BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Recunoaşterea configuraţiilor a tablelor de şah folosind algoritmi de inteligenţă artificală

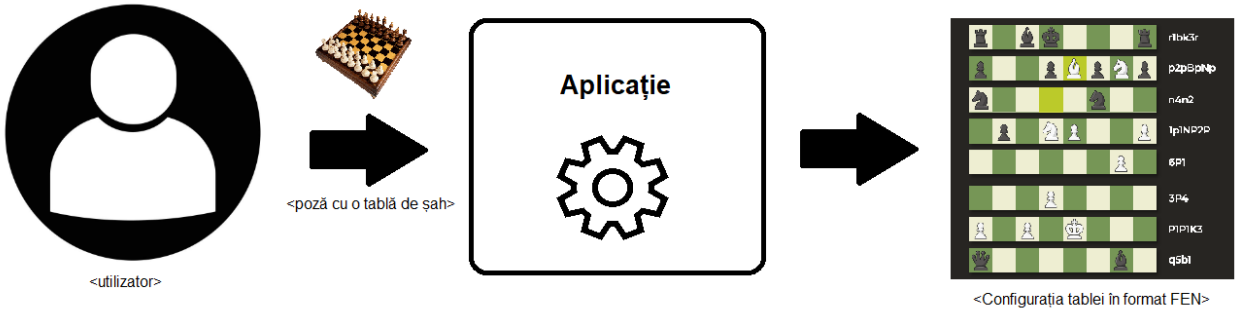
– MIRPR report –

Membrii echipă:
Puşnei Victor-Dumitru, Informatică Română, grupa 236,
victor.pusnei@stud.ubbcluj.ro

2023-2024

Rezumat

- Scopul acestui proiect este recunoaşterea configuraţia pieselor dintr-o poză cu o tablă de şah şi reprezentarea lor într-un format digital. Folosind algoritmi de inteligenţă artificală, acest proces este mult mai rapid decât dacă ar fi fost realizat manual de o persoană.
- Pentru a ajunge la rezultatul final, mai întai programul trebuie să detecteze tabla de şah din poză, apoi să detecteze fiecare piesă de pe tablă si tipul acesteia, pentru aceste două etape fiind folosiţi algoritmi de object-detection.
- Pentru antrenare şi testare a programului, au fost folosite seturi de date de pe platforma Roboflow ce au fost organizate în diferite moduri pentru a se potrivi cerinţelor acestui proiect.
- Acest proiect are potenţialul de a oferi predicţii destul de precise, dar acest lucru depinde de dimensiunea şi complexitatea setului de date, deoarece în realitate un tip de piesă poate arăta in diferite moduri, lucru ce poate pune dificultaţi în clasificarea pieselor prezente in poză.



Cuprins

Capitolul 1

Introducere

1.1 Prezentarea problemei şi importanţa acesteia

- Problema poate fi prezentată în următorul mod: primind o poză ce conţine o tablă de şah, trebuie să recunoaştem anumite caracteristici ale acesteia.
- Importanţa rezolvării problemei prezentate nu este neapărat dată de acest caz particular, ci de faptul că putem extrage din soluţia acestei probleme nişte paşi generali ce pot fi folosiţi în abordarea subiectelor cu o temă asemănatoare.
- Pentru realizarea acestui proiect, au fost încercate diferite abordări, iar în final metoda ce a oferit cele mai bune rezultate a fost de a simplifica problema puţin câte puţin (ex.: reducerea a întregii pozei primite la doar tabla de şah in sine, înainte de a trece la etapa în care se încearca detectarea şi clasificarea pieselor, etc.)

1.2 Structura articolului şi obiectivele acesteia

Obiectivul principal a acestei prezentări este de a oferi o metodă de rezolvare a problemei descrise folosind algoritmi de inteligenţă artificială.

Al doilea obiectiv este de a crea o aplicaţie software uşor de folosit pentru utilizatori, ce oferă rezultatul întregului proces de rezolvare a acestei probleme.

Acest articol conţine 3 referinte bibliografice, şi este structurat în urmatorul fel:

Capitolul 1 conţine o scurtă introducere.

Capitolul 2 descrie problema abordată din acest articol.

În capitolul 3 se vorbeşte despre probleme asemenătoare abordate în alte articole.

În capitolul 4 se descrie abordarea folosită in rezolvarea acestei probleme.

Capitolul 5 conţine diferite informaţii despre implementarea aplicaţiei, a setului de date folosit, etc.

În capitolul 6 sunt ilustrate concluziile.

Capitolul 2

Problema ştiinţifică

2.1 Definiţia problemei

O prezentare generală a problemei:

Primind ca date de intrare o poză ce conţine un anumit obiect, trebuie să extragem anumite informaţii ale acelui obiect şi să le prezentăm ca date de ieşire.

Această problemă poate fi întalnita în mai multe situaţii, de exemplu detectarea plăcuţelor de înmatriculare ale maşinilor (maşina → plăcuţa → obiect, numărul → informaţie), detectarea unui semafor (obiect) şi culoarea acestuia (informaţie).

În cazul de faţă, obiectul este o tablă de şah, iar informatiile ce trebuie extrase sunt poziţia şi tipul pieselor de pe aceasta.

Folosirea algoritmilor de inteligenţa artificală este crucială în acestă situaţie deoarece este cea mai convenabilă metodă de a detecta tabla de şah din imagine şi de a clasifica piesele de pe aceasta.

Capitolul 3

Related work

Metoda iniţială a rezolvării acestei probleme a fost inspirată dintr-un articol [1] în care se încearcă rezolvarea aceleiaşi probleme. Există o diferenţă faţă de acea metodă şi cea prezentată în acest articol:

- În cealaltă abordare se folosesc tehnici de computer-vision pentru a detecta liniile tablei de şah. Prin intersecţia acelor linii, sunt calculate coordonatele pătratelor.
- În abordarea prezentată aici, tabla de şah este detectată folosind colţurile acesteia (algoritmi de object-detection, antrenaţi pe seturi de date, ce detectează colţurile tablei), după detectarea colţurilor se folosesc tehnici de computer vision pentru a schimba perspectiva pozei pentru a obţine o imagine doar cu tabla de şah.
- Abordarea a doua va obţine rezultate mai bune deoarece, prima abordare va trebui să se bazeze pe anumiţi parametri ce nu pot fi adaptaţi mereu la orice tip de imagine (cu lumină, claritate, etc. diferite) în detectarea liniilor, de exemplu uneori vor fi detectate linii ce se află înafara tablei de şah, sau unele linii care trebuie detectate nu vor fi detectate, lucru ce poate împiedica determinarea pătratelor.

Decizia de a folosi a doua abordare a fost luată după consultarea unui alt articol [2].

De asemenea, a fost de ajutor şi un al treilea articol [3] ce abordează o problemă asemanătoare, dar mai complexă: detectarea unor ecuaţii dintr-o imagine şi rezolvarea lor.

Capitolul 4

Abordare folosită

Va fi prezentat în pseudocod procesul prin care rezultatul este obţinut:

Algorithm 1 Algoritmul de extragere a configuraţiei în format FEN din imagine

```
BEGIN
@ Recunoaşte colţurile din imagine folosind primul model sau primeşte de la utilizator dacă acesta eşuează
coordColturi ← ExtrageCoordonateColturi(imagine);
@ Schimbă perspectiva imaginii, astfel încat să conţină doar tabla de şah vazută de sus
imagineTransformata ← PerspectiveTransform(imagine, coordColturi);
@ Folosind modelul de object-detection, se detectează coordonatele pieselor şi clasele acestora (fiecare piesa va fi inclusă într-un dreptunghi)
dreptunghiuriSiClasePiese ← RecunoasteCoordonateSiClasePiese(imagineTransformata);
@ Se construieşte configuraţia tablei, sub forma de lista / matrice 8x8, folosindu-ne de dreptunghiurile detectate (ce conţin piese)
configuratie ← ConstruireConfiguratie(dreptunghiuriSiClasePiese);
@ Convertire in format FEN a configuratiei
configuratieFen ← ConversieSpreFEN(configuratie);
TrimiteSpreClient(configuratieFen);
END
```

Iniţial, se încearcă detectarea colţurilor folosind primul model de object-detection, iar dacă acesta eşuează, utilizatorul are posibilitatea de a indica poziţia colţurilor din imagine.

Dupa acest stadiu, avanând colţurile, putem transforma perspectiva imaginii astfel încat aceasta sa se vadă „de sus”, în care este inclusă doar tabla. Acest lucru este important deoarece putem împarţi apoi imaginea în 8x8, lucru ce ne va rezulta poziţia fiecărui pătrat.

Apoi, folosind al doilea model de object-detection vor fi detectate piesele de pe tablă împreuna cu clasele lor, aşadar vom şti pozitia acestora.

În final, având tipul şi poziţia fiecărei piese pe tablă, putem genera configuraţia tablei în format FEN.

Acest pseudocod este o versiune simplificată a aplicaţiei, şi nu oferă detalii specifice despre modul în care sunt calculate coordonatel sau alte lucruri ce nu au legătură cu modelele de inteligenţă artificală.

Capitolul 5

Aplicaţie (Studiu de caz)

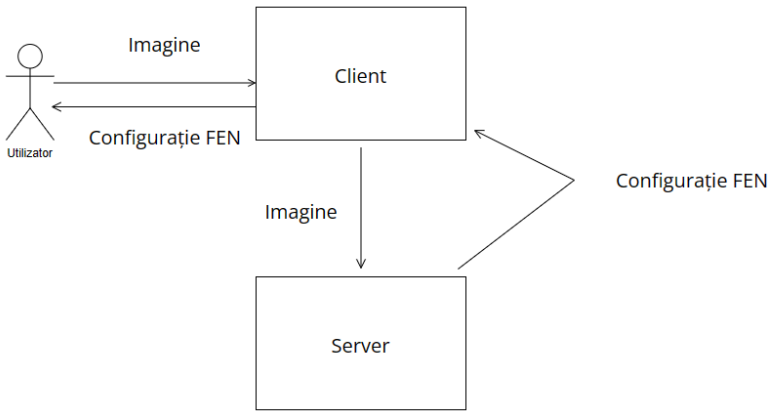
5.1 Descrierea aplicaţiei

Aplicaţia conţine două părţi: client-ul şi server-ul.

Pe partea de server, doar se calculează şi se trimite înapoi la client rezultatul problemei (configuraţia FEN) dupa ce input-ul (imaginea şi coordonatele dacă este cazul) este primit.

Pe partea de client, utilizatorul poate încarcă imaginea ce conţine tabla de şah, şi poate specifica colţurile tablei (dacă server-ul va avea nevoie). Client-ul de asemenea poate specifica din perspectiva cărui jucător a făcut poza.

O simplă diagramă ce prezintă procesul la modul general:

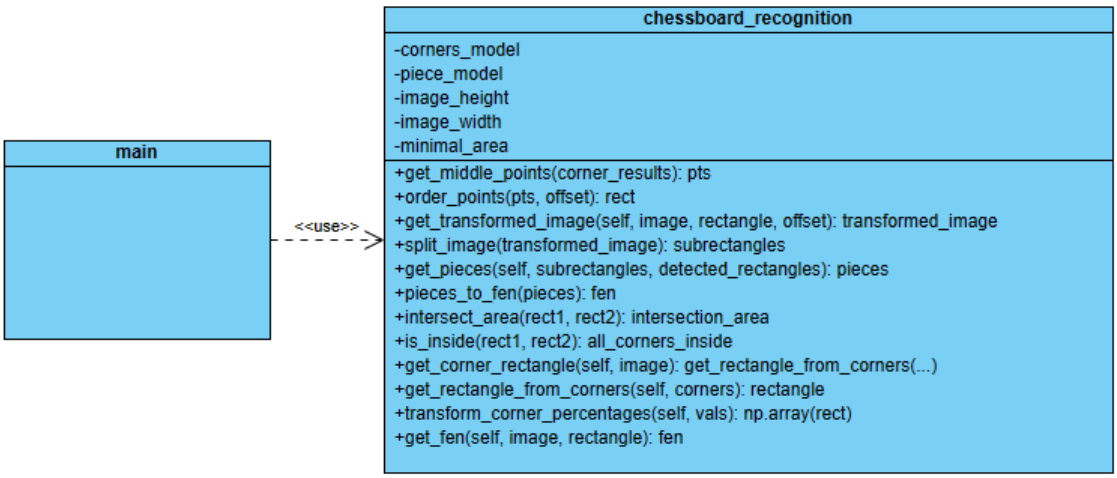


5.2 Design-ul aplicaţiei

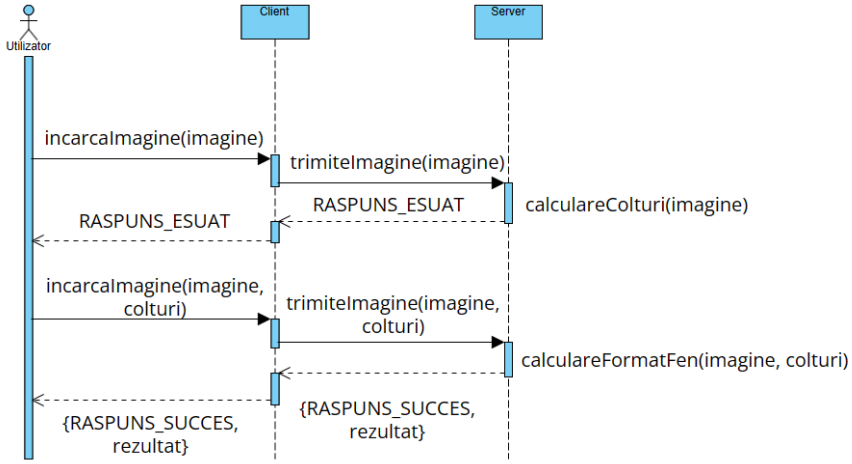
Cele trei scenarii de utilizare ale aplicaţiei:

- Caz favorabil: pe partea de server colţurile sunt identificate cu succes şi se merge mai departe (pana la trimiterea rezultatului spre utilizator).
- Caz neutru: pe partea de server colţurile nu sunt identificate cu succes, iar utilizatorul va trebui sa le indice pe imagine, apoi se merge mai departe (pana la trimiterea rezultatului spre utilizator).
- Caz nefavorabil: pe partea de server colţurile sunt identificate ca fiind 4, dar nu în poziţiile potrivite. Aşadar transformarea perspectivei a imaginii va fi incorectă, şi va duce la rezultate neprevizibile.

Structura claselor nu este foarte complexă pe partea de server, programul principal (care comunică cu client-ul) se foloseşte de o clasă suplimentară ce conţine partea de calcul / recunoaştere:



Diagramă de secvenţe (prezintă în cazul neutru doar interacţiunile dintre utilizator, client şi server, nu şi logica determinării rezultatului):



5.3 Implementare

Pe partea de client s-a folosit React pentru a putea realiza o aplicaţie web cu care utilizatorul poate interacţiona, iar pe partea de server s-a folosit Python, împreună cu Flask pentru a putea comunica cu client-ul.

5.3.1 Seturi de antrenare şi statistici

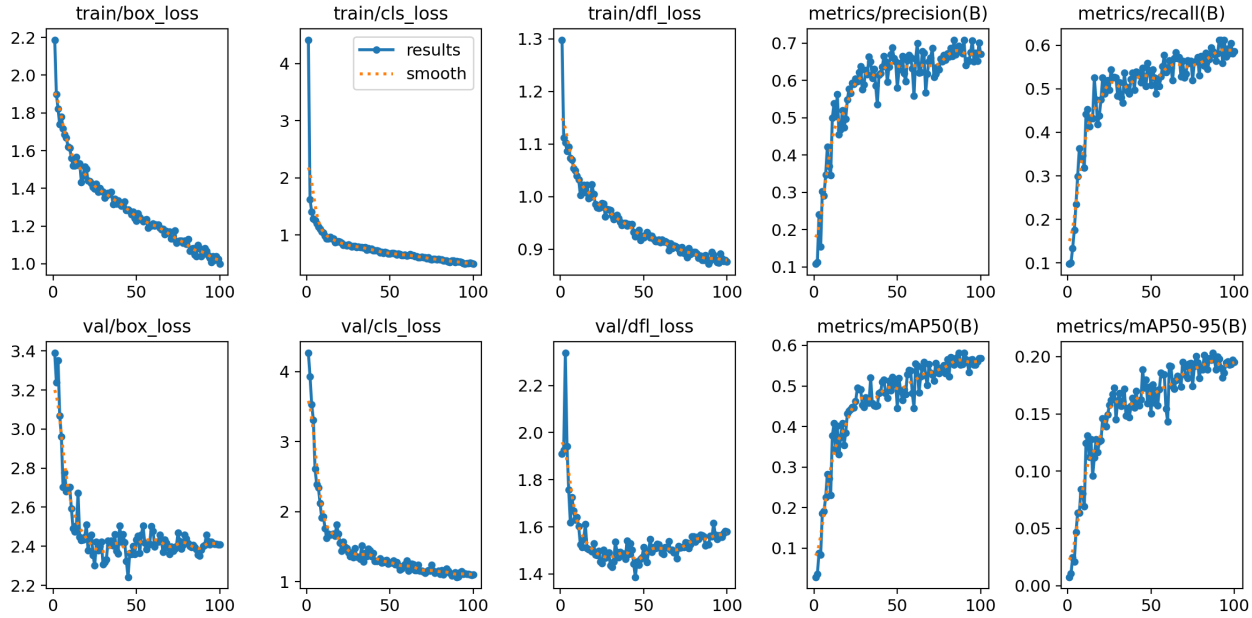
Aşa cum a fost specificat în capitolul 4, aplicaţia foloseşte două modele de intelienţa artificală:

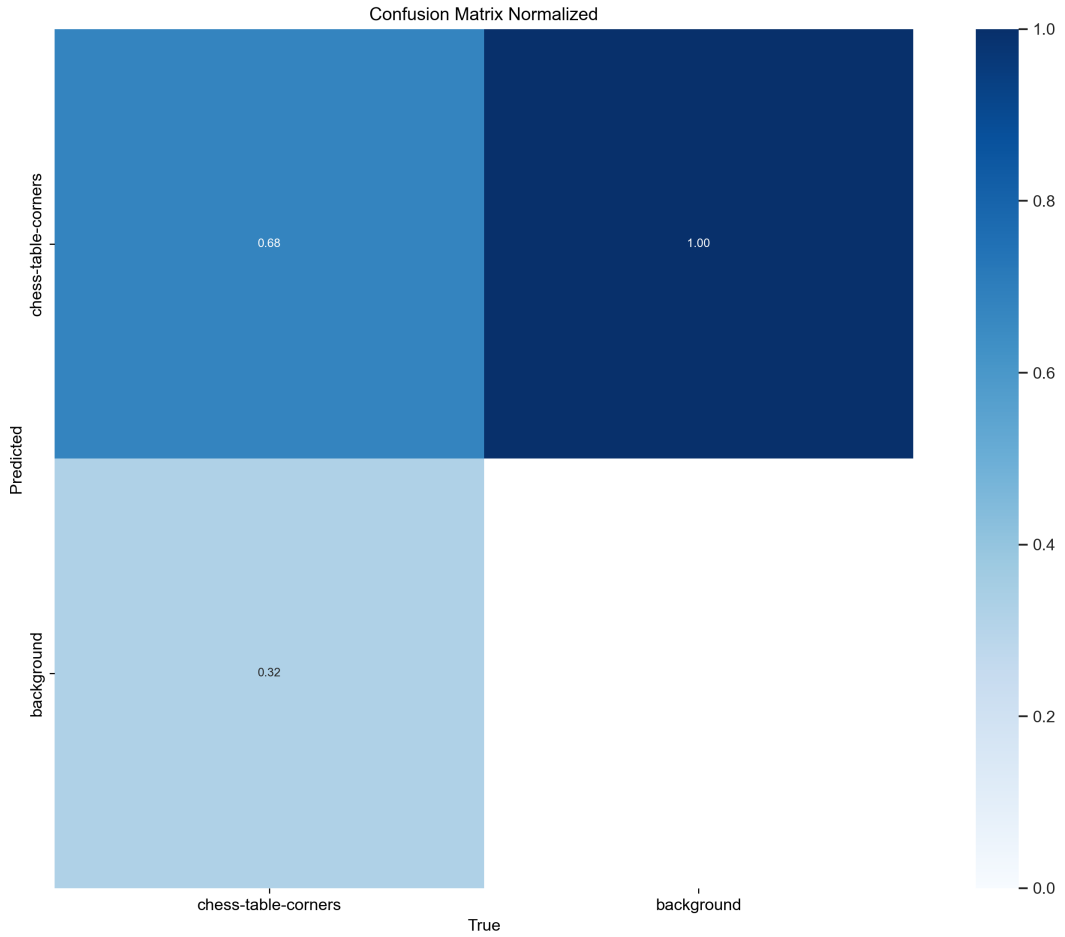
5.3.1.1 Modelul pentru detectarea colţurilor

Acest model a fost antrenat deasupra modelului YOLOv5, folosind: 603 imagini pentru antrenare, 103 imagini pentru validare si 62 pentru testare.

Majoritatea imaginilor au fost colectate de pe internet si adnotate manual pe platforma Roboflow, iar restul au fost extrase din alte seturi de date publicate pe aceasta platformă.

Folosind platforma Roboflow, numărul de imagini a putut fi mărit prin augmentarea datelor, aşadar s-a ajuns la numerele specificate mai sus.

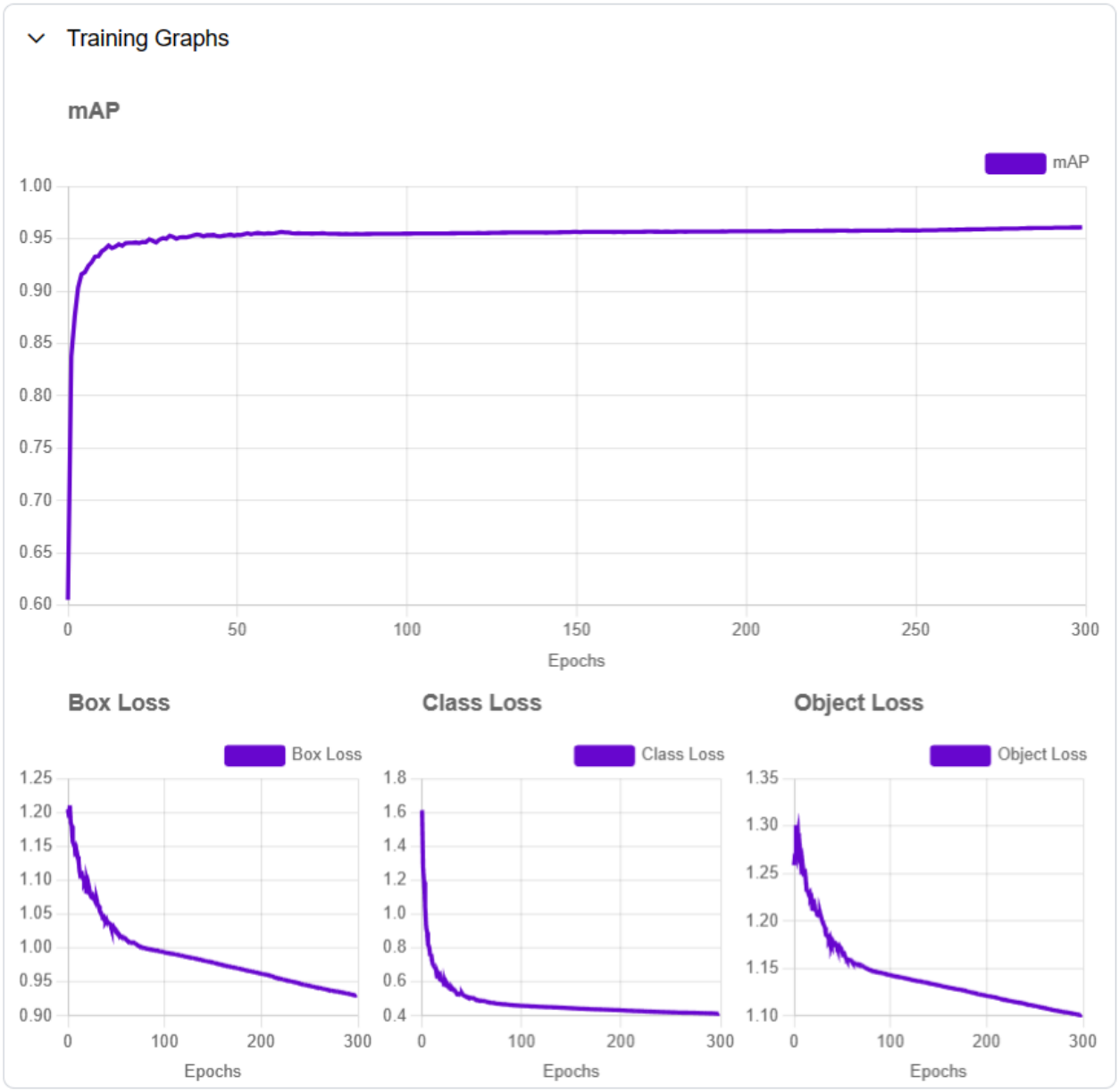

train/box_loss
train/cls_loss
results
smooth
train/dfl_loss
metrics/precision(B)
metrics/recall(B)
val/box_loss
val/cls_loss
val/dfl_loss
metrics/mAP50(B)
metrics/mAP50-95(B)


Confusion Matrix Normalized
chess-table-corners
background
Predicted
0.68
1.00
0.32
chess-table-corners
background
True

5.3.1.2 Modelul pentru detectarea poziţiilor şi clasele pieselor

Acest model a fost antrenat folosind platforma Roboflow, folosind: 12495 imagini pentru antrenare, 1706 imagini pentru validare si 1421 pentru testare. Ca şi mai sus, dimensiunea setului de date a fost marită prin augmentare.

Toate imaginile din acest set de date au fost colectate din multiple seturi de date publicate pe platforma Roboflow.

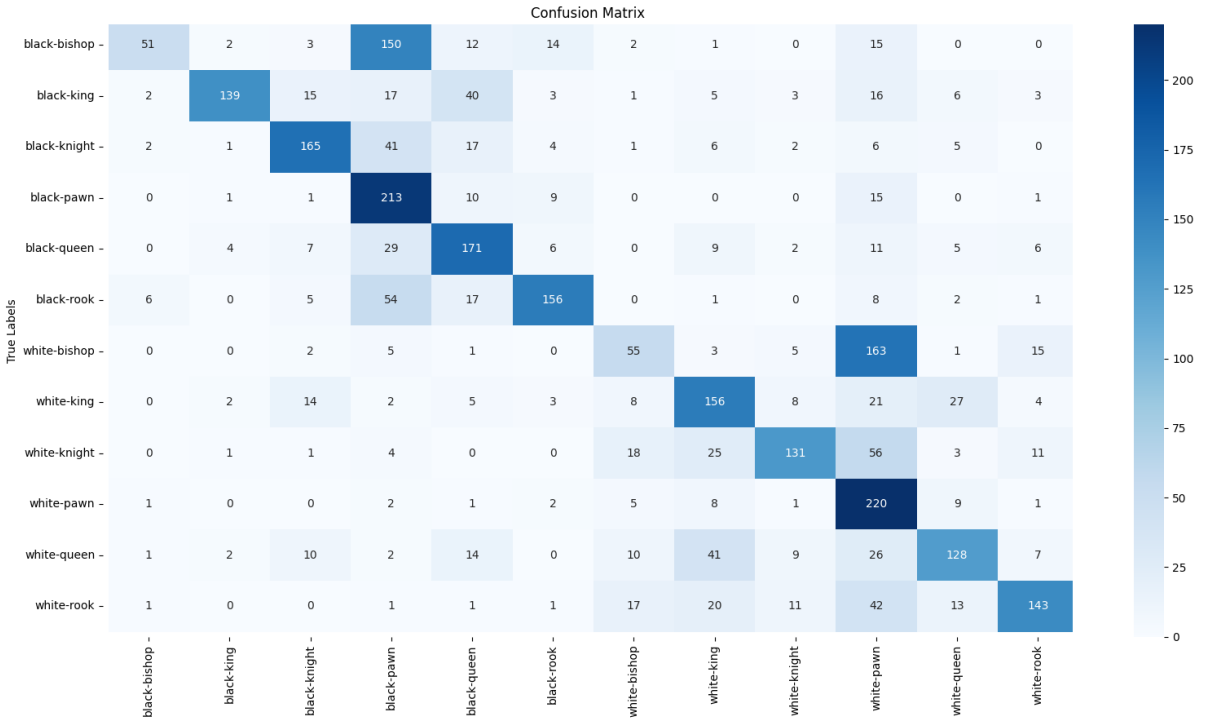


5.3.1.3 Modelul extra pentru clasificarea pieselor (experiment)

Iniţial, în rezolvarea acestei probleme s-a încercat folosirea unui model suplimentar ce ar fi avut rolul doar de a clasifica o piesă individuală dintr-o imagine (adica doar dintr-un pătrat), dar deoarece rezultatele oferite de acest model (dimensiunea setului de date a fost foarte mică comparativ cu modelul precedent) au fost extrem de slabe în practică, s-a decis eliminarea sa din aplicaţie.

Modelul folosit pentru clasificarea pieselor pleacă de la modelul preantrenat VGG16, antrenat suplimentar, folosind: 1000 imagini pentru fiecare 12 clase (culoare-piesă) pentru antrenare, 250 imagini pentru fiecare 12 clase pentru testare.

Un set de date mai mari ar fi fost foarte greu de gestionat în faza de antrenare, neavând posibilitatea de a folosi o platformă precum Roboflow.



	precision	recall	f1-score	support
black-bishop	0.80	0.20	0.32	250
black-king	0.91	0.56	0.69	250
black-knight	0.74	0.66	0.70	250
black-pawn	0.41	0.85	0.55	250
black-queen	0.59	0.68	0.63	250
black-rook	0.79	0.62	0.70	250
white-bishop	0.47	0.22	0.30	250
white-king	0.57	0.62	0.59	250
white-knight	0.76	0.52	0.62	250
white-pawn	0.37	0.88	0.52	250
white-queen	0.64	0.51	0.57	250
white-rook	0.74	0.57	0.65	250
accuracy			0.58	3000
macro avg	0.65	0.58	0.57	3000
weighted avg	0.65	0.58	0.57	3000

5.3.2 Discuţii

Din păcate, platforma Roboflow nu permite vizualizarea mai multor statistici, aşadar e greu de făcut o comparare mai detaliată dintre modelele 2 şi 3.

Din ilustratiile de mai sus, putem observa că rezultatele modelelor nu sunt foarte precise, aşadar există îmbunătaţiri ce se pot face.

Pentru a ne da seama de diferenţa dintre cele două seturi de date, modelul 2 conţine aproximativ 250000 de piese individuale doar în setul de antrenare, iar modelul experiment 12000.

Modelul VGG16 are potenţialul de a oferi rezultate mai precise deoarece excelează în clasificarea obiectelor.

De asemenea, trebuie notat faptul că fiecare set de piese arată diferit, aşadar chiar şi un model foarte bine antrenat poate oferi rezultate incorecte atunci când întâlneşte un set nou de piese ce arată diferit faţă de cele pe care a fost antrenat.

Capitolul 6

Concluzii

În concluzie, putem observa slabiciunea principală al acestui întreg experiment: seturile de date, acestea putând fi extinse. Detectarea colţurilor unei table de şah şi clasificarea pieselor este o sarcină destul de complexă, întrucât acestea pot arăta foarte diferit de la caz la caz, aşadar în viitor repetarea acestui experiment cu noi seturi de date are posibilitatea de a oferi rezultate mai bune.

Deşi în acest proiect seturile de date nu au fost tocmai ideale, un avantaj al abordării folosite a fost împarţirea sarcinii iniţiale în sub-probleme, adică performanţa celor 2 modele, aşadar fiecare dintre acestea poate fi îmbunătăţită / modificată în mod independent.

Pentru viitor, acest proiect poate fi îmbunătăţit prin extinderea setului de date, dar şi prin împărţirea sarcinilor pentru modelele de inteligenţă artificială. De exemplu, experimentul încercat în acest articol poate fi repetat, dar cu un set de date mult mai mare pentru modelul preantrenat VGG16, ce ar trebui să ofere rezultate mai bune deoarece acesta este mai precis în clasificarea de obiecte dintr-o poză, pe când modelul YOLO este mai precis în detectarea obiectelor. Asadar, modelul 2 poate fi folosit doar pentru detectarea pieselor, iar 3 doar pentru clasificarea fiecărei piese.

Bibliografie

[1] A. Underwood - "Board Game Image Recognition using Neural Networks", Towards Data Science, Oct. 21, 2020, https://towardsdatascience.com/board-game-image-recognition-using-neural-networks-116fc876dafa

[2] J. Gallagher - "Represent Chess Boards Digitally with Computer Vision", Roboflow, Mar. 10 2023, https://blog.roboflow.com/chess-boards/

[3] Huu, Phat Nguyen, and Khang Doan Xuan - "Proposing Algorithm Using YOLOV4 and VGG-16 for Smart-Education.", Applied Computational Intelligence and Soft Computing 2021 (2021): 1-14.