
- **Reconstrucția obiectelor în 3D:** Prin aplicarea transformării perspectivă asupra mai multor imagini ale aceluiași obiect din unghiuri diferite, este posibil să se reconstruiască obiectul în spațiul tridimensional. Această tehnică este utilizată în domenii precum cartografierea 3D, modelarea obiectelor și realitatea augmentată.
- **Recunoașterea și urmărirea obiectelor:** În aplicații de recunoaștere a obiectelor sau urmărire a mișcării, transformarea perspectivă poate fi folosită pentru a alinia obiectele din imagini într-o perspectivă comună, facilitând extragerea caracteristicilor și compararea acestora între cadre.
- **Repararea imaginilor deteriorate:** În unele cazuri, imaginile pot fi deteriorate sau distorsionate din cauza factorilor precum îndoiri sau încovoierea hârtiei fotografice. Transformarea perspectivă poate fi utilizată pentru a corecta aceste distorsiuni și a restabili aspectul original al imaginii.

În esență, transformarea perspectivă este o unealtă esențială în arsenalul de tehnici de prelucrare a imaginilor și este folosită într-o varietate de aplicații, inclusiv în identificarea corectă a pozițiilor pieselor de șah pe tabla de joc.

Pentru a aplica o transformare perspectivă în procesarea imaginilor, este necesară determinarea și aplicarea unei matrice de transformare, care descrie corectarea perspectivă necesară.

```
matrix = cv.getPerspectiveTransform(corner_points, output_points)
```

Parametri:

- **corner_points:** O mulțime de patru puncte care reprezintă colțurile obiectului din imaginea originală.
- **output_points:** O mulțime de patru puncte care reprezintă colțurile obiectului din imaginea rezultată.

Aplicarea matricei de transformare asigură corectarea perspectivă dorită în imaginea originală.

```
transformed_image = cv.warpPerspective(image, matrix, output_size)
```

Parametri:

- **image:** Imaginea originală asupra căreia se aplică transformarea perspectivă.
- **matrix:** Matricea de transformare perspectivă obținută anterior.
- **output_size:** Dimensiunea imaginii rezultate, specificată ca o pereche (lățime, înălțime).

Acești algoritmi formează baza procesării imaginilor în aplicația de recunoaștere a tablei și pieselor de șah, contribuind la obținerea grilei tablei de șah.

Capitolul 5

Detectia tablei și identificarea stării jocului

Acest capitol se va concentra pe metodele de identificare a stării jocului dintr-o imagine. Această problemă este împărțită în trei etape principale prezentate în continuare.

5.1 Detectia tablei

Detectia tablei urmărește identificarea și delimitarea unei table de șah într-o imagine, cu scopul de a crea ulterior o înregistrare digitală a pozițiilor pieselor de șah folosind notația Forsyth-Edwards (FEN) [3], cea mai răspândită metodă de reprezentare a stării jocurilor de șah.

Pentru a recunoaște tabla de șah într-o imagine, am implementat o clasă numită `ChessboardDetector`. Această clasă conține o metodă numită `detect`, care primește doi parametri: imaginea în care se va face recunoașterea și numărul de straturi. Acest parametru definește de câte ori se va repeta algoritmul de detectare și transformare geometrică prezentat mai jos. Cu fiecare iterare, algoritmul va încerca să obțină o estimare mai precisă a poziției tablei de șah din imagine. Să presupunem că algoritmul este setat la 5 iterări. La prima iterare, algoritmul va face o estimare inițială a poziției tablei de șah. La a doua iterare, algoritmul va utiliza această estimare inițială ca punct de plecare pentru a obține o estimare mai precisă. Acest proces va fi repetat de încă 3 ori, cu fiecare iterare rezultând o estimare mai precisă a poziției tablei de șah.

```
def layer(self):
    self.detect_lines()
    self.detect_intersections()
    if len(self.intersections) < 4:
        return
    self.detect_corners()
    self.transform()
```

Figura 5.1: Metoda de detectare a tablei de șah (o sigură iterare)

Algoritmul folosit pentru detectarea tablei de șah implică identificarea unor caracteristici specifice ale tablei de șah, cum ar fi liniile drepte și punctele de intersecție dintre pătratele

acesteia, și ajustarea acestora pentru a obține o imagine cu tabla de șah aliniată corect. Acest proces poate fi iterat de mai multe ori pentru a obține o aproximare mai bună.

Pentru a realiza o detecție eficientă a unei table dintr-o imagine, două sau trei interații ale algoritmului de detecție sunt considerate suficiente.

Algoritmul folosit este compus din patru pași:

1. **Detectia liniilor drepte.** În acest pas, algoritmul identifică liniile drepte din imagine. Acest lucru poate fi realizat folosind tehnici de detectare a muchiilor și transformări Hough pentru a găsi segmente de linii în imagine.
2. **Detectia punctelor de intersecție.** După ce s-au detectat liniile drepte, algoritmul caută punctele de intersecție ale acestora. Aceste puncte de intersecție sunt utilizate, în pasul următor, pentru a determina configurația și poziționarea liniilor, ceea ce este crucial pentru detectarea colțurilor tablei de șah.
3. **Detectia colțurilor tablei de șah.** Utilizând informațiile obținute din detectarea punctelor de intersecție, algoritmul localizează colțurile tablei de șah. Aceste colțuri sunt punctele în care se întâlnesc liniile detectate, indicând marginile pătratelor tablei de șah.
4. **Transformarea geometrică.** În ultimul pas, se efectuează o transformare geometrică pentru a corecta orice distorsiuni de perspectivă și a alinia imaginea astfel încât pătratele tablei de șah să fie paralele cu marginile imaginii. Acest lucru asigură o reprezentare corectă a tablei de șah și a poziției pieselor sale în imagine.

În urma executării funcției `detect`, rezultatul este o imagine transformată geometric, în care se află numai tabla de șah cu piesele de șah, iar colțurile tablei corespund colțurilor imaginii. Acest proces asigură o detecție precisă și robustă a tablei de șah, esențială pentru aplicarea ulterioară în diverse aplicații de analiză și procesare a imaginii, cum ar fi recunoașterea automată a pozițiilor pieselor și generarea unei notății FEN corecte.

5.1.1 Detectia liniilor drepte

Detectarea și extragerea liniilor drepte din imagini reprezintă o componentă fundamentală în prelucrarea imaginilor și viziunea computerizată. Această tehnică joacă un rol crucial într-o varietate de aplicații, de la simpla identificare a contururilor până la sarcini mai complexe, precum recunoașterea obiectelor și navigarea autonomă a vehiculelor.

Pentru a detecta liniile drepte într-o imagine, sunt utilizate diverse tehnici și algoritmi, printre care se numără:

- **Transformata Hough:** Transformata Hough este o metodă comună utilizată pentru detectarea liniilor drepte în imagini. Această metodă eficientă transformă informația din imagine într-o reprezentare abstractă, în care liniile drepte sunt simplificate și ușor de identificat. Detaliile acestui proces sunt prezentate în capitolul anterior.
- **Filtrarea imaginii.** Înainte de aplicarea algoritmilor de detectare a liniilor drepte, este adesea util să se filtreze imaginea pentru a evidenția muchiile și contururile. Filtrele precum Sobel sau Canny sunt adesea utilizate în acest scop.
- **Morfologie matematică.** Tehnici precum transformările morfologice (eroziune, dilatare, deschidere, închidere) pot fi aplicate pentru a îmbunătăți detecția liniilor drepte, în special în condiții de iluminare sau zgromot variabil.

După detectarea liniilor drepte, pot fi aplicate diverse tehnici suplimentare pentru a îmbunătăți precizia sau robustețea rezultatelor, cum ar fi conectarea segmentelor de linie pentru a obține linii continue.

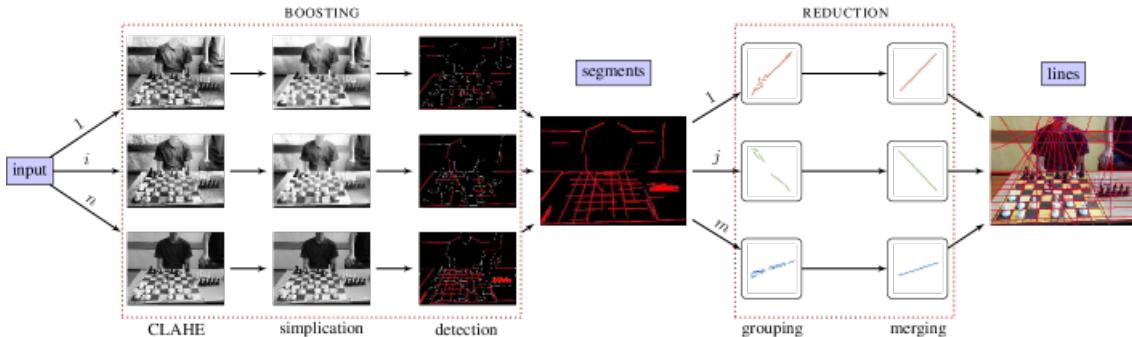


Figura 5.2: Preprocesarea imaginilor folosind patru seturi de parametri ($n=4$ de exemplu), fiecare dintre acestea fiind utilizat pentru a găsi o porțiune din segmentele de linie care sunt apoi împărțite în m grupuri și sunt unite în linii drepte (unde m indică numărul de grupuri de segmente coliniare și este setat automat de către algoritm). Maciej A. Czyzowski, Artur Laskowski, Szymon Wasik[4]

Figura prezintă un proces de procesare a imaginilor utilizând un algoritm de detectare a liniilor. Iată pașii detaliati:

- Input:** În prima coloană din stânga, sunt prezentate imaginile originale care sunt introduse în sistem.
- Segmente:** În a doua coloană, putem vedea segmentele de linie detectate în imagini folosind cei patru parametri diferiți. Aceste segmente sunt afișate cu roșu.
- Gruparea segmentelor:** Segmentele detectate sunt apoi grupate în m grupuri de segmente coliniare. Algoritmul determină automat numărul de grupuri (m) în funcție de coliniaritatea segmentelor detectate.
- Linii:** În ultima etapă, segmentele coliniare din fiecare grup sunt unite pentru a forma linii drepte complete. Aceste linii sunt prezentate în imagini separate în a treia coloană, iar rezultatul final cu liniile drepte suprapuse peste imaginea originală este prezentat în imaginea din extrema dreaptă.

Acest proces permite identificarea și unierea segmentelor de linie pentru a obține o reprezentare mai clară și mai precisă a liniilor drepte din imagine.

Procesul de detectare a liniilor drepte, ilustrat grafic în fig. 5.2, este împărțit în două etape mari: **boosting** și **reduction**.

În etapa de boosting, procesul începe cu aplicarea egalizării adaptive a histogramelor (CLAHE), care îmbunătățește contrastul și vizibilitatea liniilor în imagine. Urmează simplificarea imaginii prin aplicarea filtrului bilateral și algoritmului Canny pentru detectarea marginilor. Aceste operațiuni pregătesc imaginea pentru detectarea liniilor folosind transformata Hough, care identifică segmentele drepte din imagine.

În etapa de reducere, liniile detectate sunt supuse unei serii de operațiuni pentru a le rafina și simplifica. În primul rând, liniile sunt filtrate în funcție de lungimea și unghiul lor pentru a elimina segmente nedorite sau nesemnificative. Apoi, liniile sunt grupate

în segmente aproape coliniare, pregătindu-le pentru o îmbinare mai eficientă. În final, se aplică o operațiune de contopire pentru a uni segmentele de linii apropiate, obținând astfel liniile drepte finale și relevante pentru imaginea dată.

Detectarea marginilor

Pentru a îmbunătăți calitatea imaginii, este aplicat mai întâi filtrul bilateral. Acesta este util pentru reducerea zgomotului, menținând totodată marginile clare și bine definite în imagine. Apoi, se face calcularea pragurilor adaptate folosite de operatorul Canny. În cele din urmă, se utilizează detectorul de margini Canny pentru a identifica marginile din imagine, profitând de informațiile îmbunătățite furnizate de filtrul bilateral și de pragurile adaptate.

Procesul este următorul:

1. Calculează mediana imaginii img
2. Aplică filtrul bilateral imaginii img cu parametrii 7, 75, 75
3. Calculează pragul inferior $t1$ ca $\text{int}(\max(0, (1.0 - \alpha) \times \text{median}))$
4. Calculează pragul superior $t2$ ca $\text{int}(\min(255, (1.0 + \alpha) \times \text{median}))$
5. Detectează marginile Canny pe img cu pragurile $t1$ și $t2$

Aceste operațiuni pregătesc imaginea pentru etapele ulterioare ale procesării, cum ar fi identificarea și caracterizarea obiectelor sau aplicarea altor tehnici de analiză și prelucrare.

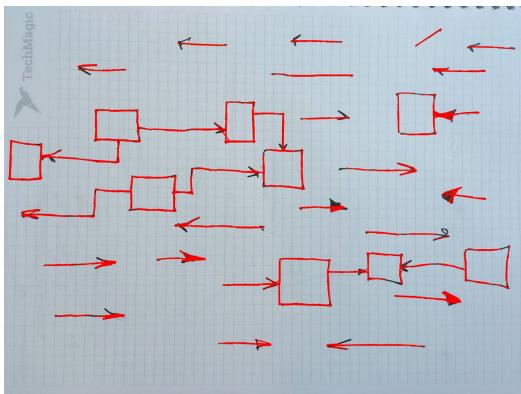
Detectare liniilor pe imagini CLAHE

După aplicarea tehnicii CLAHE (Contrast Limited Adaptive Histogram Equalization), rezultă o varietate de imagini prelucrate, fiecare cu ajustări specifice de contrast adaptiv. Numărul exact de imagini rezultate depinde de parametrii specifici folosiți în algoritmul CLAHE.

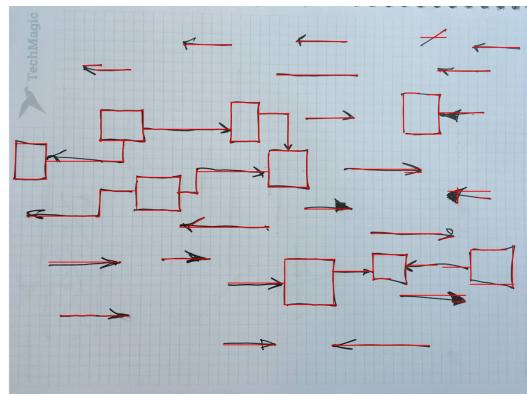
Pentru fiecare variantă a imaginii obținute prin CLAHE, continuăm procesul de detectare a liniilor drepte prin transformata Hough, care implică trei etape principale: detectarea marginilor utilizând, de exemplu, algoritmul Canny, calcularea acumulatorului Hough pentru a reprezenta posibilele liniile prin puncte într-un spațiu multidimensional și, în cele din urmă, identificarea liniilor drepte prin selectarea maximelor sau aplicarea unui prag pe acumulatorul Hough pentru a extrage liniile semnificative. Astfel, pentru fiecare imagine prelucrată, generăm o nouă serie de lini.

Pentru detectorul de table de șah, am utilizat multiple setări pentru algoritmul CLAHE, generând astfel mai multe imagini rezultate. Pentru a obține rezultatele finale, am grupat liniile detectate din fiecare imagine într-un set total, furnizând astfel o reprezentare amplă a liniilor în imaginea originală.

Această metodă este esențială pentru a asigura că sunt identificate liniile cu contrast redus și că acestea sunt vizibile în diferite condiții de iluminare și de textură a imaginii.



Înainte de a îmbina segmentele de linie (572 de linii)



După îmbinarea segmentelor de linie (89 de linii)

Figura 5.3: Înainte de a îmbina segmentele de linie (572 de linii) (Stânga) și După îmbinarea segmentelor de linie (89 de linii) (Dreapta). (<https://stackoverflow.com/a/70318827/12505886>) [8]

Filtrarea și îmbinarea liniilor

Algoritmul își propune să ofere o reprezentare mai clară și mai simplificată a liniilor din imagine, care poate fi utilă în aplicații precum detectarea contururilor, recunoașterea obiectelor sau navigarea pe bază de imagini. Este important de menționat că aceste modificări pot fi mai puțin vizibile pentru utilizatorul final, dar ele optimizează acuratețea și rapiditatea proceselor ulterioare de analiză a imaginilor. Prin reducerea numărului de linii și concentrarea pe liniile semnificative, algoritmul poate facilita identificarea și interpretarea obiectelor sau a caracteristicilor din imagine.

Algoritmul funcționează astfel:

1. Primește un set de liniî detectate folosind transformata Hough.
2. Separă liniile în grupuri orizontale și verticale:
 - Liniile orizontale sunt cele care sunt aproape de o direcție perfect orizontală, având o orientare în intervalul de, exemplu, 80 până la 100 de grade.
 - Liniile verticale sunt cele care sunt aproape de o direcție perfect verticală, având o orientare în intervalul de, exemplu, 0 până la 10 grade sau 170 până la 180 de grade.
3. Pentru fiecare grup:
 - Identifică liniile similare și le îmbină în grupuri distințe. Similaritatea poate fi determinată de criterii cum ar fi:
 - Distanța între liniile paralele.
 - Unghiul de înclinare al liniilor.
 - Suprapunerea între liniile sau segmentele acestora.
 - Algoritmii de grupare, cum ar fi algoritmul k-means sau DBSCAN, pot fi utilizati pentru a grupa liniile similare.
4. Pentru fiecare grup de liniî similare:
 - Unește segmentele într-o singură linie. Acest lucru poate implica:

- Extinderea segmentelor pentru a se întâlni sau a se suprapune cu altele din același grup.
 - Interpolarea sau ajustarea liniilor pentru a le uni într-o singură linie coerentă.
5. Returnează toate liniile îmbinate, adică liniile rezultate din procesul de îmbinare a segmentelor similare din fiecare grup.

Această abordare combină filtrarea inițială a liniilor pentru a le separe în grupuri orizontale și verticale, urmată de procesul de identificare și îmbinare a liniilor similare în fiecare grup pentru a obține liniile finale îmbinate.

5.1.2 Detectia punctelor de intersecție

Algoritmul de detectare punctele de intersecție a liniilor de pe o tablă de șah este un proces complex care utilizează tehnici de prelucrare a imaginilor și algoritmi de învățare automată pentru a identifica punctele de intersecție relevante.

Determinarea intersecțiilor dintre liniile detectate

După ce liniile pătratelor sunt detectate în imaginea reprezentând tabla de șah, următorul pas în algoritmul de determinare a intersecțiilor constă în calcularea punctelor de intersecție între aceste lini. Aceste intersecții sunt punctele în care se întâlnesc două sau mai multe lini, reprezentând colțurile sau nodurile pătratelor din cadrul tablei de șah.

Pentru a calcula aceste intersecții, se aplică o metodă matematică ce se bazează pe ecuațiile liniilor identificate[2]:

$$u_a = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \quad (5.1)$$

$$u_b = \frac{(x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \quad (5.2)$$

$$\text{denominator} = ((x_1 - x_2) \times (y_3 - y_4)) - ((y_1 - y_2) \times (x_3 - x_4)) \quad (5.3)$$

Aceste formule permit determinarea coordonatelor punctului de intersecție (x și y) între două segmente de linie definite de punctele $P_1(x_1, y_1)$ și $P_2(x_2, y_2)$, respectiv $P_3(x_3, y_3)$ și $P_4(x_4, y_4)$.

Parametrul denominator este utilizat pentru a evita împărțiri la zero și pentru a determina dacă segmentele de linie sunt paralele sau coliniare. Parametrii ua și ub trebuie să ia valori între 0 și 1 pentru ca intersecția să se afle pe segmentele liniilor, și nu pe prelungiri ale acestora.

În final, se folosesc următoarele formule pentru determinarea coordonatelor punctului de intersecție:

$$x = (x_1 + ua \times (x_2 - x_1)) \quad (5.4)$$

$$y = (y_1 + ua \times (y_2 - y_1)) \quad (5.5)$$

Filtrarea intersecțiilor detectate

Pentru a asigura acuratețea identificării punctelor de intersecție, se utilizează un model antrenat pentru a filtra punctele de intersecție detectate inițial. Acest model utilizează tehnici de învățare automată pentru a distinge între punctele de intersecție valide și cele care sunt rezultatul zgromotului sau a altor artefacte din imagine.

După detectarea inițială, un model de rețea neurală convezională (CNN) este folosit pentru a filtra și valida punctele de intersecție detectate. Modelul CNN este antrenat pe imagini de puncte de intersecție ale tablei de șah și pe alte modele care pot semăna cu puncte de intersecție.

Pentru fiecare punct de intersecție detectat, se aplică o serie de pași de procesare, care includ:

1. **Convertirea la tonuri de gri:** Imaginea color este transformată într-o imagine în tonuri de gri pentru a uniformiza informațiile și a simplifica prelucrarea ulterioară.
2. **Binarizarea:** Prin aplicarea unei tehnici de binarizare, imaginea monocromă este transformată într-o imagine binară, evidențiind astfel contrastul dintre obiectele de interes și fundal.
3. **Detectarea marginilor:** Algoritmul Canny este folosit pentru a identifica marginile și contururile din imagine.
4. **Redimensionarea:** Imaginea este redimensionată la dimensiunea specificată de intrare a detectorului neural, adică o matrice de 21x21 pixeli. Această etapă este crucială pentru a asigura coerența între dimensiunile datelor de intrare și structura modelului neural.

După aplicarea acestor procese de preprocesare, imaginea rezultată este pregătită pentru a fi introdusă în rețea neurală convezională (CNN). Rețea neurală produce două ieșiri cu valori între zero și unu, care sunt utilizate pentru clasificarea binară. În acest context, clasele indică dacă o matrice reprezintă un punct de rețea al tablei de șah sau nu.

Astfel, pentru fiecare punct de intersecție detectat pe imaginea originală, se efectuează o analiză individuală a fragmentului care îl conține, utilizând rețea neurală pentru a determina dacă este sau nu un punct de rețea al tablei de șah. Această abordare permite o identificare precisă a punctelor de rețea și elimină orice ambiguitate în clasificare.

Modelul CNN a fost antrenat pe câteva mii de imagini de puncte de intersecție ale tablei de șah și alte modele care pot să semene cu acestea (4,732 exemple pozitive și 4,933 exemple negative). Pentru a genera un set de date atât de amplu, s-a utilizat o procedură automatizată care modifică imagini artificiale ale unei table standard de șah 8x8 prin deformarea perspectivei în direcții aleatorii. Setul de date folosit are numele de LATCHESS21[7] și poate fi găsit pe platforma RepOD.

Punctele rezultate sunt apoi procesate printr-o procedură de grupare. Dacă două sau mai multe puncte sunt la o distanță mai mică sau egală cu 15 pixeli, acestea sunt înlocuite cu un singur punct situat în poziția din centrul lor, iar punctele inițiale sunt eliminate. În etapa finală, punctele extreme sunt eliminate. Este calculată valoarea mediană a distanței dintre puncte și sunt eliminate punctele care se află la o distanță mai mare de aceasta. Acest pas ajută la eliminarea punctelor care nu fac parte din grupul principal și care ar putea proveni din zgromotul imaginii sau din prezența altor table de șah în imagine.

Modelul CNN a fost antrenat utilizând biblioteca Keras și are următoarea structură:

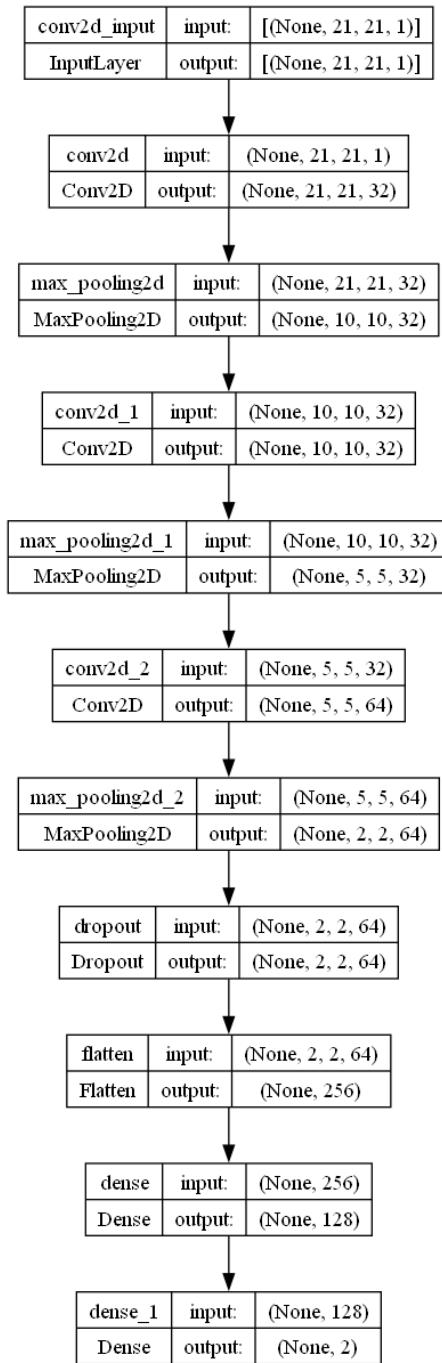


Figura 5.4: Structura modelului detectorului punctelor de intersecție. Diagramă generată cu funcția `utils.plot_model()` din biblioteca Keras.

5.1.3 Căutarea poziției tablei de șah

Modulul de Căutare a Poziției Tablei de Șah (CPS), denumit LLR în codul sursă, este o metodă dezvoltată de Maciej A. Czyzewski, Artur Laskowski și Szymon Wasik[4] pentru localizarea precisă a colțurilor unei table de șah într-o imagine, folosind liniile și punctele de intersecție rezultate din detecțiile anterioare.

Algoritmul CPS urmează mai multe etape:

1. **Identificarea grupului de puncte de intersecție detectate:** Algoritmul începe prin selectarea unui grup de puncte generate și filtrate anterior, care în mod obișnuit reprezintă tabla principală de șah din imagine. Acest grup este denumit G.
2. **Calcularea parametrilor:** Algoritmul calculează o valoare a, care aproximează lățimea unui pătrat pe tablă de șah, și determină centrul de masă al grupului G.
3. **Găsirea liniilor relevante:** Pentru fiecare punct de rețea al tablei de șah din grupul G, algoritmul identifică liniile generate de algoritmul prezentat anterior, care îndeplinește anumite criterii. Aceste criterii includ proximitatea față de punctele rețelei tablei de șah și față de centrul de masă al grupului, precum și probabilitatea de a fi în apropierea unei margini de ramă, determinată folosind o funcție de scor.
4. **Gruparea liniilor:** Liniile identificate sunt împărțite în grupuri orizontale și verticale, ținând cont de perspectivă.
5. **Formarea ramei tablei de șah și calculul scorului:** Perechile de linii din fiecare grup sunt folosite pentru a forma rame. Scorul fiecărei rame este calculat folosind o ecuație descrisă în articolul publicat de Maciej A. Czyzewski, Artur Laskowski și Szymon Wasik[4]. Această ecuație evaluează probabilitatea ca un poligon definit de patru linii să încadreze o tablă de șah, luând în considerare factori precum numărul de puncte din interiorul poligonului, distanțele acestora față de laturile și centrul poligoanelor, precum și aria poligoanelor.

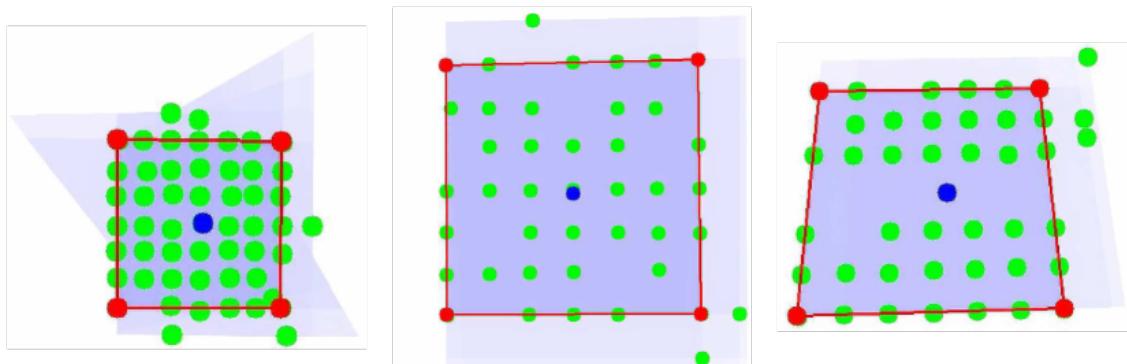


Figura 5.5: Scorurile sunt calculate pentru mai multe margini, iar rama cu cel mai mare scor este evidențiată cu o margine roșie și un fundal albastru închis. Imaginea ilustrează procesul de evaluare a probabilității încadrării unei table de șah în cadrul determinat de liniile identificate, Maciej A. Czyzewski, Artur Laskowski, Szymon Wasik[4].

În esență, modulul CPS analizează eficient datele din imagine pentru a determina probabilitatea ca o ramă definită de linii specifice să încadreze o tablă de șah, furnizând informații valoroase pentru procesarea ulterioară în sarcinile de recunoaștere a tablei de șah.

5.2 Detectia pieselor de șah

Modelul antrenat pentru detectia pieselor de șah primește ca intrare o imagine redimensionată de 416x416 pixeli. Este important de menționat că această redimensionare păstrează proporțiile originale ale imaginii, evitând astfel distorsiunile geometrice care ar putea afecta recunoașterea pieselor. Ca ieșire, modelul oferă o listă de casete de detectie, fiecare casetă având coordonatele exacte unde se află piesa pe imagine, precum și clasa piesei detectate (rege alb, rege negru, regină albă, regină neagră, nebun alb, nebun negru, cal alb, cal negru, turn alb, turn negru, pion alb, pion negru).

5.2.1 Set de date

Pentru a dezvolta un model robust de detectie a pieselor de șah, am început prin utilizarea unui set de date existent numit Chess Pieces Dataset[10], disponibil pe platforma Roboflow. Acest set de date conține 289 de imagini cu diverse stări ale tablei de șah, unde piesele sunt adnotate. Imaginile din acest set sunt capturate dintr-o singură perspectivă, ceea ce limitează variabilitatea scenariilor și unghiurilor posibile.

Pentru a îmbunătăți diversitatea setului de date și a asigura că modelul va fi capabil să detecteze piesele de șah în diverse condiții și din multiple perspective, am adăugat un set suplimentar de imagini. Am realizat un set de 350 de imagini cu o tablă și piese de șah reale, fotografiate din diferite unghiuri și distanțe față de tablă. Aceste imagini au fost adnotate corespunzător pentru a indica pozițiile și tipurile pieselor de șah.

În plus, pentru a extinde capacitatea modelului de a recunoaște piesele de șah și pe table digitale, am inclus 10 imagini ale unor table de șah 2D virtuale, capturate de pe platforme populare de șah online precum Chess.com și Lichess.com. Aceste imagini digitale asigură că modelul poate fi aplicat și în contextul jocurilor de șah online, unde designul și reprezentarea grafică a pieselor pot varia semnificativ.

5.2.2 Adnotare

Adnotarea imaginilor a fost un pas crucial în pregătirea setului de date pentru antrenarea modelului de detectie. Fiecare imagine a fost atent adnotată pentru a marca pozitia exacte ale pieselor de șah, folosind etichete specifice pentru fiecare tip de piesă și culoare. Pentru acest proces, am utilizat platforma Roboflow, care oferă unelte de adnotare asistată de inteligență artificială, redimensionare și augmentare, asigurând astfel acuratețea și consistența etichetelor aplicate. Procesul de augmentare a inclus rotirea imaginilor, ajustarea luminozității și a contrastului, precum și modificarea saturăției culorilor, rezultând imagini luminate mai mult sau mai puțin și decolorate sau contrastate.

Setul de date combinat rezultat conține un total de 649 de imagini, fiecare având adnotări detaliate care specifică locațiile și tipurile pieselor de șah prezente. Diversitatea unghiurilor și a distanțelor de fotografiere în setul de imagini reale, împreună cu imaginile digitale, oferă o bază solidă pentru antrenarea unui model robust și versatil. Împreună cu imaginile augmentate, setul de date a fost mărit considerabil, ajungând la un total

de 1557 de imagini(1362 de imagini pentru antrenare (88%), 130 de imagini pentru validare (8%) și 65 de imagini pentru testare (4%)).

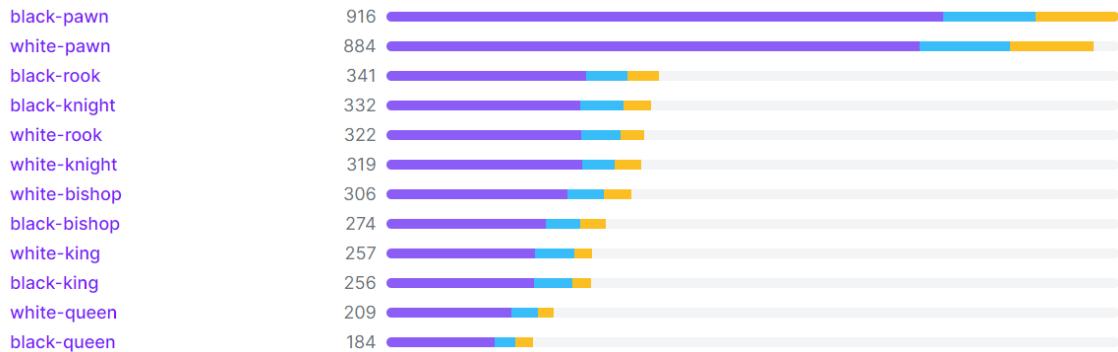


Figura 5.6: Reprezentarea numărului de adnotări pentru clasele din imagini. Culorile indică distribuția adnotărilor: mov pentru imaginile de antrenare, albastru pentru imaginile de validare și galben pentru imaginile de testare. (Un total de 4600 adnotări în cele 649 de imagini)

Conform reprezentării din fig. 5.6, se observă că adnotările pentru piesele de tip pion sunt semnificativ mai numeroase decât cele pentru celelalte piese. Această discrepanță poate sugera că modelul este susceptibil să identifice greșit pioni în loc de alte piese. Totuși, acest aspect este benefic, având în vedere că în jocul de șah, pionii reprezintă jumătate din totalul pieselor, oferind astfel modelului o perspectivă realistă asupra distribuției pieselor într-o partidă obișnuită.

5.2.3 Antrenare

Antrenarea modelului de detecție a pieselor de șah s-a desfășurat folosind arhitectura YOLO (You Only Look Once), versiunea 8, ultima versiune disponibilă la momentul începerei antrenării modelului. YOLOv8 este cunoscută pentru performanța sa în sarcini de detecție a obiectelor în timp real, oferind îmbunătățiri semnificative față de versiunile anterioare.

Am utilizat imagini din setul de date combinat, asigurând faptul că modelul a fost expus la o varietate largă de condiții și perspective. Modelul antrenat cu aceste imagini ar trebui să fie capabil să detecteze piese de șah atât pe o tablă reală, indiferent de unghiul sau distanța din care este capturată poza, atâtă timp cât imaginea este clară și bine iluminată, cât și pe table virtuale, cum sunt cele de pe platformele web de șah. Procesul de antrenare a inclus augmentarea datelor pentru a mări variabilitatea scenariilor prezentate modelului, sporindu-i astfel robustețea și capacitatea de generalizare.

Rezultatele preliminare indică o performanță promițătoare, cu modelul reușind să detecteze corect piesele de șah în majoritatea situațiilor testate, demonstrând astfel potențialul pentru aplicații practice în monitorizarea și analiza jocurilor de șah, desfășurate atât pe o tablă fizică, cât și pe una virtuală.

5.3 Detecția stării jocului de șah

Detectarea automată a pozițiilor pieselor de șah într-o imagine este esențială pentru diverse aplicații, inclusiv dezvoltarea de agenți de joc autonomi și sisteme de asistență pentru jucătorii de șah. În această secțiune, vom explora metoda utilizată pentru a

detecta pozițiile pieselor de șah într-o imagine digitală și generarea unei notări Forsyth–Edwards Notation (FEN) corespunzătoare.

Scopul principal al acestui proces este de a localiza și identifica pozițiile fiecărei piese de șah prezente într-o imagine digitală. Acest lucru este crucial pentru a reprezenta corect starea jocului de șah și pentru a putea lua decizii corespunzătoare în cadrul unui sistem de asistență pentru jucători sau a unui motor de șah.

5.3.1 Utilizarea detectoarelor tablei și pieselor de șah

Imaginea de intrare este redimensionată pentru a asigura uniformitatea în prelucrare. Apoi, folosind un detector specializat pentru tabla de șah, care are ca ieșire o imagine transformată geometric în care colțurile tablei de șah corespund cu colțurile imaginii. Împărțirea imaginii în 8x8 pătrate egale reprezintă tabla de șah.

Folosind detectorul pieselor de șah, identificăm fiecare piesă prezentă pe tabla de șah. Pentru fiecare piesă detectată, determinăm clasa acesteia și o poziționăm corespunzător pe tabla de șah, folosind o transformare adecvată a coordonatelor. Pentru a determina unde se află fiecare piesă pe grilă, punctul din mijlocul marginii inferioare al cadrului de detectie (punctul centrat între punctele de jos din stânga și din dreapta) a fost folosit ca locație a piesei pe imaginea originală. Acest lucru se datorează faptului că atunci când este privit dintr-un unghi, vârfurile pieselor se suprapun adesea sau cad în alte pătrate de pe grilă. Alegând partea de jos a piesei, locația piesei va fi întotdeauna direct pe pătratul pe care piesa stă în realitate.

5.3.2 Postprocesare și validare

Pentru a alege poziția potrivită pentru o piesă de șah într-o imagine, trebuie să ne asigurăm că aceasta se potrivește cu grila de pe tablă, chiar dacă imaginea în sine este transformată. Putem rezolva această problemă transformând coordonatele punctelor selectate pentru piese folosind aceeași matrice de transformare aplicată imaginii. Astfel, vom putea compara pozițiile pieselor cu grila transformată pentru a determina exact unde se află acestea pe tablă. Cu ajutorul funcției `cv2.transform()` din OpenCV, putem realiza această transformare pentru fiecare punct detectat, facilitând astfel determinarea poziției corecte a fiecărei piese pe tablă.

Pentru a ne asigura că detecția este corectă, monitorizăm numărul maxim de piese permise pentru fiecare tip de piesă. Dacă acest număr a fost deja atins, atunci când detectăm o nouă piesă, o vom decupa din imagine conform cadrului detectat și vom încerca din nou identificarea tipului de piesă. Această abordare ne permite să gestionăm cazurile în care o piesă poate fi detectată de mai multe ori sau când există prea multe piese de același tip pe tablă. Astfel, putem asigura că procesul de detectare și identificare a pieselor de șah este precis și eficient.

5.3.3 Generarea notării FEN

În ceea ce privește generarea notării Forsyth–Edwards (FEN), aceasta reprezintă o parte crucială a procesului de detectare a stării jocului de șah. FEN este o reprezentare compactă a poziției pieselor pe tabla de șah, care poate fi utilizată pentru a descrie complet starea jocului. Această notare conține informații despre poziția fiecărei piese, cine urmează să mute, starea rocadelor, starea en passant și numărul de mutări fără captură sau mutări de pion.

Este important de menționat că în contextul detectării pozițiilor pieselor de șah dintr-o imagine digitală, nu avem acces la informații despre cine urmează să mute, starea rocadelor, starea en passant sau numărul de mutări fără captură sau mutări de pion. Aceste detalii sunt specificate în timpul jocului de șah real, dar nu sunt direct observabile într-o imagine statică a tablei de șah.

Prin urmare, în procesul detectorului de generare a notării Forsyth–Edwards (FEN) dintr-o imagine digitală, acesta se concentrează exclusiv pe reprezentarea pozițiilor pieselor pe tabla de șah. Nu include informații despre cine urmează să mute sau alte detalii despre starea jocului, deoarece acestea nu sunt disponibile sau relevante în contextul dorit.

Generarea notării FEN se realizează pe baza pozițiilor pieselor detectate pe tabla de șah. După ce am identificat tipurile și pozițiile pieselor, construim un sir de caractere conform convențiilor FEN.



Figura 5.7: Câmpul de plasare a piesei FEN pentru această poziție este „r1bk3r/p2pBpNp/n4n2/1p1NP2P/6P1/3P4/P1P1K3/q5b1”, [https://www.chess.com/terms/fen-chess\[3\]](https://www.chess.com/terms/fen-chess[3]).

Pentru a face acest lucru, parcurgem fiecare rând al tablei de șah, de la rândul 8 la rândul 1, și pentru fiecare pătrat de pe rând, determinăm tipul piesei. Utilizând convențiile FEN, fiecare tip de piesă este reprezentat printr-un caracter specific: 'K' pentru rege alb, 'Q' pentru regină albă, 'k' pentru rege negru, 'q' pentru regină neagră, 'R' pentru turn alb, 'r' pentru turn negru, 'N' pentru cal alb, 'n' pentru cal negru, 'B' pentru nebun alb și 'b' pentru nebun negru. Spațiile goale sunt reprezentate de numărul de pătrate libere consecutive. Astfel, fiecare rând al tablei este transformat într-un segment corespunzător din notația FEN.

După ce am construit segmentele pentru fiecare rând al tablei, le combinăm, adăugând și informații despre cine urmează să mute, starea rocadelor, starea en passant și numărul de mutări fără captură sau mutări de pion.

Capitolul 6

Aplicația tip multi-platformă

În era digitală modernă, utilizatorii așteaptă ca aplicațiile să fie disponibile pe diverse dispozitive și sisteme de operare, de la telefoane mobile la computere personale. O aplicație multi-platformă (sau cross-platformă) este o soluție software care este proiectată pentru a funcționa pe mai multe platforme, cum ar fi Android, iOS, Windows și macOS, utilizând un singur cod de bază. Aceasta înseamnă că dezvoltatorii pot scrie codul o singură dată și îl pot implementa pe multiple sisteme de operare, economisind astfel timp și resurse.

Avantajele aplicațiilor multi-platformă sunt numeroase. Ele permit o lansare mai rapidă pe piață, reduc costurile de dezvoltare și întreținere și asigură o experiență unificată pentru utilizatori, indiferent de dispozitivul pe care îl folosesc. Tehnologii precum React Native, Flutter și Xamarin sunt printre cele mai populare pentru dezvoltarea aplicațiilor multi-platformă, fiecare având propriile avantaje și particularități.

6.1 Aplicabilitatea aplicației Chess Snapshot ca soluție multi-platformă

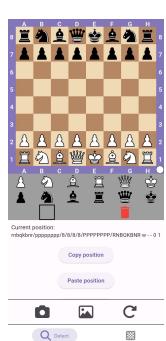


Figura 6.1: Android

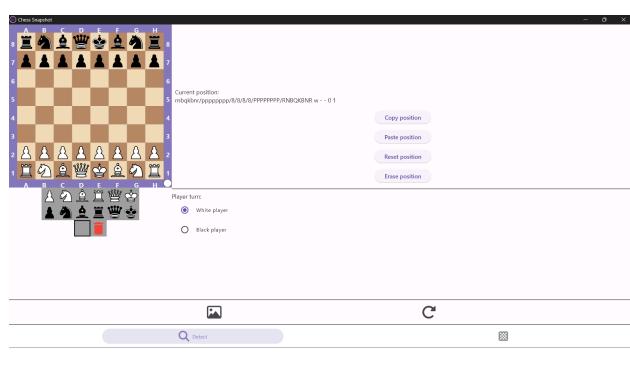


Figura 6.2: Windows

Chess Snapshot este un exemplu excelent de aplicație multi-platformă. Aceasta a fost proiectată pentru a funcționa atât pe dispozitivele mobile cu sistem de operare Android, cât și pe computerele care rulează Windows. Utilizând tehnologii moderne de dezvoltare, Chess Snapshot oferă utilizatorilor o experiență fluidă și consistentă indiferent de platformă.

6.2 Back-end-ul aplicației

Back-end-ul aplicației Chess Snapshot este construit utilizând Flask, un cadru web micro scris în Python. Flask este renumit pentru simplitatea și flexibilitatea sa, fiind o alegere excelentă pentru dezvoltarea rapidă și eficientă a aplicațiilor web. În contextul Chess Snapshot, Flask joacă un rol crucial în intermedierea între detectorul stării jocului de șah și aplicația front-end dezvoltată în Flutter.

Această arhitectură asigură o comunicare eficientă și rapidă între componentele aplicației, permitând utilizatorilor să beneficieze de funcționalități avansate de analiză și recomandare în timp real. Flask, prin natura sa modulară și ușor de extins, oferă o platformă robustă și scalabilă pentru gestionarea logicii aplicației și pentru furnizarea serviciilor esențiale către utilizatori.

În acest caz, se utilizează un REST API (Representational State Transfer Application Programming Interface), un set de reguli și convenții pentru a construi și a interacționa cu servicii web. Un REST API utilizează metode HTTP standardizate, precum GET, POST, PUT și DELETE, pentru a permite comunicarea între client și server. Într-un context RESTful, resursele sunt identificate prin URL-uri și pot fi reprezentate în diverse formate, cum ar fi JSON sau XML. Acest stil arhitectural este preferat pentru scalabilitatea și ușurința sa de utilizare, fiind larg adoptat pentru dezvoltarea aplicațiilor web moderne.

În cadrul aplicației Chess Snapshot, există două endpoint-uri principale în REST API-ul oferit de Flask:

- Detectia stării jocului de șah:** Acest endpoint procesează o imagine a unei table de șah și returnează starea curentă a jocului.
- Determinarea unei mutări optime pentru o stare de joc:** Acest endpoint primește o stare a jocului și returnează cea mai bună mutare posibilă în acea situație.

6.2.1 Detectia stării jocului de șah

Unul dintre endpoint-urile esențiale ale back-end-ului aplicației Chess Snapshot este `@app.route('/api/get_chess_position', methods=['POST'])`. Acest endpoint permite utilizatorilor să trimită o imagine a unei table de șah și să primească înapoi reprezentarea FEN (Forsyth-Edwards Notation) a poziției detectate pe tablă. Reprezentarea FEN este o modalitate standard de a descrie o poziție de șah și este utilizată pe scară largă în aplicațiile și bazele de date de șah.

Pașii procesului sunt:

- Verificarea existenței imaginii în cererea POST.** Se verifică dacă cererea POST include un fișier de imagine. Dacă nu, serverul returnează un mesaj de eroare și un cod de stare 400 (Bad Request).
- Citirea și decodificarea imaginii.** Fișierul de imagine este citit în format binar și apoi transformat într-un șir NumPy. Acest șir este decodificat folosind OpenCV pentru a obține imaginea originală în format color.
- Detectarea poziției de șah.** Un obiect `ChessPositionDetector` analizează imaginea și detectează poziția pieselor de șah. Metoda de detectare returnează poziția în format FEN.
- Returnarea rezultatului.** Poziția FEN este returnată ca răspuns JSON, permitând front-end-ului să o prelucreze și să o afișeze utilizatorului.

6.2.2 Determinarea unei mutări optime pentru o stare de joc

Un alt endpoint relevant în aplicație este `@app.route('/api/get_best_move', methods=['POST'])`. Acest endpoint permite utilizatorilor să trimită o poziție de șah în format FEN și să primească înapoi cea mai bună mutare calculată pentru acea poziție. Pentru a determina cea mai bună mutare, se folosește motorul de șah Stockfish, care analizează poziția și sugerează o mutare optimă.

Stockfish este unul dintre cele mai puternice și populare motoare de șah open-source disponibile în prezent. Este scris în C++, ceea ce îl face rapid și eficient în analiza pozițiilor de șah. Stockfish este bazat pe un algoritm de căutare a arborilor de joc și folosește o varietate de tehnici avansate pentru a evalua pozițiile și a determina cea mai bună mutare posibilă într-o anumită situație de joc.

Pașii procesului sunt:

- Obținerea datelor de intrare.** Serverul primește un obiect JSON în cererea POST, care conține informația despre poziția actuală a pieselor în format FEN.
- Calcularea celei mai bune mutări.** Motorul Stockfish este utilizat pentru a analiza poziția primită. Aceasta calculează cea mai bună mutare posibilă în funcție de poziția curentă a pieselor.
- Returnarea rezultatului.** Cea mai bună mutare calculată este returnată ca răspuns JSON de forma `{best_move: best_move}`, permitând front-end-ului să o prelucreze. Cea mai bună mutare calculată este returnată ca un sir de caractere format din patru litere, reprezentând poziția de plecare și de sosire a piesei. De exemplu, "e2e4" indică mutarea unei piese de pe poziția e2 pe poziția e4.

6.2.3 Publicarea API-ului pe internet

Pentru a face API-ul Chess Snapshot disponibil pe internet, am explorat mai multe platforme de hosting, precum Azure, PythonAnywhere și Render. Cu toate acestea, am constatat că planurile gratuite oferite de aceste platforme nu oferă suficientă capacitate de procesare și stocare pentru a susține aplicația în mod eficient.

Astfel, am optat pentru utilizarea platformei ngrok, care folosește un protocol de tunel ce permite ca o adresă locală a serverului Flask să fie accesibilă online. Iată pașii pe care i-am urmat pentru a face acest lucru:

- Execuția serverului:** Serverul Flask este executat local. Acest lucru se face prin rularea comenții `python app.py` în terminal.
- Utilizarea ngrok pentru a face serverul accesibil online:** Este apoi executată comanda `ngrok http --domain=knowing-fit-poodle.ngrok-free.app 8080` în terminal. Această comandă inițiază un tunel către serverul local pe portul 8080 și face aplicația accesibilă online prin intermediul unui URL static generat de ngrok.
- Accesarea API-ului online:** Ngrok ne furnizează un URL care poate fi folosit pentru a accesa API-ul online. Astfel, oricine poate trimite cereri către endpoint-urile noastre prin intermediul acestui URL.

Utilizând ngrok, API-ul Chess Snapshot poate să fie accesibil online într-un mod simplu și eficient, fără dezavantajul plății unui plan de hosting costisitor.

6.3 Front-end-ul aplicației

Pentru aplicația Chess Snapshot, accentul principal se pune pe interfața utilizatorului și pe o interacțiune fluidă. Este esențial să avem o prezentare clară a informațiilor despre starea jocului de șah detectată, dar și să oferim utilizatorului opțiuni intuitive pentru a juca și a analiza jocurile.

Aplicația Chess Snapshot are două pagini principale: Detect și Play. Navigarea între acestea este ușoară prin intermediul unei bare de navigație situată în partea de jos a aplicației. Această bară conține două taburi realizate cu ajutorul pachetului persistent_bottom_nav_bar_v2.



Figura 6.3: Chess Snapshot - Bara de navigație persistentă

6.3.1 Pagina Detect

Pagina Detect este o componentă esențială a sistemului nostru dedicat șahului, oferind funcționalități avansate de interpretare a stării unei table de șah dintr-o imagine și de editare a poziției rezultate. Această pagină permite utilizatorilor să efectueze următoarele acțiuni:

- **Vizualizare a stării curente a tablei de șah.** Utilizatorii pot vizualiza poziția actuală a pieselor pe tablă, reprezentată în format FEN.
- **Copierea poziției în clipboard.** Pagina Detect permite utilizatorilor să copieze FEN-ul poziției curente în clipboard, pentru a fi utilizat ulterior în alte aplicații sau scopuri.
- **Editarea poziției.** Oferă o tablă de șah editabilă, permitând utilizatorilor să modifice poziția pieselor pe tablă conform preferințelor lor. Această funcționalitate este utilă în cazul în care este necesară corectarea sau ajustarea poziției detectate automat.
- **Inserarea unei poziții din clipboard.** Utilizatorii pot insera o poziție anterior copiată în clipboard sub formă de FEN în tablă, ușurând astfel introducerea unei poziții cunoscute sau salvate anterior.

6.3.2 Bara de acțiuni a paginii Detect

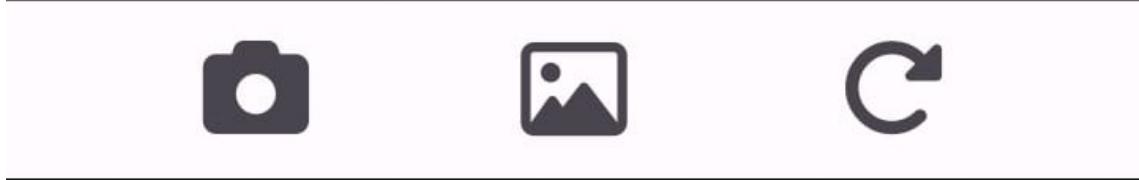


Figura 6.4: Chess Snapshot - Bara de acțiuni a paginii Detect

În partea de jos a paginii, deasupra barei de navigație, se găsesc trei butoane reprezentate prin pictograme. Acestea oferă funcționalități principale pentru detectarea pozițiilor pieselor pe tabla de șah.

Capturarea unei imagini

Acest buton, reprezentat printr-o cameră foto, deschide aplicația de fotografiere a telefonului pentru a permite utilizatorului să fotografieze tabla de șah. Imaginea capturată este apoi transmisă către server pentru detectarea stării jocului. Răspunsul primit sub formă de FEN poate fi vizualizat în aplicație, atât în format text, cât și pe o tablă virtuală. Această funcționalitate este disponibilă exclusiv pe dispozitivele mobile.

Selectarea unei imagini din galeria foto

Butonul reprezentat printr-o pictogramă de galerie foto permite utilizatorului să selecteze o imagine din galeria telefonului. Imaginea selectată este apoi transmisă către server pentru a fi procesată similar cu cea capturată prin intermediul camerei foto.

Rotirea poziției pieselor de pe tablă cu 90 de grade

Acest buton permite rotirea tuturor pieselor de pe tabla de șah cu 90 de grade spre dreapta. Această funcționalitate este utilă în situații în care imaginea capturată este rotită sau este fotografiată dintr-o poziție laterală a tablei, permitând utilizatorului să alinieze piesele în poziții corecte conform notărilor pătratelor.

6.3.3 Lista de funcționalități a paginii Detect

Pe pagina Detect există diverse funcționalități care permit utilizatorului să interacționeze și să modifice pozițiile pieselor de șah detectate sau create manual.



Figura 6.5: Chess Snapshot - Lista de funcționalități a paginii Detect

Reprezentarea FEN a poziției curente

În pagina Detect, utilizatorul poate vedea reprezentarea FEN (Forsyth-Edwards Notation) a poziției curente de pe tabla editabilă. Această reprezentare se actualizează automat de fiecare dată când sunt adăugate sau șterse piese de pe tablă sau când se

face o detectare a stării jocului dintr-o imagine. Aceasta oferă o modalitate rapidă și exactă de a comunica și salva poziția pieselor.

Copierea poziției în clipboard

Un buton numit *Copy position* permite copierea reprezentării FEN a poziției curente în clipboard. Aceasta facilitează partajarea rapidă a poziției sau utilizarea acesteia în alte aplicații sau servicii care acceptă formatul FEN.

Inserarea unei poziții din clipboard

Un alt buton, denumit *Paste position*, permite inserarea unei poziții FEN din clipboard. Dacă utilizatorul a copiat o reprezentare FEN în clipboard, această poziție poate fi inserată pe tabla editabilă prin apăsarea butonului, actualizând astfel vizualizarea și poziția pieselor conform datelor inserate.

Resetarea poziției

Butonul *Reset position* permite resetarea poziției pe tabla editabilă la configurația inițială de început a unui joc de șah. Aceasta rearanjează piesele pe tablă conform pozițiilor standard de start, oferind utilizatorului posibilitatea de a începe o nouă configurare sau analiză de la zero.

Stergerea poziției

Butonul denumit *Erase position* permite ștergerea tuturor pieselor de pe tabla editabilă. Aceasta resetează complet tabla, lăsând-o goală, și oferă utilizatorului flexibilitatea de a crea sau analiza noi poziții fără a fi nevoie să mute fiecare piesă individual în afara tablei.

Selecția rândului jucătorului

O opțiune permite utilizatorului să selecteze al cui este rândul să mute în poziția curentă. Aceasta este o setare importantă, mai ales în analiza jocurilor sau configurarea pozițiilor pentru jocul efectiv, și poate fi ajustată ușor pentru a reflecta corect situația de joc.

6.3.4 Tabla de șah editabilă

Tabla de șah editabilă este un instrument esențial pe pagina Detect, oferind utilizatorilor posibilitatea de a vizualiza și modifica starea detectată a unei poziții de șah. Aceasta este implementată pentru a oferi o interacțiune intuitivă și flexibilă cu pozițiile pieselor.

Vizualizarea și editarea stării detectate

Tabla editabilă afișează poziția detectată de aplicație dintr-o imagine cu o tablă de șah 2D sau 3D. Dacă există erori în detectare, acestea pot fi corectate de utilizator prin intermediul pieselor aflate sub tabla principală. Utilizatorul poate selecta o piesă și alege pătratul pe care dorește să o plaseze, corectând astfel eventualele greșeli de detectare.

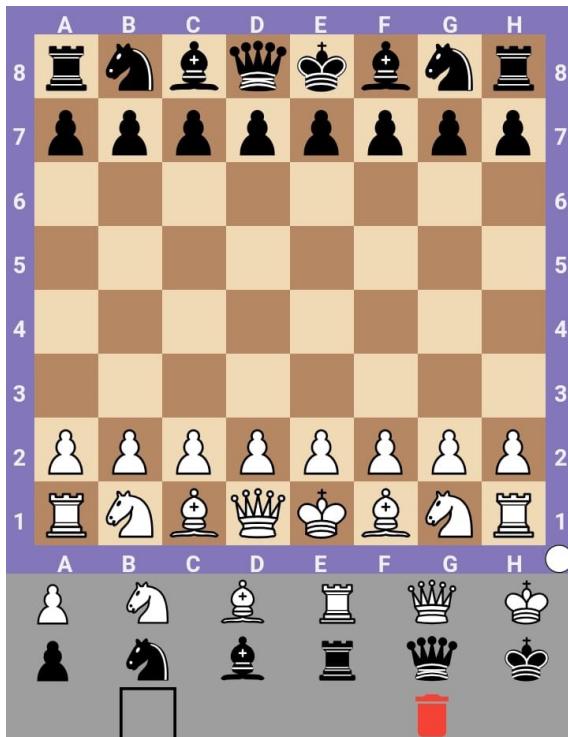


Figura 6.6: Chess Snapshot - Tabla de șah a paginii Detect

Interfața de utilizare

Utilizatorul poate selecta orice piesă din lista pieselor disponibile aflată sub tabla principală și o poate plasa pe orice pătrat de pe tablă. Această funcționalitate permite o modificare rapidă și precisă a poziției pieselor, oferind flexibilitate în configurarea și corectarea pozițiilor de joc.

Inițierea unui joc de șah de la o poziție detectată

Dacă poziția de pe tabla editabilă este corectă și utilizatorul dorește să înceapă un joc de șah de la acea poziție, poate accesa pagina Play. De aici, jocul poate continua cu poziția configurată pe tabla editabilă, permitând utilizatorului să înceapă un joc de șah de la orice configurație dorită.

Implementare tehnică

Pentru această tablă, s-a folosit și modificat pachetul `editable_chess_board` pentru a se potrivi cât mai bine cerințelor aplicației Chess Snapshot. Modificările aduse permit o interacțiune mai specifică funcționalităților de detectare și editare a pozițiilor de șah.

6.3.5 Pagina Play

Pagina Play reprezintă inima experienței de joc în cadrul sistemului dedicat șahului, oferind utilizatorilor posibilitatea de a juca partide de șah împotriva altor jucători sau împotriva calculatorului. Această pagină oferă funcționalități esențiale pentru gestionarea jocului și analiza pozițiilor.

- **Încărcarea și validarea poziției detectate.** Utilizatorii pot încărca poziția detectată anterior de pe pagina Detect. Această poziție este apoi validată pentru a se

asigura că respectă regulile șahului (de exemplu, prezența unui rege pentru fiecare culoare și poziționarea corectă a pieselor).

- **Joc local cu altă persoană sau calculatorul.** Pagina Play permite utilizatorilor să înceapă o partidă de șah locală cu alt jucător sau cu calculatorul. Interfața oferă instrumente pentru a gestiona fluxul jocului, inclusiv mutări valide și promovarea pionilor.
- **Vizualizarea celei mai bune mutări.** Utilizatorii pot solicita afișarea celei mai bune mutări în poziția curentă. Această funcționalitate este utilă pentru îmbunătățirea abilităților de joc și pentru a primi sugestii în timpul partidelor.
- **Analiza jocului pe platforma Lichess.** Utilizatorii au opțiunea de a analiza jocul pe platforma Lichess, pornind de la poziția curentă de pe tabla virtuală. Acest lucru le permite să acceseze instrumente avansate de analiză și să își îmbunătățească strategiile de joc.

Pagina Play oferă o experiență interactivă pentru pasionații de șah, combinând aspecte practice ale jocului cu posibilități de analiză și îmbunătățire a abilităților.

6.3.6 Bara de acțiuni a paginii Play

În partea de sus a paginii Play, sub bara de navigație, se găsesc trei butoane reprezentate prin pictograme, oferind funcționalități esențiale pentru gestionarea jocului de șah.



Figura 6.7: Chess Snapshot - Bara de acțiuni a paginii Play

Schimbarea perspectivei

Primul buton, ilustrat printr-o săgeată rotită, permite schimbarea perspectivei tablei de șah. Dacă perspectiva implicită este din poziția albă, apăsarea acestui buton va roti tabla astfel încât perspectiva să fie din poziția neagră și invers. Această funcționalitate ajută jucătorii să vadă jocul din ambele perspective.

Încărcarea poziției FEN

Al doilea buton, reprezentat printr-o pictogramă de încărcare, facilitează încărcarea unei poziții FEN (Notarea Forsyth-Edwards) direct de pe pagina de detectare. Dacă poziția FEN furnizată este validă, tabla de joc va fi actualizată conform poziției specificate, permitând jucătorului să continue jocul într-o anumită configurație.

Analiza jocului

Ultimul buton, ilustrat printr-un simbol de ochi lup, direcționează utilizatorul către o pagină web a platformei Lichess, unde este încărcată poziția curentă de pe tablă virtuală. Aici, utilizatorii pot analiza mai în detaliu jocul folosind instrumentele și funcționalitățile oferite de platforma Lichess.

6.3.7 Lista de funcționalități a paginii Play

Pagina Play oferă câteva funcționalități esențiale care îmbunătățesc experiența utilizatorului și îi oferă mai multe opțiuni de joc. Aceste funcționalități sunt reprezentate prin butoane de tip toggle și sunt descrise mai jos:

Play against a bot:



Show the best move:



Play as:



White



Black

Figura 6.8: Chess Snapshot - Lista de funcționalități a paginii Play

Jocul contra unui robot

Un buton de tip toggle permite utilizatorului să aleagă dacă dorește să joace contra unui robot sau nu. Activând acest buton, utilizatorul poate selecta ca oponent un algoritm de joc automatizat, ceea ce îi permite să practice și să-și îmbunătățească abilitățile de șah fără a avea nevoie de un partener uman.

Afișarea celei mai bune mutări

Un alt buton de tip toggle oferă posibilitatea de a afișa cea mai bună mutare în poziția actuală. Când această opțiune este activată, pe tablă va apărea o săgeată care indică mutarea considerată optimă conform algoritmului de analiză al jocului. Această funcționalitate este utilă pentru începători și pentru cei care doresc să învețe și să înțeleagă mai bine jocul de șah.

Alegerea culorii la jocul contra unui robot

O opțiune importantă pentru utilizatorii care joacă împotriva unui robot este posibilitatea de a alege culoarea pieselor. Utilizatorul poate selecta dacă dorește să joace cu piesele albe sau negre. Dacă opțiunea "White" este aleasă, robotul va juca automat toate mutările pentru culoarea negru și invers. Această setare permite utilizatorului să își configureze preferințele de joc și să experimenteze diferite scenarii și deschideri de joc.

6.3.8 Tabla de șah pentru jocul local

Tabla de șah prezentă pe pagina Play este locul unde utilizatorul poate juca fie local, împotriva unui alt jucător prezent fizic, fie contra unui robot. Această tablă oferă o experiență interactivă și intuitivă prin utilizarea funcționalității de tragere și plasare(drag and drop) pentru mutarea pieselor.

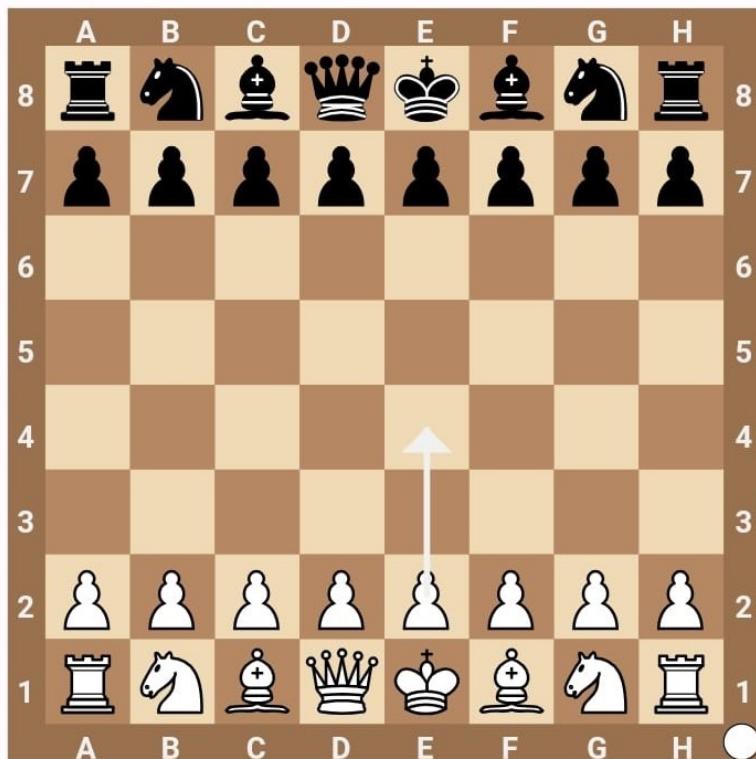


Figura 6.9: Chess Snapshot - Tabla de șah a paginii Play

Mecanismul de mutare a pieselor

Pe această tablă, mutările se fac printr-un simplu drag and drop. Utilizatorul poate prinde o piesă și o poate muta pe orice patrat valid al tablei. Aceasta simulează cât mai realist posibilitatea de a muta fizic piesele pe o tablă reală, oferind o experiență familiară și naturală.

Afișarea celei mai bune mutări

În cazul în care opțiunea de afișare a celei mai bune mutări este activată, cea mai bună mutare va fi indicată printr-o săgeată albă pe tablă. Această săgeată indică direcția și destinația mutării, oferind utilizatorului o sugestie vizuală despre cea mai bună mișcare în contextul poziției curente.

Implementare tehnică

Pentru implementarea acestei table de șah, s-a utilizat pachetul publicat pe pub.dev `simple_chess_board`[9]. Acest pachet oferă funcționalitățile de bază necesare pentru a desena o tablă de șah și pentru a permite interacțiunea prin mutarea pieselor. Prin integrarea acestui pachet, aplicația asigură o experiență fluidă pentru utilizatori, indiferent de dispozitivul folosit.

Capitolul 7

Testare

În această secțiune este prezentată testarea algoritmilor pentru detectarea tablei de șah, a punctelor de intersecție și a pieselor de șah. Aceste teste sunt esențiale pentru a valida performanța și precizia modelelor, asigurând faptul că acestea funcționează corect în condiții variate și provocatoare.

7.1 Testarea detecției punctelor de intersecție

Am testat modelul CNN pentru detectarea punctelor de intersecție de pe tabla de șah utilizând setul de date LATCHESS21. Acesta conține mii de imagini cu puncte de intersecție ale tablei de șah, atât exemple pozitive, cât și negative, create prin deformarea artificială a imaginii tablei de șah standard.

7.1.1 Rezultatele funcției de activare pe parcursul epocilor

Am monitorizat evoluția acurateței și erorii pe parcursul a 50 de epoci. Acest proces a fost esențial pentru evaluarea performanței și pentru ajustarea parametrilor rețelei neurale.



Figura 7.1: Evoluția acurateței și erorii pe parcursul a 50 de epoci

Pe măsură ce modelul a fost antrenat, am observat următoarele:

- **Acuratețea.** Graficul arată o creștere rapidă a acurateței în primele epoci, atât pentru setul de date de antrenare, cât și pentru cel de validare. Această creștere sugerează că modelul învață repede caracteristicile relevante ale datelor de antrenare. După primele 10 epoci, creșterea devine mai lentă, dar acuratețea continuă să crească până la un nivel stabil de peste 97% pentru ambele seturi de date.
- **Eroarea (Loss).** Pe măsură ce acuratețea crește, eroarea (loss) scade. Observăm o scădere semnificativă a erorii în primele 20 de epoci, după care ritmul scăderii devine mai lent. Cu toate acestea, eroarea continuă să scadă până la un nivel redus, sub 0.05 pentru setul de date de antrenare și sub 0.15 pentru cel de validare.

Aceste rezultate sugerează că modelul CNN s-a antrenat eficient și a atins o performanță excelentă pe setul de date LATCHESS21, fără semne de supraantrenare sau subantrenare.

7.1.2 Puncte de intersecție filtrate prin această detecție pe imagini cu table de șah



Figura 7.2: Evoluția acurateței și erorii pe parcursul a 50 de epoci

După antrenarea modelului, acesta a fost testat pe imagini reale și virtuale ale tablelor de șah. Rezultatele obținute au demonstrat o detectare și filtrare precisă a punctelor de intersecție formate de pătratele tablei de șah. Modelul a reușit să identifice cu succes punctele relevante, contribuind astfel la o detectie eficientă și precisă a tablei de șah în imagini. Acest lucru sugerează că algoritmul dezvoltat este robust și adekvat pentru utilizare în cadrul aplicației Chess Snapshot.

7.2 Testarea detecției tablei de șah

Pentru a evalua algoritmul nostru, am pregătit un set de date de referință provocator, care conține imagini ale tablelor de șah captureate în condiții variate și dificile. Pentru a colecta imagini ce conțin table de șah și piese de șah dificil de recunoscut, am stabilit următoarele condiții pentru imaginile de referință:

- **Prezența altor obiecte în imagine.** Imaginile includ nu doar tabla și piesele de șah, ci și alte obiecte care pot induce în eroare algoritmul.
- **Linii drepte generate de alte obiecte.** În imagini, liniile drepte nu sunt generate doar de tabla de șah, ci și de alte obiecte, complicând astfel detectarea corectă.
- **Tabele imperfecte.** Tabla de șah nu umple întreg cadrul imaginii și nu este evidențiată în mod special, astfel încât colțurile și piesele de șah nu sunt accentuate.
- **Condiții de iluminare și mediu.** Imaginile includ umbre, reflexii, distorsiuni și zgomot, adăugând un nivel suplimentar de dificultate pentru algoritmul de detectare.

Numărul imaginii	Detectare corectă
1	Nu
2	Da
3	Da
4	Da
5	Da
6	Partial
7	Partial
8	Partial
9	Da
10	Da

Tabela 7.1: Rezultatele testării detecției tablei de șah pe 10 imagini de referință

Aceste imagini se regăsesc în proiectul chess-position-detector în folderul test/paper/, numerotate cu p1, p2, etc. Rezultatele sunt în folderul test/out/paper/.

Rezultatele testării arată că algoritmul nostru a obținut o rată ridicată de detecție corectă, demonstrând capacitatea sa de a funcționa eficient chiar și în condiții dificile. În cele mai multe cazuri, algoritmul a reușit să identifice corect tabla de șah, chiar și atunci când aceasta era parțial obstrucționată sau când exista un zgomot semnificativ în imagine. Totuși, în câteva imagini, detectarea a fost doar parțial corectă, indicând posibilitatea de îmbunătățire a modelului pentru a gestiona mai bine cazurile complexe.

7.3 Testarea detecției pieselor de șah

După antrenarea modelului de detectare a pieselor de șah cu imagini de piese și table virtuale și reale, folosind setul standard USCF[11], cel mai des utilizat la concursurile de șah (cu pătrate în culorile verde și alb) și piese de culori negru și alb de diverse dimensiuni, au fost generate anumite metrii pentru a evalua performanța modelului.

7.3.1 Matricea de confuzie

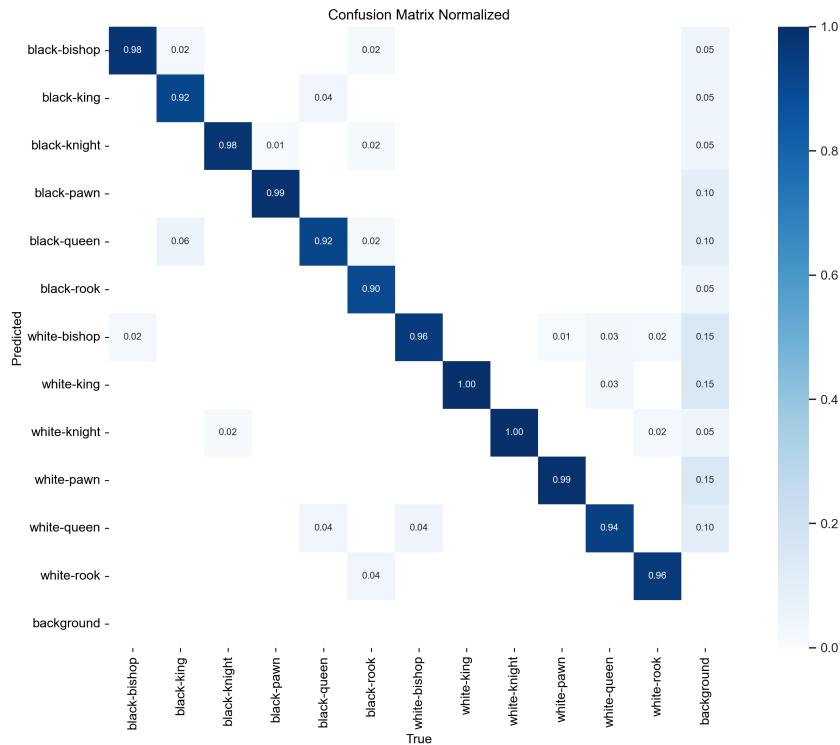


Figura 7.3: Matricea de confuzie

Matricea de confuzie este un instrument utilizat pentru a evalua performanța unui algoritm de clasificare. Aceasta arată numărul de predicții corecte și incorecte efectuate de model, împărțite pe clase. O matrice de confuzie normalizată a fost generată pentru modelul nostru, în care valorile sunt scalate astfel încât să fie între 0 și 1. Diagonala principală a matricei de confuzie prezintă valorile cele mai mari, reprezentate în albastru. Aceste valori mari pe diagonală indică faptul că modelul are o rată mare de clasificare corectă pentru fiecare clasă de piese de șah, sugerând o performanță bună a modelului.

7.3.2 Curba Precision-Recall

Curba Precision-Recall este un grafic utilizat pentru a evalua performanța unui sistem de clasificare binară. Precizia (precision) este definită ca proporția de predicții pozitive corecte față de toate predicțiile pozitive, în timp ce rata de reamintire (recall) este proporția de predicții pozitive corecte față de toate cazurile pozitive reale.

În graficul generat pentru fiecare tip de piesă și pentru toate piesele, se observă valori ridicate, cu o medie a preciziei (mAP@0.5) de 0.986 pentru toate clasele. mAP (mean Average Precision) este o metrică utilizată pentru a sumariza precizia unui model

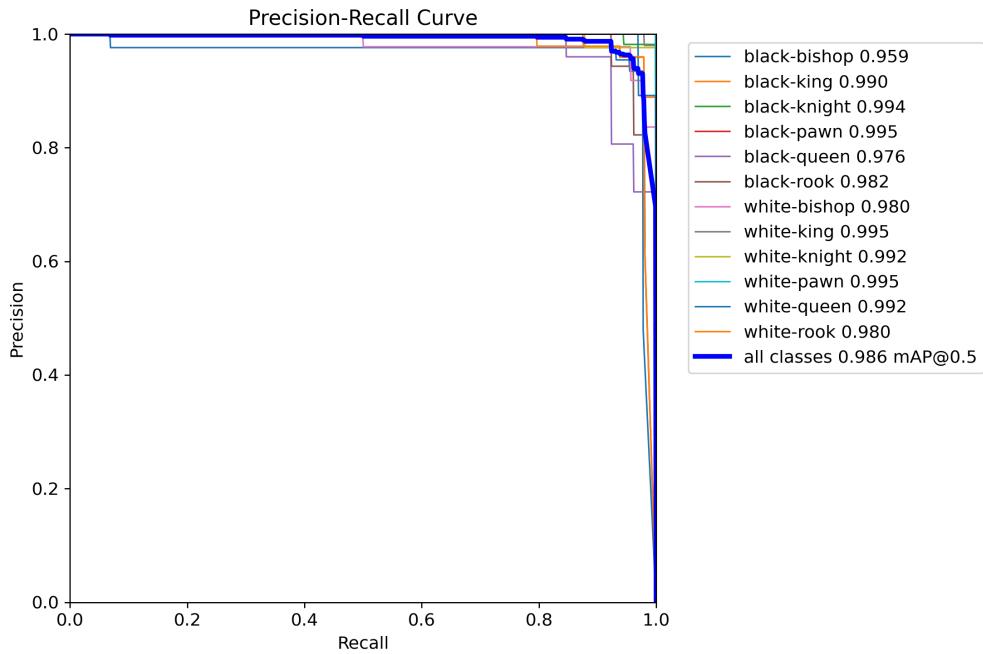


Figura 7.4: Curbă Precision-Recall

de detectare pe mai multe clase. Această valoare indică faptul că modelul funcționează foarte bine pe setul de date utilizat pentru testare, demonstrând o capacitate ridicată de a detecta corect piesele de șah.

7.3.3 Predicții vizuale ale pieselor



Figura 7.5: Predicții ale pieselor de șah pe imagini de testare

O imagine intitulată `val_batch0_pred.jpg` prezintă 16 imagini în care putem vedea predicțiile de piese efectuate de model pe setul de date de testare. Aceasta ilustrează modul în care modelul detectează și clasifică piesele de șah în imagini reale sau virtuale, confirmând vizual performanța ridicată a algoritmului.

Capitolul 8

Concluzii și rezultate

În această lucrare, am explorat utilizarea viziunii computerizate pentru recunoașterea și înregistrarea automată a stării jocului de șah, propunând proiectul Chess Snapshot. Am implementat și testat algoritmi avansați de detectare a tablei de șah și a pieselor de șah, abordând provocările legate de condițiile variate și dificile în care sunt capturate imaginile. Scopul nostru a fost să oferim o soluție practică și accesibilă pentru jucătorii de șah, eliminând nevoia de introducere manuală a mutărilor și permitând înregistrarea rapidă și precisă a jocului.

8.1 Rezultatele testării

Am evaluat performanța algoritmului nostru folosind un set de date de referință provocator, ce conține imagini ale tablelor de șah capturate în diverse condiții dificile. Rezultatele testării, prezentate în Tabelul 7.1, arată că algoritmul a obținut o rată ridicată de detecție corectă. Dintre cele 10 imagini testate, algoritmul a reușit să detecteze corect tabla de șah în 6 imagini, iar în 3 imagini a avut o detecție parțial corectă. Aceste rezultate subliniază capacitatea algoritmului nostru de a funcționa eficient în condiții dificile, inclusiv prezența altor obiecte, linii drepte generate de alte obiecte, table de șah deteriorate și condiții de iluminare variabile.

8.2 Concluzii

Implementarea Chess Snapshot demonstrează potențialul viziunii computerizate de a revoluționa modul în care jucătorii de șah își înregistrează și analizează partidele. Algoritmul implementat a arătat o performanță solidă în condiții variate, subliniind capacitatea sa de a oferi o soluție practică și accesibilă pentru detectarea automată a tablei și a pieselor de șah.

Capitolul 9

Dezvoltări viitoare

Pe măsură ce tehnologia și metodele de viziune computerizată continuă să evolueze, există numeroase direcții în care Chess Snapshot poate fi îmbunătățit și extins. Iată câteva dintre dezvoltările viitoare ce ar putea spori performanța și utilitatea acestui proiect:

9.1 Recunoașterea în timp real

O direcție importantă de dezvoltare este implementarea recunoașterii în timp real a poziției pieselor de șah. Aceasta ar permite detectarea continuă și actualizată a poziției pieselor pe tablă, oferind o experiență de utilizare mai dinamică și interactivă. Acest lucru ar putea fi deosebit de util în timpul transmisiunilor în direct ale turneelor de șah, permitând spectatorilor să vadă pozițiile pieselor actualizate în timp real.

9.2 Redimensionarea imaginilor de înaltă calitate

Un aspect crucial pentru îmbunătățirea detectării și recunoașterii este menținerea calității imaginii atunci când aceasta este redimensionată sau decupată. Dezvoltarea unor algoritmi avansați de redimensionare care păstrează detaliile esențiale ale imaginii va contribui la o detectare mai precisă a tablei și pieselor de șah, indiferent de transformările aplicate imaginii.

9.3 Antrenarea pe un set de date extins pentru detectorul de piese de șah

Un pas semnificativ pentru îmbunătățirea performanței algoritmului de recunoaștere a pieselor de șah este antrenarea pe un set de date extins și diversificat. Acest set de date ar trebui să includă imagini din diferite unghiuri, cu diverse condiții de iluminare și medii, pentru a asigura robustețe și adaptabilitate la situațiile din lumea reală. Utilizarea unor astfel de seturi de date ar crește acuratețea detecției și ar reduce erorile de recunoaștere.

9.4 Utilizarea rețelelor neuronale generative

Explorarea utilizării rețelelor neuronale generative, cum ar fi GAN-urile (Generative Adversarial Networks), pentru a crea imagini sintetice ale tablelor de șah ar putea oferi

seturi de date suplimentare pentru antrenarea algoritmilor. Aceste imagini sintetice pot contribui la diversificarea setului de date și la îmbunătățirea generalizării modelului.

9.5 Îmbunătățirea detectării colturilor și marginilor

Pentru a obține o detecție și mai precisă a tablei de șah, se pot dezvolta metode mai avansate de detectare a colturilor și marginilor. Utilizarea unor tehnici bazate pe învățare profundă pentru identificarea colturilor tablei, în locul metodelor tradiționale de detectare a marginilor, ar putea oferi rezultate mai bune în condiții dificile de iluminare și zgromot.

9.6 Extinderea funcționalităților de analiză a jocului

În prezent, aplicația mobilă dezvoltată în cadrul proiectului Chess Snapshot utilizează motorul de șah Stockfish pentru a afișa cea mai bună mutare disponibilă într-o poziție dată. Cu toate acestea, există numeroase oportunități de extindere a funcționalităților de analiză a jocului pentru a oferi o experiență și mai bogată utilizatorilor.

O direcție de dezvoltare viitoare ar fi integrarea completă a analizei jocului direct în aplicație. Aceasta ar include funcționalități avansate, cum ar fi:

- **Evaluarea detaliată a poziției.** Utilizatorii ar putea primi o evaluare detaliată a poziției actuale de pe tabla de șah, inclusiv avantaje/dezavantaje, puncte forte și puncte slabe ale poziției curente.
- **Recomandări de mutări strategice.** Pe lângă afișarea celei mai bune mutări, aplicația ar putea sugera mai multe mutări strategice, explicând avantajele și dezavantajele fiecărei opțiuni.
- **Analiza post-joc.** După finalizarea unui joc, aplicația ar putea oferi o analiză completă a partidei, identificând momentele cheie, greșelile critice și mutările excelente. Aceasta ar include și sugestii pentru îmbunătățirea jocului utilizatorului pe baza partidei jucate.
- **Salvarea și partajarea analizelor.** Utilizatorii ar putea salva analizele partidelor lor și le-ar putea partaja cu alții jucători sau antrenori pentru feedback și discuții suplimentare.

9.7 Jocul în rețea

O altă dezvoltare importantă pentru Chess Snapshot ar fi implementarea funcționalității de joc în rețea. Aceasta ar permite utilizatorilor să joace șah împotriva altor persoane în timp real, extinzând astfel interactivitatea și comunitatea din jurul aplicației. Funcționalitățile specifice ar putea include:

- **Meciuri online în timp real.** Utilizatorii ar putea invita prietenii să joace partide de șah online sau ar putea fi potriviti automat cu alții jucători disponibili.
- **Clasamente și statistici.** Aplicația ar putea include un sistem de clasamente și statistici, permitând utilizatorilor să urmărească performanțele lor și să compare rezultatele cu alții jucători.

- **Turnee online:** Organizarea de turnee online în cadrul aplicației ar putea atrage și mai mulți utilizatori, oferindu-le oportunitatea de a concura pentru premii și recunoaștere.
- **Chat și comunitate.** Adăugarea unei funcționalități de chat ar permite jucătorilor să comunice între ei în timpul jocurilor, facilitând discuțiile și interacțiunile sociale.

Aceste direcții de dezvoltare viitoare subliniază potențialul vast al proiectului Chess Snapshot și oportunitățile de a continua îmbunătățirea și extinderea funcționalităților sale pentru a servi mai bine comunitatea șahistă.

Capitolul 10

Bibliografie

- [1] Craig Belshe. Chess piece detection. 12 2021.
- [2] Paul Bourke. Intersection point of two line segments in 2 dimensions. 1989.
- [3] Chess.com. Forsyth-edwards notation (fen). <https://www.chess.com/terms/fen-chess>.
- [4] Maciej Czyzowski, Artur Laskowski, and Szymon Wasik. Chessboard and chess piece recognition with the support of neural networks. *Foundations of Computing and Decision Sciences*, 45:257–280, 12 2020.
- [5] Jialin Ding. Chessvision : Chess board and piece recognition. 2016.
- [6] Tomasz Kacmajor. Hough lines transform explained. <https://medium.com/@tomasz.kacmajor/hough-lines-transform-explained-645feda072ab>.
- [7] Szymon Wasik Maciej A. Czyzowski, Artur Laskowski. Latchess21: dataset of damaged chessboard lattice points (chessboard features) used to train laps detector (grayscale/21x21px), 2018. <https://repod.icm.edu.pl/dataset.xhtml?persistentId=doi:10.18150/repod.7606646>.
- [8] Stack Overflow. Hough bundler. <https://stackoverflow.com/a/70318827/12505886>.
- [9] passion-programmation laurentdu64.blogspot.com. Simple chess board flutter widget. https://pub.dev/packages/simple_chess_board.
- [10] Roboflow. Chess pieces dataset. <https://public.roboflow.com/object-detection/chess-full>.
- [11] Colin Staczynski. Chess board dimensions | basics and guidelines. <https://www.chess.com/article/view/chess-board-dimensions>.
- [12] Ultralytics. Ultalytics github repository for yolov8. <https://github.com/ultralytics/ultralytics>.

Capitolul 11

Anexă

11.1 Anexă A: Imagini de test pentru detectarea tablei de șah



Figura 11.1: p1.jpg



Figura 11.2: p2.jpg



Figura 11.3: p3.jpg



Figura 11.4: p4.jpg



Figura 11.5: p5.jpg



Figura 11.6: p6.jpg



Figura 11.7: p7.jpg

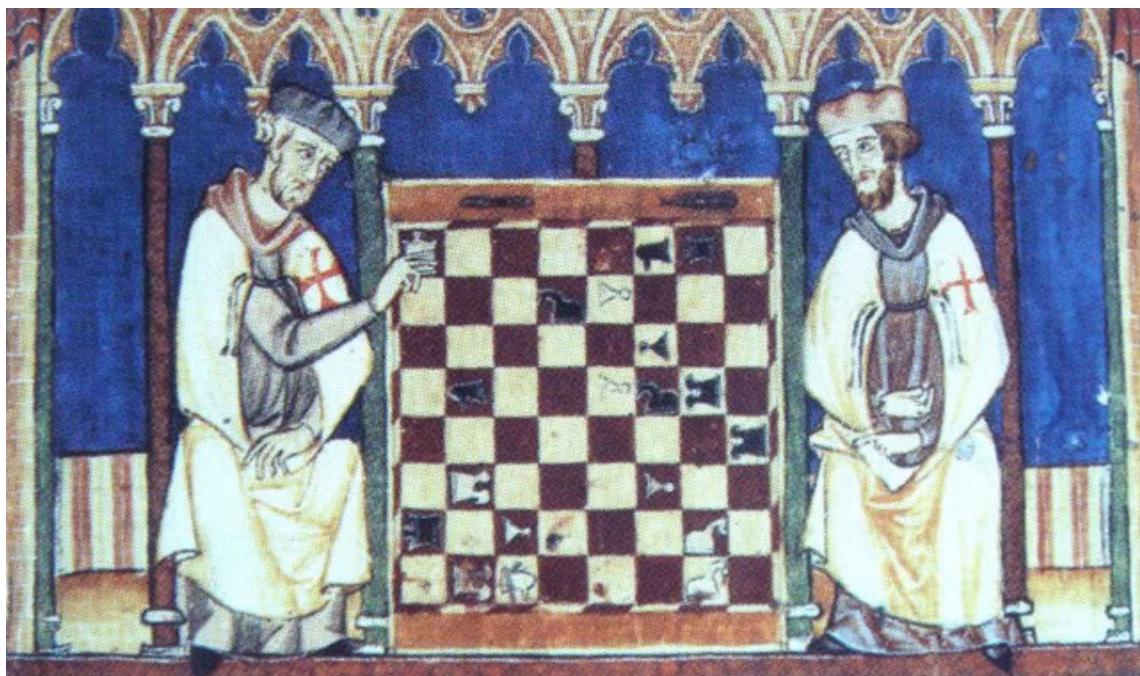


Figura 11.8: p8.jpg



Figura 11.9: p9.jpg



Figura 11.10: p10.jpg

11.2 Anexă B: Imagini de test pentru detectarea pieselor



Figura 11.11: p1.jpg



Figura 11.12: p2.jpg



Figura 11.13: p3.jpg



Figura 11.14: p4.jpg



Figura 11.15: p5.jpg