UNIVERZITA SV. CYRILA A METODA V TRNAVE
FAKULTA PRÍRODNÝCH VIED

ROZPOZNÁVANIE ROZLOŽENIA FIGÚROK NA ŠACHOVNICI UMELOU INTELIGENCIOU

Bakalárska práca

2024 **Serhii Vielkin**

UNIVERZITA SV. CYRILA A METODA V TRNAVE
FAKULTA PRÍRODNÝCH VIED

ROZPOZNÁVANIE ROZLOŽENIA FIGÚROK NA ŠACHOVNICI UMELOU INTELIGENCIOU

Bakalárska práca

Študijný program: Aplikovaná informatika
Študijný odbor: 18. Informatika
Školiace pracovisko: Ústav počítačových technológií a informatiky
Školiteľ: prof. RNDr. Jiří Pospíchal. DrSc.

2024 **Serhii Vielkin**

Čestné vyhlásenie

Prehlasujem, že som svoju bakalársku prácu na tému: Rozpoznávanie rozloženia figúrok na šachovnici umelou inteligenciou vypracoval samostatne pod odborným vedením svojho školiteľa a s použitím literatúry uvedenej v zozname.

Trnava, 2024 **.....**

Podpis

Poďakovanie

Ďakujem Mgr. Martinovi Kubovčíkovi za jeho rady a pripomienky pri vypracovaní bakalárskej práce.

ABSTRAKT (V SLOVENSKOM JAZYKU)

VIELKIN, Serhii: Rozpoznávanie rozloženia figúrok na šachovnici umelou inteligenciou (Bakalárska Práca) - Univerzita sv. Cyrila a Metoda, Fakulta prírodných vied, Ústav počítačových technológií a informatiky. Vedúci práce: prof. RNDr. Jiří Pospíchal. DrSc – UCM v Trnave, 2024. – 55 strán.

Cieľom tejto bakalárskej práce je upraviť a analyzovať algoritmy strojového učenia na rozpoznávanie šachových figúr na obrázkoch. V rámci výskumu sme použili metódy hlbokého učenia s využitím predtrénovaných neurónových sietí VGG16 a ResNet101, ktoré sú prispôsobené na úlohu klasifikácie. Osobitnú pozornosť sme venovali predspracovaniu obrazu vrátane rozšírenia a normalizácie s cieľom zlepšiť výkonnosť rozpoznávania. Experimentálne výsledky ukázali vysokú presnosť oboch modelov, čo potvrdzuje účinnosť zvolených metód a architektúr. Práca prispieva k vývoju algoritmov počítačového videnia na analýzu šachových partií a môže byť použitá na vytvorenie automatizovaných tréningových a analytických systémov v oblasti šachu.

Kľúčové slová: strojové učenie, hlboké učenie, rozpoznávanie vzorov, šachové figúrky, predspracovanie obrazu, rozšírenie dát, VGG16, ResNet101.

ABSTRAKT (V CUDZOM JAZYKU)

VIELKIN, Serhii: Artificial intelligence recognition of the layout of pieces on a chessboard (Bachelors Thesis) - University of St. Cyril and Methodius, Faculty of Natural Sciences, Department of Computer Technologies and Informatics. Thesis supervisor: prof. RNDr. Jiří Pospíchal. DrSc - UCM in Trnava, 2024. - 55 pages.

The aim of this thesis is to develop, adapt and analyze machine learning algorithms for the recognition of chess pieces in images. The study employed deep learning methods using pretrained neural networks VGG16 and ResNet101, adapted for the task of classification. Special attention was given to image preprocessing, including augmentation and normalization, which enhanced recognition accuracy. The experimental results demonstrated high accuracy for both models, confirming the effectiveness of the selected methods and architectures. This work contributes to the development of computer vision algorithms for analyzing chess games and can be utilized to create automated training and analytical systems in chess.

Kľúčové slová: machine learning, deep learning, image recognition, chess pieces, image preprocessing, data augmentation, VGG16, ResNet101

OBSAH

ZOZNAM ILUSTRÁCIÍ A ZOZNAM TABULIEK

Tabuľky

Obrázky

ZOZNAM SKRATIEK A ZNAČIEK

AI - umelá inteligencia

ML - strojové učenie

DL - hlboké učenie

CNN - konvolučná neurónová sieť

ANN - umelá neurónová sieť

SGD - stochastické gradientné klesanie

ROC - graf prahovej operačnej charakteristiky

AUC - plocha pod krivkou ROC

TPR - True Positive Rate (skutočná pozitívna miera)

FPR – False Positive Rate (počet falošných poplachov)

TP - True Positives (skutočne pozitívne výsledky)

FP – False Positive (falošne pozitívne výsledky)

TN – True Negative (pravdivé negatívne)

FN – False Negative (falošne negatívne)

F1-skóre - F1-meranie, harmonický priemer Precision a Recall

VGG16 - model neurónovej siete vyvinutý v skupine Visual Geometry Group v Oxforde

ResNet101 - hlboká neurónová sieť so 101 vrstvami vyvinutá spoločnosťou Microsoft Research

Adam - optimalizátor používaný pri trénovaní neurónových sietí

Fine-tuning - jemné ladenie, metóda predtrénovania neurónovej siete na nových údajoch

ÚVOD

V dnešnom svete technológií sú témy ako strojové učenie, umelá inteligencia a neurónové siete čoraz aktuálnejšie. Táto práca sa zaoberá základnými pojmami umelej inteligencie, rôznymi modelmi strojového učenia, ich použitím na riešenie konkrétnych problémov a metódami výberu vhodného typu modelu pre konkrétny problém. Osobitná pozornosť je venovaná praktickému využitiu získaných poznatkov.

Jedným z nových smerov skúmaných v tejto práci je problém multiklasifikácie obrazov pomocou neurónových sietí. Cieľom výskumu bolo vytvoriť algoritmus, ktorý by nielen efektívne rozpoznával polohu šachových figúrok na šachovnici, ale aj prekonával existujúce riešenia v tejto oblasti.

Táto práca sa zameriava na aplikáciu pokročilých techník strojového učenia a umelej inteligencie na problém rozpoznávania a analýzy polohy šachových figúrok na obrázkoch. Práca predstavuje významný príspevok do oblasti počítačového videnia a strojového učenia tým, že kombinuje najmodernejšie techniky hlbokého učenia, ako je jemné ladenie a transferové učenie, s inovatívnymi prístupmi k predspracovaniu a rozšíreniu obrazu.

Zaujímavosť práce spočíva v jej schopnosti riešiť a vyriešiť špecifický problém rozpoznávania šachových figúrok, ktorý je dôležitý nielen v oblasti športu, ale aj v širších aplikáciách, ako je vzdelávanie, strategické plánovanie a umelá inteligencia. Práca ukazuje, ako môžu najmodernejšie techniky spracovania obrazu a strojového učenia zvýšiť presnosť a efektívnosť automatického spracovania a interpretácie komplexných vizuálnych údajov. Tieto technológie majú potenciál výrazne zlepšiť tréningové nástroje a analytické systémy používané šachistami a trénermi a vyvinúť nové aplikácie v oblastiach vyžadujúcich rýchle a presné vizuálne rozpoznávanie.

1 ÚVOD DO STROJOVÉHO UČENIA

1.1 Python

Python je vysokoúrovňový programovací jazyk s prísnym dynamickým typovaním, ktorý sa široko používa v rôznych oblastiach vrátane vývoja webových stránok, vedeckých výpočtov, analýzy údajov a strojového učenia. Tento jazyk, ľahko sa používa a je veľmi lojálny voči používateľom.

1.2 Python ako jazyk pre strojové učenie

Python nie je jediný jazyk používaný na spracovanie údajov, existujú aj iné, ako napríklad:

- Java

Java sa dá použiť na strojové učenie, existujú na to niektoré knižnice, napríklad Apache OpenNLP a Deeplearning4j.

Nevýhody: Na vyriešenie rovnakého problému môže vyžadovať oveľa viac kódu ako Python.

- C++

Jazyk C++ poskytuje vysoký výkon a môže byť užitočný na vývoj v oblasti strojového učenia.

Nevýhody: Hoci je Python pomalší jazyk, faktom je, že väčšina výpočtových operácií v neurónových sieťach pomocou jazyka Python sa vykonáva pomocou knižníc (NumPy, Pandas, OpenCV, TensorFlow, PyTorch), ktoré boli napísané v C++, čo spôsobuje, že výhoda výkonu nie je taká veľká. Preto si väčšina vývojárov vyberá Python pre jeho jednoduchosť a čitateľnosť kódu.

- R

Jazyk R Špecializuje sa na štatistiku a analýzu údajov a je vynikajúcou voľbou na riešenie problémov dátovej vedy.

Nevýhody: Python sa ľahšie učí, má jednoduchšiu štruktúru kódu, pre Python bolo vytvorených viac knižníc a tento jazyk je univerzálnejší.

- Julia

Jazyk Julia je určený na vedecké výpočty a má vysoký výkon. Julia V poslednom čase sa stáva populárnou v oblasti strojového učenia. Tento jazyk sa zameriava na vysoký výkon. Má špeciálny systém dátových typov a kompilácie, ktorý mu umožňuje dosiahnuť rýchlosť jazykov nižšej úrovne (C, Fortran), ale je dynamicky typovaný, aby poskytoval flexibilitu a jednoduchosť písania kódu. Julia: Julia dokáže ľahko spolupracovať s kódom napísaným v iných jazykoch, ako sú C, Fortran a Python.

Nevýhody: Žiaľ, všetky výhody tohto jazyka nemohli kompenzovať jednu z jeho hlavných nevýhod. Vzhľadom na to, že jazyk je relatívne nový, nemá ešte veľkú komunitu, pomerne málo výukových programov, nie najlepšiu dokumentáciu a oveľa menej knižníc ako Python.

- Python

Pyton má rozsiahly ekosystém knižníc pre strojové učenie. TensorFlow, PyTorch, scikit-learn, Keras a ďalšie knižnice poskytujú vývojárom množstvo nástrojov na vytváranie, trénovanie a nasadzovanie modelov. To umožňuje vývojárom sústrediť sa na tvorbu algoritmov a premýšľať o architektúre aplikácie bez toho, aby venovali toľko pozornosti kódu. Python sa bez problémov integruje s kódom napísaným v iných jazykoch. To umožňuje vytvárať knižnice s používateľsky prívetivým a čitateľným rozhraním, ktoré budú fungovať rovnako rýchlo ako v jazyku C++

- Mojo

Ide o nový jazyk od spoločnosti Modular, ktorý má ohromiť svet. Je to jazyk, ktorý je úplne zameraný na strojové učenie a budovanie umelej inteligencie. Hlavnou črtou tohto jazyka je, že takmer úplne kopíruje syntax jazyka Python, je s ním spätne kompatibilný a má možnosť vytvárať funkcie aj s dynamickým aj statickým typovaním. Z niektorých systémových dôvodov je tento jazyk dokonca rýchlejší ako C++ a Rust.

Nevýhody: Existuje len jedna nevýhoda, tento jazyk je stále v beta testovaní a má uzavretý zdrojový kód.

Pri objektívnom pohľade na všetky tieto jazyky vidíme, že Python má množstvo výhod, ktoré nie sú dostupné pre iné jazyky, a nemá žiadne významné nevýhody, preto bol pre túto prácu vybraný tento jazyk

1.3 Strojové učenie

Strojové učenie je podsekcia umelej inteligencie, ktorá sa zaoberá vytváraním algoritmov a modelov, ktoré sa dokážu učiť zo súboru údajov a následne premietnuť výsledné závislosti do nových súborov údajov s podobnou štruktúrou.

Rozdiel medzi modelmi strojového učenia a priamym programovaním je v tom, že model sa neriadi vopred definovanými podmienkami programátora, ale vyvodzuje závery na základe svojich predchádzajúcich skúseností.

1.4 Hlavné typy metód strojového učenia

Existujú 3 hlavné typy metód strojového učenia:

1. Učenie s učiteľom (Supervised Learning): Konkrétny model sa trénuje na vopred definovanom súbore údajov. Model dostane súbor údajov, ktorý sa skladá z

dvoch častí. X sú všetky atribúty a parametre, od ktorých závisí hodnota odpovede. Y je odpoveď, ktorá je priradená danému záznamu. V procese trénovania by mal model nájsť závislosti medzi súborom údajov X a získanou odpoveďou Y. Neskôr model dostane len súbor znakov X a jeho úlohou bude aplikovať na tento súbor získané závislosti na predpovedanie hodnoty Y. Úlohy tohto typu modelu pozostávajú z klasifikácie (predpovedanie označenia triedy dát) a regresie (predpovedanie číselných hodnôt).

2. Učenie bez dozoru (Unsupervised Learning): v tomto prípade sú údaje neoznačené a model nepozná odpovede. Model sa v nich snaží nájsť vzor použitím svojich algoritmov. Úlohou tohto typu modelu je zhlukovať údaje (zoskupovať podobné objekty) alebo redukovať dimenzionalitu (znižovať počet znakov pri zachovaní štruktúry údajov).
3. Učenie posilnením (Reinforcement Learning): Model sa učí interakciou s prostredím a za každú vykonanú akciu je penalizovaný alebo odmeňovaný. Cieľom tohto typu modelu je naučiť sa robiť rozhodnutia s cieľom maximalizovať odmenu.

Zvyčajne zložitosť tvorby modelu závisí od konkrétnej úlohy a je veľmi situačná, ale všeobecne sa predpokladá, že algoritmy modelov strojového učenia s učiteľom sú jednoduchšie a ľahšie pochopiteľné.

Stojí za zmienku, že uvedené tvrdenie nie je vždy pravdivé. Napríklad jednoduché zhlukovanie pomocou učenia bez dohľadu môže byť oveľa jednoduchšie vytvoriť ako niektoré problémy klasifikácie riešené pomocou učenia s učiteľom.

1.5 Výber modelu

Na úspešné vyriešenie akéhokoľvek problému pomocou strojového učenia nestačí použiť prvý model, na ktorý narazíte, a napísať pre neho kód. Prvým a najdôležitejším krokom je pochopiť podstatu problému, akú otázku potrebujeme zodpovedať, aby sme tento problém vyriešili, a aký model použiť na vyriešenie tohto problému.

1.6 Hlavné modely strojového učenia

V rámci tejto práce máme vytvoriť program, ktorý nájde všetky figúrky na šachovnici a predpovie typ figúrky (čierny pešiak, biely jazdec, čierny kráľ). Keďže možnosti odpovedí sú nám vopred známe, vyplýva z toho, že máme klasifikačný problém. Na vyriešenie klasifikačnej úlohy musíme použiť model strojového učenia s učiteľom.

V ďalšom texte každá zmienka o modeli strojového učenia znamená model strojového učenia s učiteľom, pretože budeme používať len modely tohto typu

Existuje mnoho modelov na riešenie klasifikácie. Nasleduje stručný opis tých najobľúbenejších a dôvody použitia konkrétneho modelu.

Binárny rozhodovací strom

Opis: Strom, v ktorom každý uzol predstavuje test jednej z vlastností a každá vetva predstavuje výsledok tohto testu. Každý list stromu zodpovedá určitej triede alebo rozhodnutiu (STEPIK 2022).

Výhody: Jednoduchá interpretácia, nevyžaduje zložité ladenie hyperparametrov, dá sa vizualizovať.

Nevýhody: Môže byť náchylný na pretrénovanie, najmä v prípade hlbokých stromov; nie vždy sa dokáže dobre zovšeobecniť na nové údaje

Náhodný les

Opis: Ansámblový model pozostávajúci z viacerých rozhodovacích stromov (STEPIK 2022).

Výhody: Odolný voči preučeniu, schopný spracovať veľký počet prvkov, vhodný na prácu s nevyváženými údajmi.

Nevýhody: Môže sa ťažko interpretovať, vyžaduje si úpravu hyperparametrov.

Naivný bayesovský klasifikátor

Opis: Vychádza z Bayesovej vety a predpokladá nezávislosť medzi prvkami (STEPIK 2022).

Výhody: Jednoduchá implementácia, vhodný pre textové údaje.

Nevýhody: zanedbáva závislosti medzi atribútmi.

K-najbližších susedov (KNN)

Opis: Klasifikácia objektu je založená na väčšine tried medzi jeho K najbližšími susedmi (STEPIK 2022).

Výhody: Jednoduché na pochopenie, vhodné pre malé súbory údajov.

Nevýhody: Vyžaduje veľa výpočtových zdrojov s rastúcou veľkosťou údajov, citlivý na odľahlé hodnoty.

Gradient Busting (XGBoost, LightGBM, CatBoost)

Opis: Metódy súboru, ktoré postupne vytvárajú stromy a opravujú chyby predchádzajúcich stromov (STEPIK 2022).

Výhody: Veľmi vysoká presnosť, účinné na rôznych typoch údajov.

Nevýhody: Vyžaduje starostlivé ladenie parametrov, môže sa pomalšie učiť.

Metóda podporných vektorov (SVM)

Opis: Hľadá hyperplochu, ktorá maximálne oddeľuje triedy v priestore príznakov (STEPIK 2022).

Výhody: Efektívny vo vysokodimenzionálnych priestoroch, funguje dobre, keď je ťažké oddeliť triedy.

Nevýhody: Môže byť výpočtovo nákladný na veľkých súboroch údajov, citlivý na šum.

Logistická regresia

Opis: Model na binárnu a viactriednu klasifikáciu založený na logistickej funkcii (STEPIK 2022).

Výhody: Jednoduchá interpretácia, vyžaduje málo údajov, funguje dobre, keď sú triedy lineárne oddeliteľné.

Nevýhody: Nemusí zvládnuť komplexné nelineárne závislosti.

Neurónové siete

Opis: Modely inšpirované štruktúrou mozgu, pozostávajúce z neurónov a vrstiev (STEPIK 2022).

Výhody: Dokáže sa učiť zložité nelineárne závislosti, dokáže spracovať veľké množstvo údajov.

Nevýhody: Vyžaduje veľa údajov na trénovanie, výpočtovo nákladné, môže byť ťažké ich interpretovať.

1.7 Výber vhodného modelu

Výber modelu závisí od mnohých faktorov vrátane veľkosti a vlastností údajov, požiadaviek na interpretovateľnosť, výpočtových zdrojov a ďalších. Často sa experimentuje s viacerými modelmi, aby sa určilo, ktorý z nich najlepšie vyhovuje danému problému.

Teraz je čas rozhodnúť, aký typ klasifikácie potrebujeme vyriešiť. Existujú dva rôzne typy klasifikácie:

- Binárna klasifikácia. Ide o typ klasifikácie, keď potrebujeme rozdeliť všetky prichádzajúce prípady do 2 rôznych skupín. Napríklad: pravdivé - nepravdivé, ľavé - pravé atď.
- Multiklasifikácia. Typ klasifikácie, keď existujú viac ako dva varianty tried.

Musíme nájsť všetky šachové figurky a zatriediť ich. V jednom súbore šachových figúrok máme 6 rôznych druhov figúrok, pričom môžu mať 2 farby (čiernu a bielu) a ešte jednu možnosť odpovede v prípade, že je na obrázku prázdne políčko. Celkovo je k dispozícii 13 tried. Jasnou voľbou je multiklasifikácia.

Modely, ktoré boli pôvodne vytvorené pre binárnu klasifikáciu: logistická regresia, metóda podporných vektorov (SVM), binárny rozhodovací strom.
V prípade potreby je možné tieto modely prispôsobiť tak, aby riešili problém viacnásobného triedenia, napríklad pomocou algoritmov jeden proti všetkým alebo jeden proti jednému.

- Jeden proti všetkým (One-vs-All): Každá trieda sa považuje za samostatnú binárnu klasifikačnú úlohu. Pre každú triedu sa vytvorí iný strom a objekt patrí do tej triedy, ktorej strom predpovedá najvyššiu hodnovernosť.
- Jeden-v-jednom (One-vs-One): Pre každú dvojicu tried sa skonštruuje samostatný strom na rozlíšenie objektov dvoch tried. Objekt sa potom priradí k triede, ktorá zvíťazí v najväčšom počte takýchto porovnaní.

Žiaľ, aj pri použití týchto algoritmov tieto modely pri riešení multiklasifikácie vo väčšine prípadov prehrávajú s ostatnými modelmi, preto ich nebudeme používať a nebudeme sa nimi zaoberať ďalej.

Gradient bousting a K-nearest neighbours nie sú pre nás vhodné, pretože tieto modely vyžadujú veľké množstvo výpočtových zdrojov a zväčšovanie množiny údajov má veľmi negatívny vplyv na rýchlosť trénovania týchto modelov, a náš model budeme trénovať na množine údajov, ktorá obsahuje 2406 obrázkov, z ktorých každý je maticou pixelov 224x224. Zo zvyšných možností máme k dispozícii náhodný les, naivný bayesovský klasifikátor a neurónové siete. Každý z týchto modelov má svoje výhody a nevýhody, ale našou voľbou je neurónová sieť. Dôvodom je forma, v ktorej sú naše údaje reprezentované. Budeme používať súbor údajov, ktorý pozostáva z obrázkov. Aby sme tieto informácie preniesli do modelu, budeme musieť obrázky previesť do číselnej podoby. Najjednoduchším spôsobom je vytvoriť maticu pixelov z každého obrázka, kde každý pixel bude mať hodnotu od 0 do 255, táto hodnota bude určovať farbu pixelu. Niekedy sa do pixelu zapisuje tuple, ktorý určuje farby RGB, nie odtieň medzi čiernou a bielou, napríklad (255, 255, 255, 255, 255) - biela farba. Takto prezentované informácie by boli príliš nízkoúrovňové a každý jednotlivý pixel nemá prakticky žiadny vplyv na výsledok, v našom prípade je dôležité zohľadniť: v akom poradí idú tieto pixely, ich susedov a mnoho ďalších vecí. Z tohto dôvodu nebudú náhodný les a Bayesov klasifikátor schopné určiť závislosti medzi počiatočnými pozíciami pixelov a výsledkom, ktorý sa má priradiť, preto na túto úlohu použijeme neurónové siete.

1.8 Úvod do neurónových sietí

Umelá neurónová sieť je softvérová implementácia matematického modelu biologickej neurónovej siete. Jej účelom je využiť prenášané údaje, urobiť predikciu, určiť, čo je na obrázku, vypočítať, koľko bude stáť dom s danými parametrami a podobne.

Neurónové siete sa skladajú z troch typov vrstiev: vstupnej, vnútornej a výstupnej. Vstupná vrstva umožňuje zadávanie údajov do siete a musí byť zostavená tak, aby zodpovedala forme prichádzajúcich údajov. Vnútorné vrstvy sú vrstvy zodpovedné za učenie, práve v nich

sa vytvára vzťah medzi údajmi a prebieha učenie. Výstupná vrstva je vrstva, ktorá nám vypisuje odpoveď neurónovej siete vo vopred určenej forme.

Samotné vrstvy sa skladajú z jednotlivých neurónov, ktoré sú podľa určitých algoritmov prepojené s inými neurónmi v susedných vrstvách a dokážu s nimi posilniť spojenie.

Prostredníctvom neurónových sietí sa v procese trénovania opakovane odovzdávajú akékoľvek údaje, napríklad obrázky, a uvádza sa, čo sa na nich nachádza. Vďaka práci rôznych algoritmov sú neuróny prepojené tak, aby v budúcnosti pri odovzdávaní nových údajov neurónovej sieti dokázali pre tieto údaje predpovedať výsledok, napríklad odpovedať na otázku: Čo je na obrázku?.

Rozpoznávanie obrazu a počítačové videnie nie je jedinou úlohou, ktorú možno riešiť pomocou neurónových sietí. Existuje mnoho typov a kategórií neurónových sietí, ale táto práca sa priamo týka klasifikácie obrázkov, preto budeme hovoriť hlavne o CNN (Convolutional Neural Network), známej aj ako konvolučná neurónová sieť.

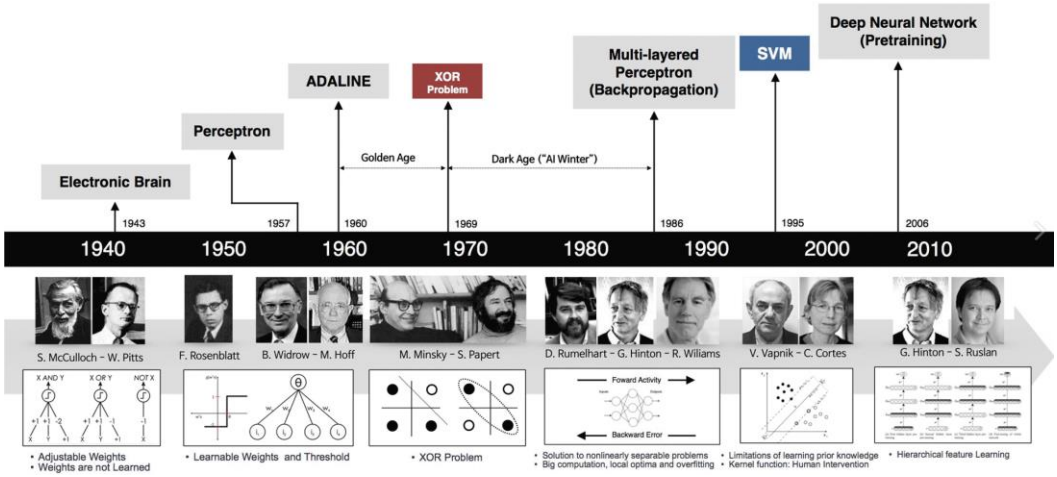
1.9 História umelých neurónových sietí

Matematický model umelého neurónu navrhli W. McCulloch a W. Pitts v 50. rokoch 20. storočia (SMITH & TOPIN 2017).

V podobe počítačového systému neurónovú sieť prvýkrát realizoval Frank Rosenblatt v roku 1960. Frank Rosenblatt vytvoril "Mark-1", ktorý bol hardvérovo-softvérovým komplexom a v jednej vrstve stelesňoval jednoduchú neurónovú sieť. Možno poznamenať, že moderné siete obsahujú 50 - 150 vrstiev a experimentálne siete dosahujú veľkosť viac ako 100 vrstiev (HE et al. 2016).

Žiaľ, ale ako vidíte na obrázku 1, po 60. rokoch záujem o neurónové siete výrazne poklesol kvôli ťažkostiam pri vykonávaní niektorých logických operácií (napr. logická operácia XOR) a nemožnosti získať praktické výsledky. Dominantným prístupom na vytváranie umelej inteligencie sa stali expertné systémy. Boli v podstate pokročilou encyklopédiou znalostí v určitej oblasti.

V roku 2012 záujem o hlboké neurónové siete vzrástol, t. j. siete s počtom vnútorných vrstiev väčším ako jedna. Spočiatku bol počet vrstiev takýchto neurónových sietí približne 3 až 10. Ale vďaka účinnosti prístupu k zvyšovaniu počtu vrstiev sa rýchlo objavili neurónové siete s desiatkami vrstiev



Obr. 1 História neurónových sietí

Zdroj: Nikolenko S. I., Kadurin, Deep Learning 2018 (NIKOLENKO et al. 2018).

Od roku 2012 sa neurónové siete stali dominantným spôsobom riešenia problémov v strojovom učení. Stalo sa tak hneď z niekoľkých dôvodov: technologický pokrok, pokročilejšia architektúra samotných neurónových sietí a pravdepodobne hlavným dôvodom bol nástup výkonných grafických kariet, ktoré umožnili trénovať veľké a zložité neurónové siete pomocou GPU.

1.10 Štruktúra neurónu

V kontexte neurónovej siete je neurón najzákladnejšou jednotkou spracovania. Neurónová sieť je založená na tom, ako funguje ľudský mozog. Môžeme teda povedať, že napodobňuje spôsob, akým si biologické neuróny navzájom odovzdávajú signály.

Každý neurón sa skladá zo vstupov, váh, sumátora, aktivačnej funkcie a výstupu. Pokúsime sa zistiť, čo je za čo zodpovedné. (obrázok 2)

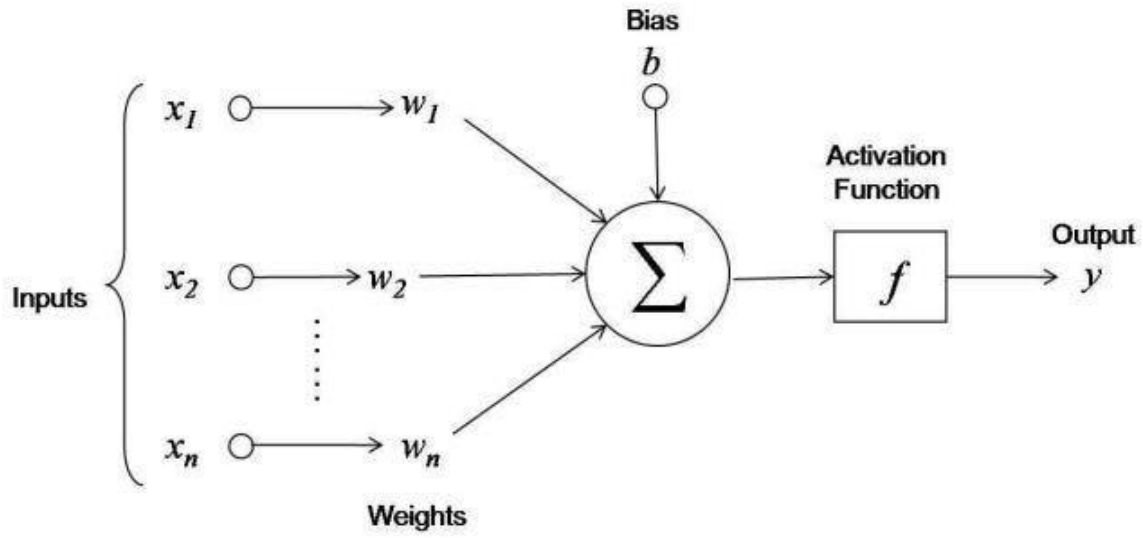
- Vstupy - každý parameter, ktorý môže ovplyvniť výsledok. Napríklad pri posudzovaní domu sú to: počet izieb, poschodie, dostupnosť parkovania a podobne. Zvyčajne sa ako vstupy uvádzajú číselné hodnoty alebo kategorické atribúty v štýle 0 - 1.
- Váhy - každý z parametrov ovplyvňuje konečný výsledok, ale ovplyvňujú ho s rôznou silou. Váhy sú koeficientom pre každý parameter, ktorý by mal upravovať mieru vplyvu tohto parametra na výsledok.
- Sumátor - táto časť sčíta všetky vstupy pomocou jedného pomerne jednoduchého výrazu: $x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + \cdots + x_n * w_n + b$ kde x je jeden parameter, w je jeho váha a b je bias.
- Aktivačná funkcia - v neurónových sieťach sa používa na zavedenie nelinearity do modelu. Bez aktivačnej funkcie bude neurónová sieť aj s viacerými vrstvami ekvivalentná lineárnemu modelu, pretože kombinácia lineárnych transformácií zostane

lineárna. Zavedenie nelinearity umožňuje neurónovej sieti modelovať zložitejšie závislosti v údajoch. Hodnota vypočítaná sumátorom sa pošle aktivačnej funkcii ako vstup, spracuje sa a potom sa pošle na výstup. Podrobnejšie informácie o aktivačných funkciách budú opísané ďalej, kde si objasníme ich typy a na čo sú určené.

- Výstup - táto časť je zodpovedná za zobrazenie získaných výsledkov

Nemali by sme zabúdať ani na tzv. voľnú váhu, bias (prah).

V neurónových sieťach je parameter bias jednou z váh, ale nie je násobený vstupným signálom. Namiesto toho sa bias pridáva k váženému súčtu vstupov pred použitím aktivačnej funkcie. Zavedenie parametra bias do neurónu umožňuje modelu pružnejšie sa prispôsobiť údajom a zlepšiť všeobecnosť



Obr. 2 Matematický opis neurónu

Zdroj: Nikolenko S. I., Kadurin, Deep Learning 2018 (NIKOLENKO et al. 2018).

1.11 Všeobecné informácie o tom, ako sa trénujú neurónové siete

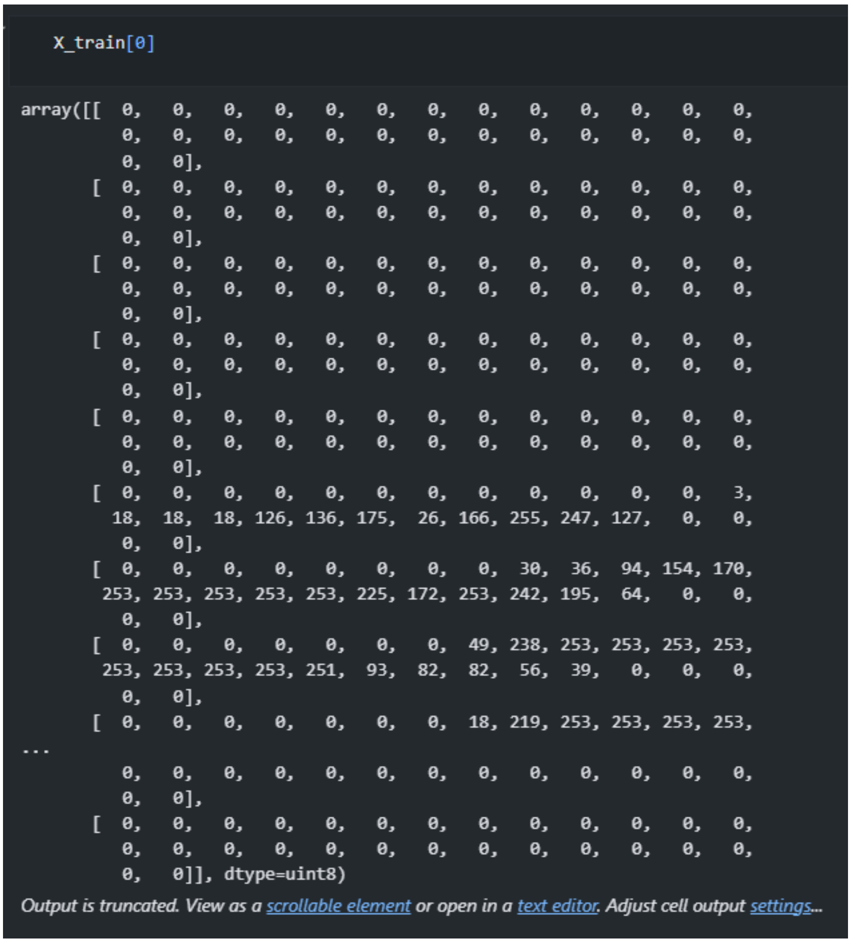
Zoberme si typickú úlohu pre neurónovú sieť, aby sme ju viac priblížili téme práce, napríklad určiť z obrázka, kde je mačka a kde pes.

Najprv sa zamyslíme nad tým, ako by sa táto úloha riešila bez použitia neurónových sietí. Uvedomme si, že obrázok je súbor pixelov. Pixely sú v počítači reprezentované ako matica čísel, najčastejšie vyzerá takto:

```
X_train[0].shape
✓ 0.0s                                                    MagicPython
(28, 28)
```

Obr. 3 Príkaz shape

Na obrázkoch 3,4 sme sa pomocou príkazu shape naučili, že veľkosť matice je 28x28 pixelov.



```
X_train[0]

array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
...
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Obr. 4 Výsledok príkazu shape

Tu sme sa práve pozreli na to, ako vyzerá samotná matica v číselnej podobe. Ako vidíme, všetko sú to čísla, pričom každé číslo predstavuje odtieň farby od bielej po čiernu, kde 0 je úplne čierna, t. j. žiadna farba, a 255 je biela. A kde sú čísla, tam sú aj vzory, stačí ich len nájsť.

Mačky, rovnako ako psy, majú veľa charakteristických znakov, rovnako ako psy, vymenovanie týchto znakov bude trvať príliš dlho. Ale stojí za to zvážiť aj ďalšie okolnosti, fúzy, labky, chvost, uši a mnoho ďalšieho majú obe zvieratá. Ako teda zistíme, kto je kto? S najväčšou pravdepodobnosťou to musíme urobiť tak, že zohľadníme najmenšie rozdiely, ako sú: tvar, dĺžka, uhol a mnoho ďalších vecí. Boli pokusy robiť to ručne, tesne pred érou neurónových sietí, ale výsledky boli nevýrazné. Ako si uvedomujete, objem funkcií je príliš veľký.

Úlohou neurónovej siete je teda obsiahnuť potrebný súbor znakov, ktorý jej umožní rozlíšiť jeden objekt od druhého. To sa vykonáva trénovaním na obrovskom množstve údajov. Model porovnáva vstupné údaje a snaží sa zvoliť váhy pre jednotlivé parametre tak, aby sme po dosadení do opísanej rovnice dostali rovnako dobré výsledky pre akékoľvek údaje. V prvých fázach nie sú výsledky veľmi pôsobivé, chyba je zvyčajne takmer 90 %, model sa snaží znova a znova a znova. Zakaždým na novej epoche s ohľadom na výsledky minulej epochy, aby neopakoval svoje chyby. Nakoniec so správne vybudovanou neurónovou sieťou získame model, ktorý dokáže s istotou rozlíšiť mačku od psa.

1.11.1 Proces vytvárania neurónovej siete

Architektúra neurónovej siete takmer úplne závisí od úlohy. V závislosti od úlohy si budeme musieť vybrať: (Nikolenko 2018)

- Počet vrstiev, z ktorých sa bude neurónová sieť skladať
- Koľko neurónov bude v týchto vrstvách

Okrem toho budeme musieť vybrať aj nasledujúce komponenty, ktoré budú podrobnejšie opísané nižšie: (Nikolenko 2018)

- Systém na inicializáciu váh neurónov
- Aktivačná funkcia neurónu
- Algoritmus na nastavenie váh neurónov

Toto je určite zoznam najbežnejších komponentov, o ktorých budeme musieť rozhodnúť, ale v tejto fáze nám na všeobecné pochopenie princípu postačia.

2 SÚBOR ÚDAJOV CHESSCOG

Chesscog je súbor údajov určený na rozpoznávanie a klasifikáciu šachových figúrok na obrázkoch šachovnice. Bol vytvorený na podporu vývoja a testovania systémov počítačového videnia, ktoré dokážu automaticky zisťovať stav šachovej hry v reálnom čase. (Wolflein 2021)

2.1 Obsah súboru údajov

- Obrázky šachovnice: Súbor údajov obsahuje množstvo fotografií šachovnice s figúrkami v rôznych konfiguráciách. Snímky sú zhotovené z rôznych uhlov a za rôznych svetelných podmienok.

Označenie: Každý obrázok je doplnený o metadáta vrátane informácií o type a farbe šachových figúrok na každom políčku šachovnice. Tieto údaje sa často uvádzajú vo formáte JSON, ktorý pre každé políčko uvádza, aká figúrka sa na ňom nachádza (ak existuje).

Obr. 5 Príklad obrázka zo súboru údajov Chesscog.

Zdroj: Dataset ChessCog

2.2 Používanie súboru údajov

- trénovanie modelov strojového učenia: Súbor údajov možno použiť na trénovanie modelov hlbokého učenia schopných rozpoznávať a klasifikovať šachové figúrky z obrázkov.

- Testovanie a hodnotenie výkonu: Chesscog poskytuje rôzne údaje na testovanie presnosti a spoľahlivosti rozpoznávacích algoritmov.

2.3 Výhody a funkcie

- Vizuálna rozmanitosť: Súbor údajov obsahuje snímky nasnímané za rôznych svetelných podmienok a z rôznych uhlov, čo pomáha pri vývoji adaptívnych a robustných systémov rozpoznávania.
- Široká škála aplikácií: Môže sa použiť nielen na úlohy počítačového videnia, ale aj na vytvorenie aplikácií, ktoré automatizujú proces analýzy šachových partií, alebo na školenie systémov, ktoré pomáhajú šachistom analyzovať partie.

2.4 Možné ťažkosti

Zložitosť označovania: Presné označovanie tvarov v obrázkoch si vyžaduje pozornosť k detailom a môže byť časovo náročné.

Spracovanie obrázkov: Obrázky sa musia starostlivo spracovať, aby sa odstránili skreslenia spôsobené perspektívou a osvetlením, čím sa zabezpečí vysoká kvalita rozpoznávania.

2.5 Zhodnotenie súboru údajov

Súbor údajov Chesscog je veľmi zaujímavý pre vývojárov systémov umelej inteligencie v oblasti šachu a počítačového videnia. Poskytuje údaje potrebné na vytváranie a testovanie algoritmov schopných rozpoznávať šachové figúrky a analyzovať partie, čím sa stáva cenným zdrojom v príslušných výskumných a komerčných projektoch.

3 ÚPRAVA FOTOGRAFIE

Pred vytvorením modelu strojového učenia bola použitá myšlienka rozdelenia podobného obrázka na 64 menších, jeden obrázok na šachové políčko. Urobilo sa to s cieľom uľahčiť úlohu neurónovej siete a získať presnejšie odpovede

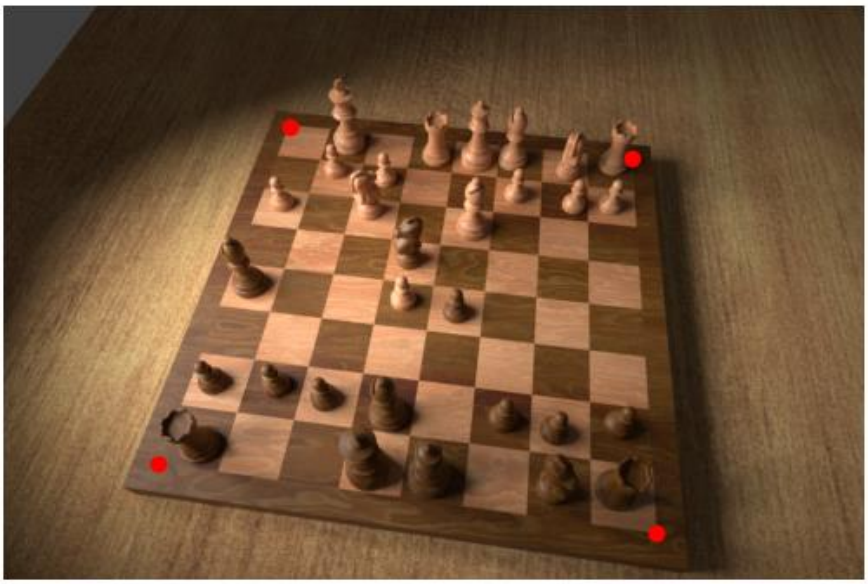
Táto fáza je veľmi dôležitá pre úspech celého projektu, pretože kvalita rozpoznávania šachových figúrok priamo závisí od toho, ako presne sú jednotlivé políčka šachovnice izolované od ostatných. Čistota a presnosť zdrojových údajov priamo ovplyvňuje tréning modelov strojového učenia, ako aj ich schopnosť správne interpretovať a klasifikovať objekty na nových obrázkoch.

3.1 Príprava obrazu

Prvým krokom bolo predbežné spracovanie šachovnicových snímok, ktoré zahŕňalo korekciu osvetlenia a kontrastu, ako aj filtrovanie šumu. To je potrebné na zlepšenie vizuálnych vlastností obrazu, čo uľahčuje následnú extrakciu hraníc a rohov šachovnice. Napríklad zlepšenie kontrastu pomáha jasnejšie zvýrazniť línie šachovnice, čo je rozhodujúce pre presnosť detekcie rohov.

3.2 Detekcia rohov

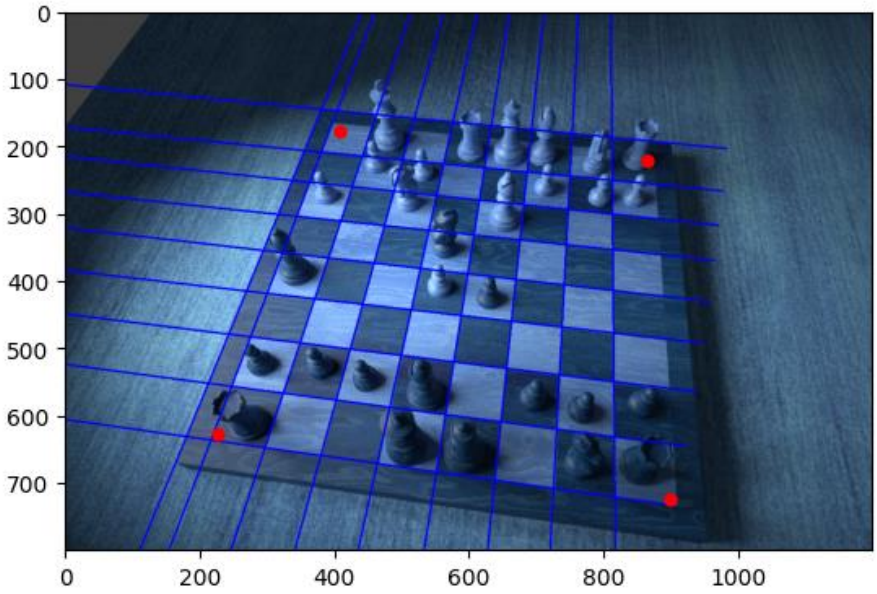
V tomto kroku sa na detekciu štyroch rohov šachovnice (Obrázok 6) používajú algoritmy počítačového videnia. Tento proces zahŕňa použitie algoritmov na detekciu hraníc, ako sú Sobelove detektory hraníc, a techniky transformácie perspektívy na korekciu a zarovnanie obrazu šachovnice. Tým sa zabezpečí správne umiestnenie políčok a ich príprava na presnú segmentáciu. Táto korekcia je potrebná na následné presné spracovanie každej bunky jednotlivo.



Obr. 6 Zistenie uhlov dosky

3.3 Detekcia línie

Pomocou algoritmov počítačového videnia boli vytvorené funkcie na nájdenie všetkých priamok na obrázku. (Obrázok 7) Toto riešenie je založené na nasledujúcom princípe: algoritmus hľadá všetky sekvencie pixelov približne rovnakej farby, ktoré sú dostatočne dlhé a úzke na to, aby sa mohli považovať za čiar

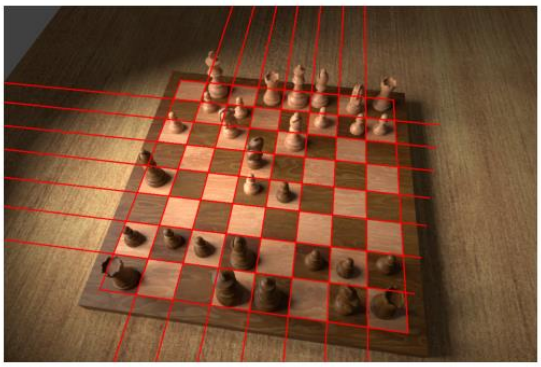


Obr. 7 Hľadanie priamok

3.4 Zrezanie línie

Ďalšou vecou, ktorú sme museli urobiť, bolo skrátiť čiary tak, aby nepresahovali šachovnicu. (Obrázok 8) Náš algoritmus hľadá presne 9 vertikálnych a 9 horizontálnych čiar a ak nájde čiaru tam, kde by nemala byť, potom algoritmus nenájde jednu z čiar tam, kde by mala

byť. Preto je veľmi dôležité uistiť sa, že hľadáme čiary len vo vnútri obdĺžnika, ktorý získame spojením bodov rohov.



Obr. 8 Nastavenie vyhľadávania línie.

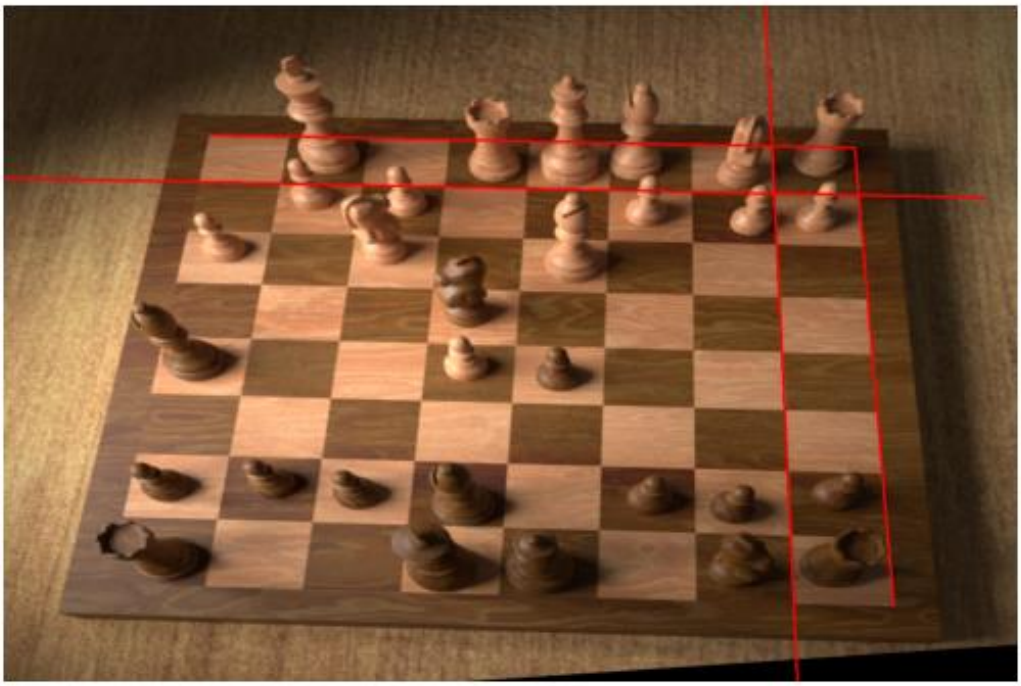
Obr. 9 Grafická transformácia

3.5 Transformácia obrazu

Pomocou algoritmov počítačového videnia a grafických transformácií sme zmenili obrázok tak, aby viac pripomínal pohľad zhora bez zohľadnenia náklonu. To nám pomôže lepšie rozdeliť na jednotlivé bunky. (Obrázok 9)

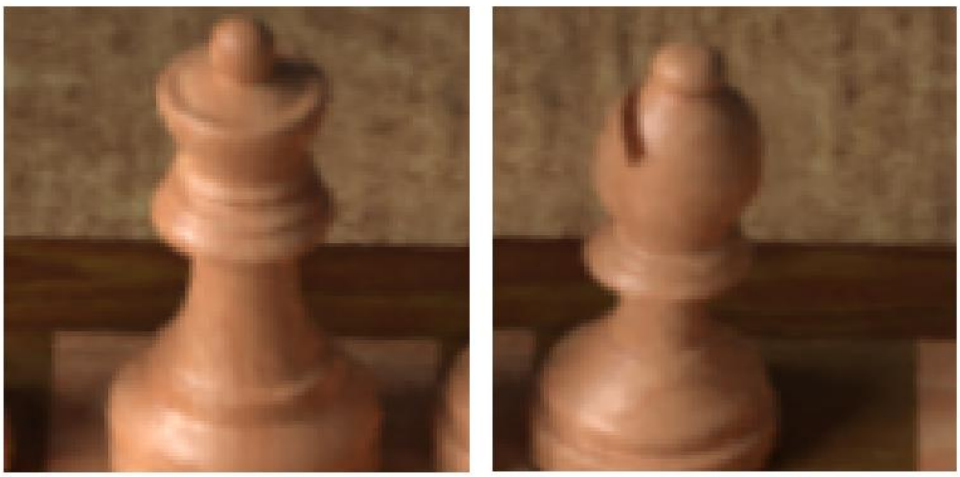
3.6 Delenie na bunky

Pomocou zistených uhlov sa obraz dosky rozdelí na 64 rovnakých častí zodpovedajúcich štvorcom (pozri Obrázok 10). Každý štvorec sa oreže a uloží ako samostatný súbor, čo umožňuje podrobnú analýzu a tréning modelov na každom štvorci osobitne. Toto rozdelenie umožňuje vyčleniť každý tvar do samostatného obrazu, čím sa zjednoduší úloha ich kategorizácie



Obr. 10 Nájdenie 1 bunky

Potom algoritmus určí, v akom smere bol náklon, či vľavo alebo vpravo, ako ďaleko postava stojí a podobne, a na základe toho pridá na okraje získaných obrázkov odsadenie, aby mohli zachytiť postavu ako celok. Výsledok môžete vidieť na obrázku 11



Obr. 11 Výsledok úpravy fotografie

3.7 Overovanie a nastavovanie

Po separácii sa skontroluje kvalita a presnosť extrakcie buniek. Je potrebné sa uistiť, že každá bunka je správne zarovnaná a neobsahuje časti susedných buniek. Tým sa zabezpečia čisté údaje na trénovanie a zabráni sa možným chybám rozpoznávania v dôsledku nesprávne orezaných obrázkov. Ak sa zistia nepresnosti, hranice sa opravia.

4 PREDBEŽNÉ SPRACOVANIE A ČÍTANIE ÚDAJOV

Pred začatím trénovania modelov na rozpoznávanie šachových figúrok je dôležité vykonať kvalitné predbežné spracovanie a organizáciu údajov. Tento krok je potrebný na zabezpečenie toho, aby bol model natrénovaný na údajoch správneho formátu a kvality, čo následne zlepší presnosť rozpoznávania.

4.1 Načítanie údajov

Funkcia load_data načíta obrázky a ich príslušné metadáta z JSON. Obrázky sú prenesené do rovnakej veľkosti, čo je dôležité pre trénovanie neurónovej siete, pretože všetky vstupné údaje by mali byť štandardizované.

Privedenie všetkých obrázkov do rovnakej veľkosti zjednodušuje spracovanie obrázkov neurónovou sieťou a znižuje pravdepodobnosť chýb vo fáze trénovania. Môže to však viesť aj k strate niektorých detailov, najmä ak pôvodné obrázky obsahujú veľa malých detailov. Z tohto dôvodu sme obrázky rozdelili na menšie časti, aby sme sa mohli sústrediť vždy na jednu vec

4.2 Zmena údajov

Funkcia change_data nielen mení veľkosť obrázkov, ale ich aj reorganizuje do nových adresárov na základe obsahu zadaného v súbore JSON. Tým sa vytvorí štruktúrované dátové úložisko, v ktorom sa každá trieda obrázkov nachádza vo vlastnom adresári.

Táto organizácia súborov zjednodušuje ďalšiu prácu s údajmi a umožňuje ich jednoduché rozdelenie na trénovaciu, validačnú a testovaciu sadu. Vyžaduje si však ďalší čas a zdroje vo fáze predbežného spracovania údajov.

4.3 Vytváranie súborov údajov

Funkcie create_first_dataset a create_second_dataset načítajú spracované údaje a vytvárajú súbory údajov na trénovanie, validáciu a testovanie. Je dôležité poznamenať, že v druhom prípade je adresár s prázdnymi bunkami dočasne vylúčený z procesu, čo nám umožňuje zamerať sa na údaje obsahujúce tvary.

Diferencované vytváranie súborov údajov umožňuje prispôsobiť proces trénovania a zlepšiť kvalitu trénovania pomocou relevantnejších údajov.

4.4 Optimalizácia a rozšírenie údajov

Optimalizácia načítavania údajov a rozširovanie údajov zohrávajú kľúčovú úlohu pri trénovaní neurónových sietí, najmä v úlohách počítačového videnia. (Wölflein 2021)

4.4.1 Optimalizácia načítavania údajov

Na urýchlenie procesu učenia a skrátenie času odozvy pri trénovaní modelu sa údaje vopred načítavajú, ukladajú do vyrovnávacej pamäte a načítavajú asynchrónne. Pomocou

funkcie tf.data.AUTOTUNE TensorFlow automaticky optimalizuje procesy ukladania do vyrovnávacej pamäte a prefetchingu, aby sa efektívnejšie využívali zdroje CPU a GPU:

- Ukladanie do vyrovnávacej pamäte (cache()): Údaje sa stiahnu raz a uložia sa do pamäte alebo na disk, čím sa urýchli prístup k nim v nasledujúcich tréningových epochách.
- Predbežné načítanie (prefetch()): Zatiaľ čo model spracováva jeden balík údajov, ďalší balík údajov sa už pripravuje, čím sa minimalizujú prestoje a urýchľuje proces učenia..

4.4.2 Rozšírenie údajov

Rozšírenie údajov je proces vytvárania nových trénovaných údajov z existujúcich údajov náhodnou úpravou údajov (napr. otáčaním, zmenou mierky, orezaním alebo zmenou farebnej schémy obrázkov). (Wölflein 2021)

Je to technika ktorá umelo zvyšuje dimenzionalitu súboru trénovaných údajov náhodnými, realistickými zmenami pôvodných trénovaných vzoriek. To pomáha zlepšiť všeobecnosť modelu a znížiť pretrénovanie, najmä ak je množstvo údajov obmedzené. Funkcia create_augmentation_layer() vytvorí augmentačnú vrstvu, ktorú možno aplikovať na súbor trénovaných údajov:

- data_augmentation: Aplikuje zadané transformácie (napr. rotácie, škálovanie, posunutie) na obrázky za behu počas trénovania, čím poskytuje väčšiu rozmanitosť príkladov na trénovanie.

4.4.1 Konverzia údajov a dávkovanie

Po rozšírení a optimalizácii sa údaje zoskupia do dávky batch(batch_size)), čo modelu umožňuje spracovať viacero vzoriek súčasne, čím sa zvyšuje efektivita trénovania. Dávkovanie zahŕňa aj mapovacie transformačné funkcie, ako napríklad replace_values, ktoré môžu pred trénovaním vykonať potrebné úpravy štítkov alebo atribútov údajov:

Mapovanie funkcií na údaje: ds_train_first.map(lambda image_batch, labels_batch: (image_batch, replace_values(labels_batch))) aplikuje funkciu replace_values na každú dávku štítkov, čo umožňuje prispôsobiť štítky požiadavkám modelu.

5 VYTVÁRANIE A TRÉNOVANIE MODELOV

5.1 Príprava údajov na trénovanie

Pred tréningom modelov sa venuje veľká pozornosť predbežnému spracovaniu údajov, ktoré je rozhodujúcim krokom pre úspešný tréning neurónových sietí.

5.2 Dva modely

Pri vytváraní neurónových sietí bolo rozhodnuté rozdeliť úlohu na niekoľko samostatných častí a pre každú z nich vytvoriť inú neurónovú sieť.

Prvý model bude musieť určiť, či sa na danom obrázku nachádza figúrka alebo nie.

Druhý model bude musieť určiť, aký typ figúrky je na poskytnutom obrázku.

Toto sa robí z viacerých dôvodov. Modely, ktoré vytvárame, musia byť na niečo potrebné a nejakým spôsobom použiteľné na reálne úlohy. A najčastejšie sa modely strojového učenia hodnotia nielen na základe technických ukazovateľov, ako je presnosť, ale aj na základe biznis ukazovateľov a požiadaviek zákazníkov.

Možno keby sme toto všetko urobili v rámci jedného modelu, tak by štatisticky takýto model fungoval lepšie a presnejšie, pretože pri takejto štruktúre, akú máme teraz, ak urobíme chybu v prvej fáze, ovplyvní to druhú fázu, pretože druhý model v tejto fáze nepredpokladá prítomnosť prázdnych buniek.

Tento projekt však plánujeme ďalej rozvíjať do podoby plnohodnotnej aplikácie na vyhodnocovanie šachových pozícií, posudzovanie najlepších ťahov a modelovanie šachových partií na základe obrázkov, preto je pre nás dôležitá aj rýchlosť spracovania. Zo štruktúry vytvorených neurónových sietí, informácie o ktorých nájdete v ďalších kapitolách, vidíme, že prvý model s konfiguráciou, ktorú budeme používať, má len 16488258 parametrov a druhý 47662733, čo je takmer 3-krát viac.

Na základe toho bude model, ktorý určuje, či je pole prázdne alebo nie, pracovať rýchlejšie ako model, ktorý klasifikuje figúrky. Na šachovnici je 64 políčok a v jednom a tom istom čase môže byť maximálne 32 figúr, to znamená, že pri maximálne zaplnenej šachovnici budeme mať 32 prázdnych políčok a čím ďalej hra pokračuje, tým viac prázdnych políčok, preto bolo rozhodnuté úlohu rozdeliť. Ušetrí nám to čas a druhému modelu dá možnosť sústrediť sa len na klasifikáciu figúrok.

Taktiež v budúcnosti plánujeme vyriešiť problém, že druhý model nemôže fyzicky opraviť chybu prvého a v budúcnosti bude mať takúto možnosť, viac o tom na konci práce.

Taktiež bude na konci práce demonštrovaná praktická aplikácia nášho algoritmu na reálnom obrázku šachovnice, z ktorej bude zrejmé, že pomerne často nepotrebujeme zistiť,

ktorá figúrka sa na danom políčku nachádza, a stačí nám vedieť, či tam niečo je alebo nie, napríklad pri výpočte trajektórie ťahu dámy.

5.3 Výber a konfigurácia modelu

Výber modelov je založený na použití vopred natrénovaných architektúr, ktoré môžu výrazne urýchliť proces učenia a zlepšiť kvalitu rozpoznávania prostredníctvom prenosu znalostí.

Predtrénované modely, ako napríklad VGG16 a ResNet101, sú vybrané z dôvodu ich vynikajúceho výkonu v úlohách počítačového videnia. Model VGG16 je známy svojou jednoduchosťou a efektívnosťou pri rozpoznávaní textúr, zatiaľ čo ResNet101 ponúka hlboké zvyškové siete, ktoré umožňujú efektívne trénovať veľmi hlboké siete bez straty výkonu.

Optimalizátor Adam je vybraný pre svoju schopnosť prispôsobiť rýchlosť učenia pre každý parameter, vďaka čomu je preferovaný pre úlohy náročné na údaje a parametre. Adam poskytuje rýchlejšiu a stabilnejšiu konvergenciu v porovnaní s inými optimalizátormi, napríklad SGD.

5.3.1 Model VGG16

- Štruktúra: Používa sa základný model VGG16 bez vrchných vrstiev, ku ktorému sa pridávajú vlastné vrstvy pre úlohu klasifikácie. To umožňuje prispôsobiť architektúru špecifickým potrebám konkrétnej aplikácie pri zachovaní osvedčených silných stránok pôvodnej architektúry (SIMONYAN & ZISSERMAN 2014).
- Trénovanie: Na trénovanie nových vrstiev sa vykonáva počiatočná fáza trénovania so zmrazenými váhami predtrénovaného modelu, čo znižuje riziko pretrénovania a urýchľuje proces trénovania (SMITH & TOPIN 2017). Jemné ladenie sa potom vykonáva rozmrazením niektorých vrstiev s cieľom ďalej spresniť váhy pre špecifickosť úlohy.
- Optimalizácia: Na dosiahnutie stabilnej konvergencie sa používa optimalizátor Adam s nízkou počiatočnou rýchlosťou učenia, čo uľahčuje presnejšie ladenie váh bez rizika výrazných výkyvov počas trénovania (KERAS 2022).

5.3.2 Model ResNet101

- Štruktúra: Podobne ako pri modeli VGG16 sa používa základný model ResNet101 s pridaním vlastných vrstiev na prispôsobenie sa úlohe. Toto rozšírenie modelu umožňuje použitie hlbokých reziduálnych sietí, čo zlepšuje učenie hlbokých sietí zlepšením gradientového toku (Czyzewski et al. 2020).

- Trénovanie: Stratégia trénovania zahŕňa dva kroky: predtrénovanie pridaných vrstiev a jemné ladenie. Toto rozdelenie umožňuje, aby sa nové vrstvy spočiatku ladili na vysokej úrovni abstrakcie a potom sa celý model mohol jemnejšie vyladiť pre konkrétnu úlohu.
- Optimalizácia: Proces jemného ladenia využíva redukovaný krok trénovania s cieľom minimalizovať straty, čo pomáha dosiahnuť vysokú presnosť a zároveň zachovať robustnosť procesu učenia (SMITH & TOPIN 2017).

5.4 Fine Tuning modelov hlbokého učenia

Jemné ladenie (Fine tuning) je proces prispôsobenia vopred natrénovaného modelu konkrétnej úlohe jeho pretrénovaním na novom, najčastejšie menšom súbore údajov. Táto metóda sa široko používa v hlbokom učení na zlepšenie kvality modelu na konkrétnej úlohe, najmä ak je k dispozícii obmedzené množstvo trénovaných údajov (Czyzewski et al. 2020; SMITH & TOPIN 2017).

Tento prístup šetrí zdroje a čas na trénovanie, pretože väčšina modelu je už natrénovaná na veľkej a rôznorodej množine údajov, ako je napríklad ImageNet pre úlohy počítačového videnia. Použitie vopred natrénovaných váh ako východiskového bodu urýchľuje proces trénovania a často vedie k lepšej kvalite ako trénovanie od začiatku, najmä ak je trénovacích údajov nedostatok (CHOLLET 2017; SMITH & TOPIN 2017).

5.4.1 Použitie Fine Tuning v našej práci

V našej práci sme použili modely VGG16 a ResNet101 predtrénované na súbore údajov ImageNet. Tieto modely boli vybrané pre ich vynikajúci výkon v úlohách klasifikácie obrázkov a ich schopnosť efektívne prenášať znalosti na nové úlohy.

Jemné doladenie sme použili takto:

- Inicializácia modelov s predtrénovanými váhami: Modely boli načítané s váhami získanými po tréningu na ImageNet. To nám poskytlo silný základ na rozpoznávanie rôznych obrazových prvkov.
- Rozmrazenie a predtrénovanie posledných vrstiev: V procese jemného doladenia sme rozmrazili posledné vrstvy modelov a predtrénovali ich na našich cieľových údajoch - obrázkoch šachovnice. To nám umožnilo špecializovať modely na vlastnosti obrázkov šachovnice.
- Optimalizátor a rýchlosť učenia: Na pretrénovanie bol použitý optimalizátor Adam s nízkou rýchlosťou učenia (počiatočná rýchlosť 0,0001), čo pomohlo zabrániť zničeniu predtrénovaných funkcií a zdokonaliť ich pre našu špecifickú úlohu (KERAS 2022).

5.4.2 Výhody Fine Tuning

Použitie jemného doladenia nám umožnilo dosiahnuť výrazne vyššiu presnosť ako tréning od začiatku, najmä vzhľadom na obmedzené množstvo špecializovaných údajov. Modely vykazovali lepšie rozpoznávanie rôznych typov šachových figúrok a ich umiestnenia na šachovnici, čo je pre našu prácu rozhodujúce.

Týmto prístupom sme tiež znížili čas a výpočtové zdroje potrebné na trénovanie pri zachovaní vysokej kvality modelovania.

5.5 Trénované modely

Použitie pokročilých techník predspracovania, rozšírenia a optimalizácie nám umožňuje maximalizovať potenciál predtrénovaných modelov na zlepšenie kvality a presnosti klasifikácie šachových figúr. Vytvorené modely sú prispôsobené špecifikám úlohy s ohľadom na osobitosti šachových údajov, čo zabezpečuje vysokú presnosť a spoľahlivosť pri riešení zadaných úloh

6 TESTOVANIE MODELOV

6.1 Rozdelenie údajov do množín

Na začiatku boli údaje rozdelené do 3 súborov: tréningový, testovací a validačný. Trénovacia množina sa používa na trénovanie modelu, validačná množina sa používa na ladenie parametrov a kontrolu priebežných výsledkov a testovacia množina sa používa na konečné hodnotenie výkonnosti modelu. Toto rozdelenie pomáha predchádzať pretrénovaniu a poskytuje poctivejšie posúdenie schopnosti modelu zovšeobecňovať (SCIKIT-LEARN 2024).

6.2 Metriky hodnotenia

Pre úlohy binárnej klasifikácie sa bežne používajú tieto metriky: Presnosť (Accuracy), Správnosť (Precision), Návratnosť (Recall), F1 metrika (F1-measure), Krivka ROC (ROC curve) .

Pre klasifikáciu viacerých tried: Presnosť viacerých tried, Matica zámen (Confusion Matrix), metriky Precision, Recall a F1-measure pre každú triedu (SCIKIT-LEARN 2024).

Tieto metriky budeme používať na kontrolu kvality.

6.2.1 Accuracy

- Definícia: Presnosť klasifikácie meria celkový podiel správne klasifikovaných prípadov vzhľadom na všetky príklady. Je to jedna z najintuitívnejších dostupných metrík (SCIKIT-LEARN 2024).
- Uplatnenie: Presnosť funguje dobre, keď sú triedy vyvážené a chyby rôznych tried sú rovnako významné.

  Vzorec 1 "Accuracy"

$$\text{Accuracy} = \frac{Počet\ správnych\ predpovedí}{\text{Celkový počet predpovedí}}$$

  Zdroj: scikit-learn, Metrics and scoring: quantifying the quality of predictions (SCIKIT-LEARN 2024).

V našom prípade nie je zjavná nerovnováha tried, takže tejto metrike môžeme dôverovať.

6.2.2 Precision

- Definícia: Presnosť predpovede udáva, aká časť výsledkov označených ako pozitívne je skutočne pozitívna..
- Použitie: Užitočné, keď sú náklady na falošné pozitívne výsledky vysoké. Napríklad pri filtroch spamu je dôležité neoznačiť normálny e-mail ako spam.

Vzorec 2 "Precision"

$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Zdroj: scikit-learn, Metrics and scoring: quantifying the quality of predictions (SCIKIT-LEARN 2024).

Toto je pre nás v súčasnosti asi najdôležitejšia metrika, pretože je veľmi dôležité, aby sa našli všetky prázdne polia a žiadne z nich nebolo náhodne označené ako vyplnené

Prázdne pole je negatívny výsledok, vyplnené pole je pozitívny výsledok.

6.2.3 Recall

- Definícia: Návratnosť meria, aký podiel skutočných pozitívnych prípadov bol modelom identifikovaný ako pozitívny.
- Uplatnenie: Dôležité, keď sú náklady na chýbajúce pozitívne výsledky vysoké, napríklad pri lekárskych diagnostických testoch.

Vzorec 3 "Recall"

$$\text{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Zdroj: scikit-learn, Metrics and scoring: quantifying the quality of predictions (SCIKIT-LEARN 2024).

6.2.4 F1 – Score

- Definícia: Miera F1 je harmonický priemer medzi presnosťou a návratnosťou. Snaží sa vyvážiť presnosť a úplnosť v jednom čísle.
- Použitie: Používa sa, keď sa vyžaduje rovnováha medzi presnosťou a návratnosťou, najmä keď je rozdelenie tried nevyvážené.

Vzorec 4 "F1-Score"

$$\text{F1} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Zdroj: scikit-learn, Metrics and scoring: quantifying the quality of predictions (SCIKIT-LEARN 2024).

6.2.5 Krivka ROC

Krivka ROC (Receiver Operating Characteristic curve) je grafická metóda na hodnotenie kvality binárnych klasifikátorov, ktorá sa široko používa na analýzu presnosti diagnostických testov a prediktívnych modelov v rôznych oblastiach vrátane medicíny a strojového učenia.

Os abscis (os X) predstavuje "mieru falošnej pozitivity" (FPR), známu aj ako špecifickosť.

Ordinátna os (os Y) predstavuje "True Positive Rate" (TPR), známu aj ako citlivosť alebo výťažnosť.

Krivka ROC sa vykresľuje zmenou klasifikačného prahu. Pre každú prahovú hodnotu sa vypočítajú hodnoty TPR a FPR a bod s týmito súradnicami sa vyznačí na grafe.

Plocha pod krivkou ROC (AUC - Area Under Curve) slúži ako miera presnosti klasifikácie. AUC rovná 1 znamená dokonalú klasifikáciu, AUC 0,5 znamená, že nie je lepšia ako náhodné hádanie

Krivka ROC hodnotí, ako dobre je model schopný rozlišovať medzi dvoma triedami (napr. chorý a zdravý) pri všetkých možných klasifikačných prahoch (SCIKIT-LEARN 2024).

6.2.6 Presnosť (Accuracy) pri klasifikácii viacerých tried

Presnosť v kontexte klasifikácie viacerých tried odhaduje celkový podiel správnych predpovedí modelu vzhľadom na všetky predpovede. Vzorec zostáva rovnaký ako pri binárnej klasifikácii, ale berie do úvahy všetky triedy.

V situácii s viacerými triedami to znamená, že presnosť meria, ako dobre model identifikuje každú triedu spomedzi všetkých dostupných tried. Je to jednoduchá a intuitívna metrika, ale môže byť zavádzajúca v prípadoch nevyvážených tried, keď sú niektoré triedy zastúpené výrazne viac ako iné (SCIKIT-LEARN 2024).

6.2.7 Matica zámen (Confusion Matrix)

Matica zámen je tabuľka, ktorá umožňuje vizualizovať výkonnosť klasifikačného algoritmu. Je obzvlášť užitočná pri hodnotení výkonnosti klasifikačných modelov viacerých tried. Pre každú triedu matica zobrazuje, koľko príkladov bolo správne klasifikovaných (pravdivé pozitívne a pravdivé negatívne výsledky pre každú triedu), ako aj počet chýb rôznych typov (falošne pozitívne a falošne negatívne výsledky) (SCIKIT-LEARN 2024).

Štruktúra matice zámen pre klasifikáciu viacerých tried:

- Riadky matice zvyčajne predstavujú pravdivé triedy.
- Stĺpce matice zvyčajne predstavujú predpovedané triedy.
- Na diagonále matice sú hodnoty označujúce počet správnych predpovedí pre každú triedu.
- Mimo diagonály sú hodnoty, ktoré udávajú, ako často bola namiesto jednej triedy predpovedaná iná trieda.

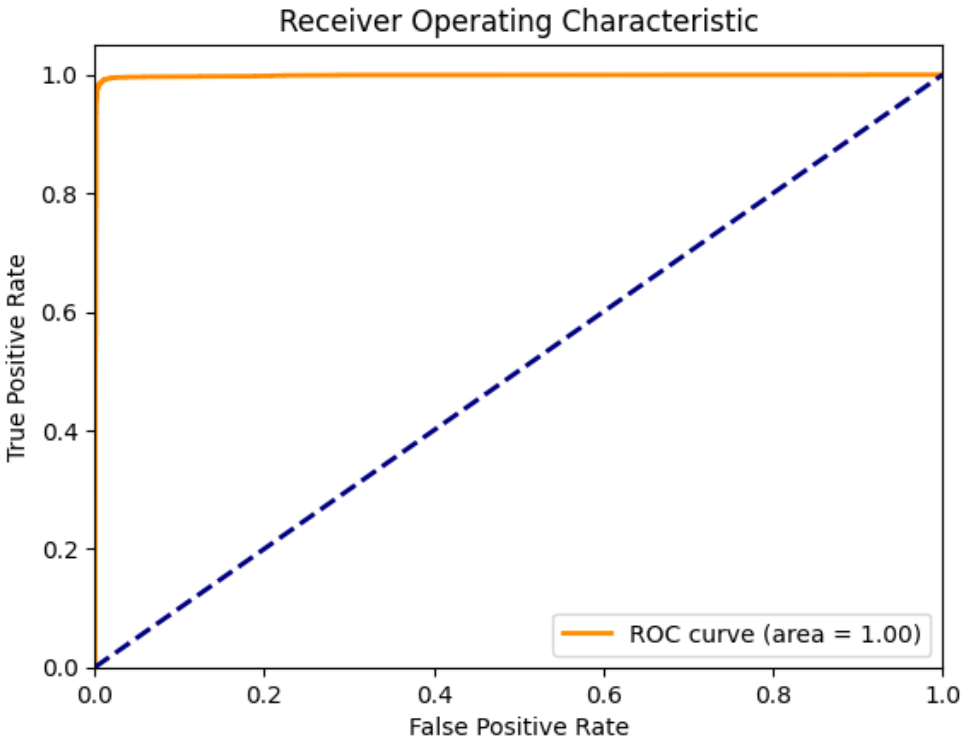
6.3 Hodnotenie modelov

Nižšie budú uvedené výsledky hodnotenia modelov, ktoré sme získali po prejdení všetkých vyššie uvedených krokov, o ktorých sa v tejto práci diskutujeme.

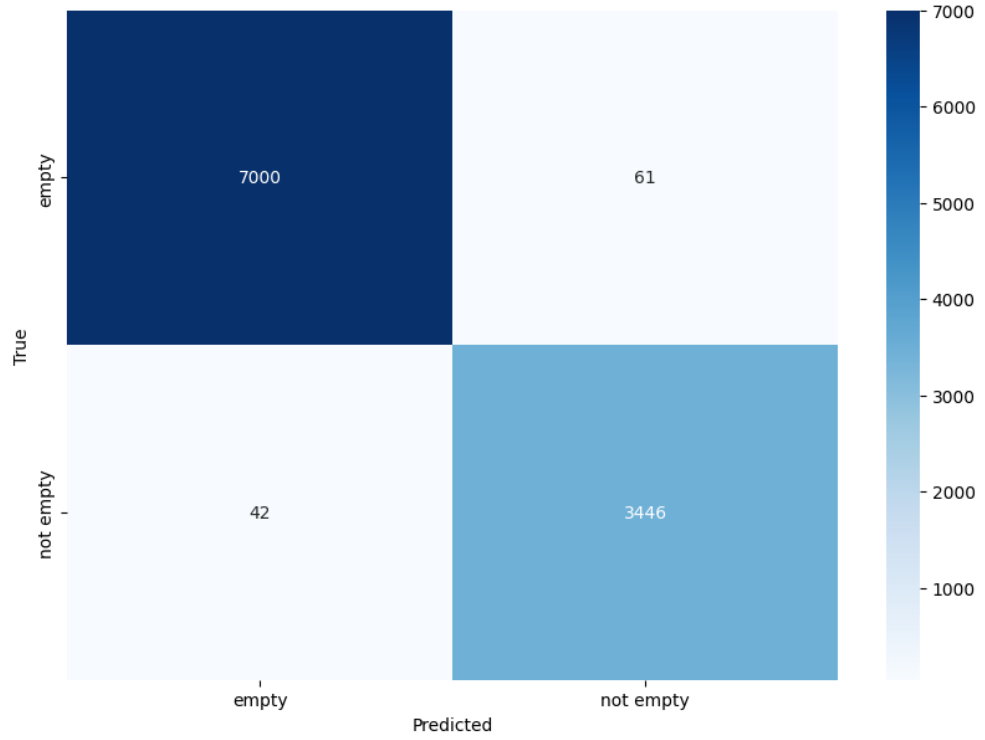
6.3.1 Model 1

Tabuľka 1 Hodnoty prvého modelu

Metrika	Hodnotenie
Accuracy	0.99023604
Precision	0.98260623
Recall	0.9879587
F1 Score	0.98527521



Obr. 12 Krivka ROC



Obr. 13 Confusion Matrix prvého modelu

Z výsledkov hodnotenia výkonnosti prvého modelu, obrázky 12 a 13, možno vyvodiť tieto závery:

Vysokú presnosť (Accuracy): Model vykázal presnosť 99,02 %, čo naznačuje jeho schopnosť správne klasifikovať prevažnú väčšinu prípadov. Táto vysoká úroveň presnosti naznačuje, že model funguje efektívne a chyby predpovede sú veľmi zriedkavé.

Vynikajúca presnosť (Precision): Hodnota presnosti 98,26 % ukazuje, že keď model predpovedá určitú triedu, pravdepodobnosť, že trieda bude predpovedaná správne, je veľmi vysoká. To je rozhodujúce v aplikáciách, kde je dôležité minimalizovať počet falošne pozitívnych výsledkov.

Vysoká miera návratnosti (Recall): Skóre 98,8 % naznačuje, že model dokáže efektívne identifikovať relevantné prípady. Model len zriedkavo vynechá pravdivé prípady, čo je dôležité najmä v situáciách, keď vynechanie pravdivých výsledkov môže mať vážne dôsledky.

Vynikajúce skóre F1: Harmonický priemer presnosti a úplnosti 98,53 % zdôrazňuje, že model vyvažuje presnosť a úplnosť a poskytuje celkovú spoľahlivosť v celom rade prípadov použitia.

Ideálna krivka ROC (AUC = 1): Plocha pod krivkou ROC je 1, čo je ideálny výsledok a znamená, že model má vynikajúcu schopnosť rozlišovať medzi triedami bez chýb. Model dokonale oddeľuje triedy bez ich miešania, čo naznačuje jeho vysokú rozlišovaciu schopnosť.

Na základe analýzy prezentovaných metrík prvý model vykazuje vynikajúci výkon a spoľahlivosť pri klasifikácii. Má vysokú schopnosť správne identifikovať a klasifikovať rôzne triedy, pričom minimalizuje falošne pozitívne aj falošne negatívne výsledky. Vďaka týmto vlastnostiam je vhodný na úlohy, kde chyby môžu viesť k vážnym následkom, čo zdôrazňuje jeho potenciál na použitie v aplikáciách kritických z hľadiska kvality.

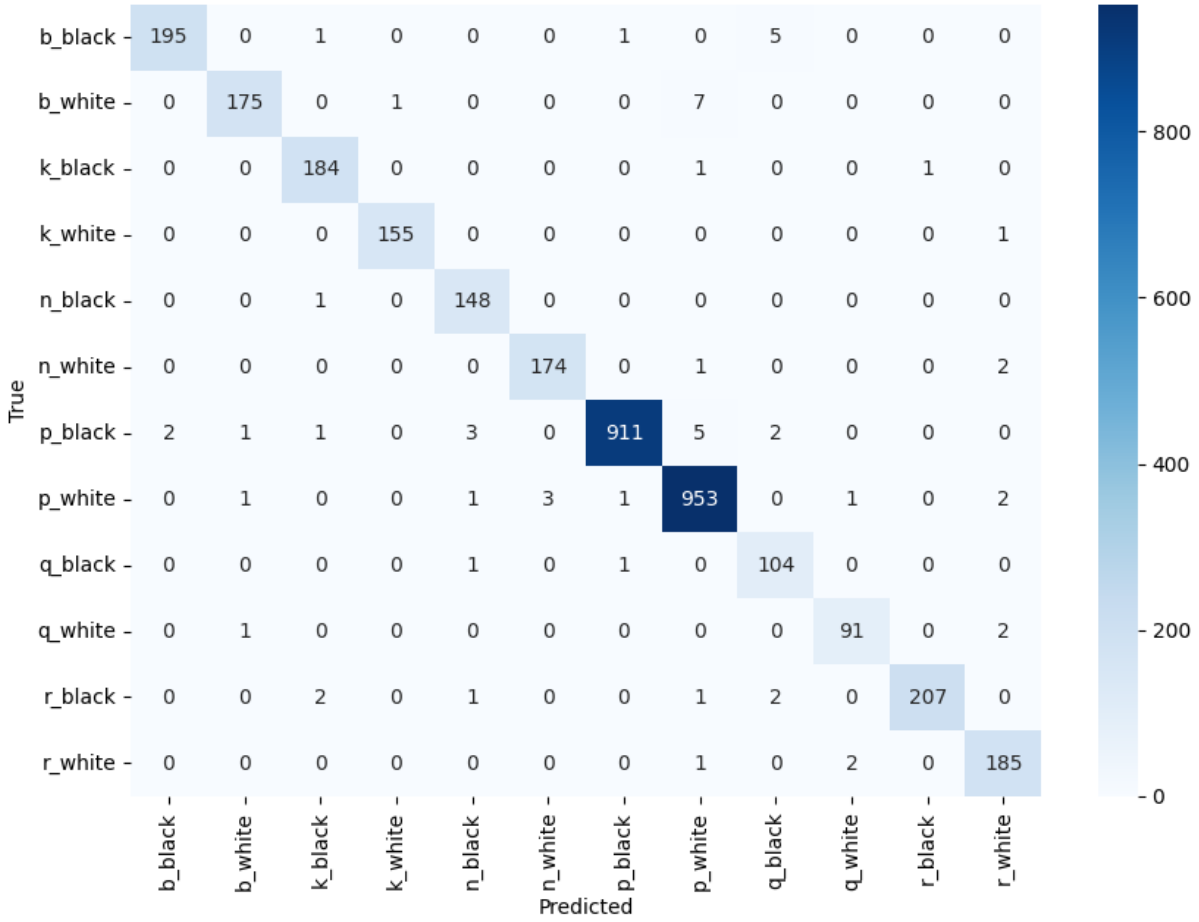
6.3.2 Model 2

Tabuľka 2 Hodnoty druhého modelu

Metrika	Hodnotenie
Categorical Accuracy	0.99937645
Precision	0.99790672
Recall	0.99940103
F1 Score	0.99865326

Tabuľka 3 Hodnoty druhého modelu pre rôzne triedy

Šachová figúrka	precision	recall	f1-score
b_black	0.99	0.97	0.98
b_white	0.98	0.96	0.97
k_black	0.97	0.99	0.98
k_white	0.99	0.99	0.99
n_black	0.96	0.99	0.98
n_white	0.99	0.98	0.98
p_black	1.0	0.99	0.99
p_white	0.98	0.99	0.99
q_black	0.95	0.98	0.95
q_white	0.97	0.97	0.97
r_black	1.0	0.97	0.98
r_white	0.96	0.98	0,97



Obr. 14 Confusion Matrix druhého modelu

Na základe predložených výsledkov, obrázok 14, možno vyvodiť nasledujúce závery o výkonnosti druhého modelu:

Vynikajúca celková presnosť (kategorická presnosť): Dosiahnutá hodnota 99,94 % naznačuje, že model dobre klasifikuje všetkých 12 tried šachových figúrok. To zdôrazňuje jeho vysokú schopnosť presne identifikovať každú triedu, čo je nevyhnutné pre aplikácie, kde je presné rozlišovanie medzi triedami rozhodujúce.

Vysoká miera presnosti a úplnosti tried: Priemerné hodnoty pre Precision 99,79 % a Recall 99,94 % potvrdzujú, že model účinne minimalizuje chybné identifikácie a vynechania. To je obzvlášť cenné v scenároch, kde chyby môžu viesť k vážnym následkom.

Vynikajúce priemerné skóre F1: Skóre F1 99,87 % ukazuje, že model úspešne kombinuje presnosť a úplnosť a poskytuje spoľahlivé výsledky aj v náročných prostrediach rozpoznávania.

Výsledky špecifické pre jednotlivé tvary: Vysoké hodnoty presnosti a úplnosti pre každý tvar dokazujú, že model nie je zaujatý a rovnako dobre rozpoznáva všetky typy tvarov vrátane tých s perfektným skóre 100 %, ako sú p_black a r_black.

Rovnomernosť výsledkov: Rovnomerne vysoké skóre vo všetkých triedach naznačuje, že model funguje konzistentne a spoľahlivo v rôznych podmienkach.

Na základe analýzy výsledkov možno konštatovať, že druhý model má vysokú mieru spoľahlivosti a presnosti pri klasifikácii šachových figúr vo viacerých triedach. Jeho schopnosť minimalizovať chyby a poskytovať vysoký výkon v rôznych metrikách ho predurčuje na použitie v aplikáciách, kde sa vyžaduje bezchybné rozpoznávanie. Tieto vlastnosti z neho robia ideálnu voľbu pre systémy, kde sú presnosť a spoľahlivosť rozpoznávania kritické.

6.4 Praktický test

Testovanie modelu je kľúčovou fázou overovania účinnosti natrénovaných neurónových sietí. V rámci tohto bodu sa testuje výkonnosť modelov na praktických úlohách rozpoznávania šachových figúrok a overovania šachových ťahov. Testovanie zahŕňa niekoľko zložiek:

Predpovedanie triedy obrazu: funkcia predict_image_class načíta a predspracuje obrázok a potom pomocou modelu určí triedu objektu na obrázku. Výsledky predpovede sa porovnávajú so skutočnými údajmi.

Porovnávanie pravdy: Používa sa na kontrolu, či sa predpovedané údaje zhodujú so skutočnými hodnotami uloženými v príslušných súboroch .json. To umožňuje posúdiť presnosť a spoľahlivosť modelov v reálnych podmienkach.

Overovanie šachových ťahov: Funkcia is_valid_chess_move kontroluje logickú a fyzickú správnosť predpovedaných ťahov na šachovnici pomocou modelov na určenie prítomnosti a typu figúr v daných pozíciách.

Proces testovania

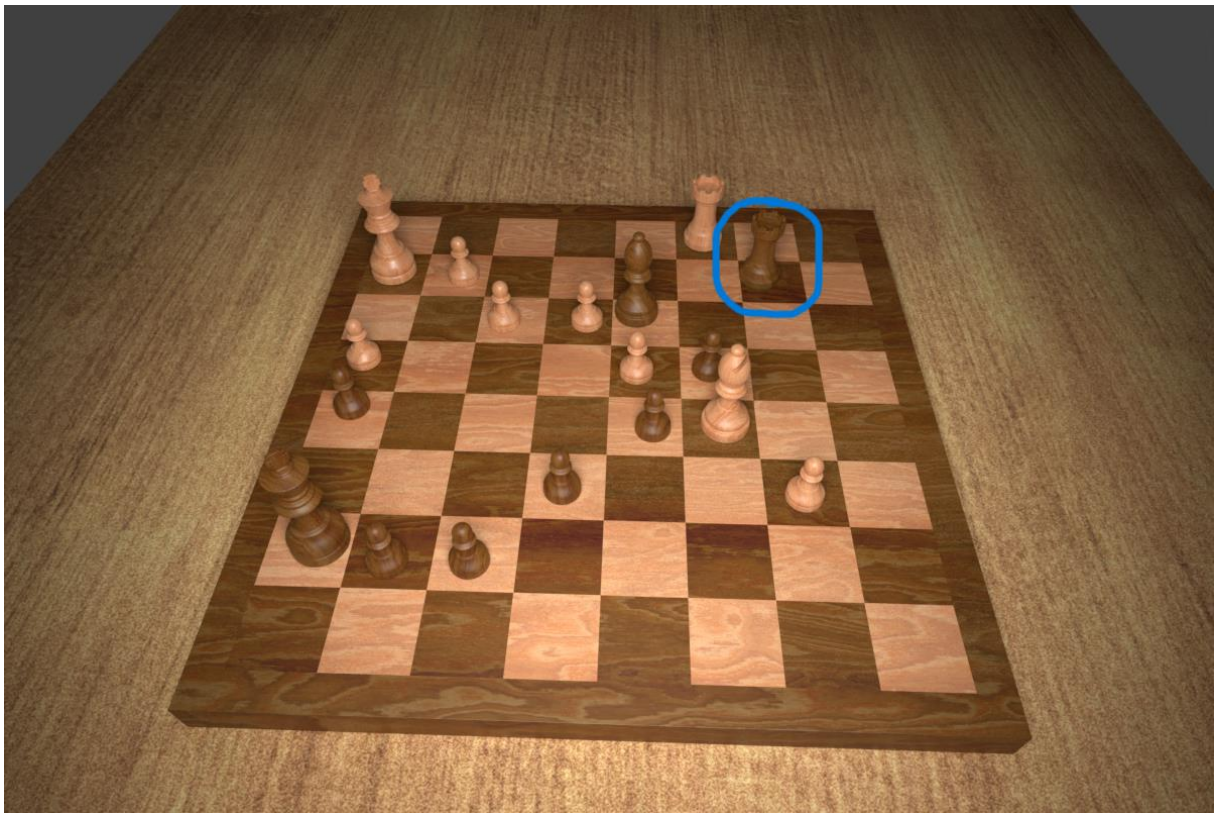
Testovanie sa začína načítaním a predbežným spracovaním údajov pomocou funkcií vyvinutých v predchádzajúcich krokoch. Potom sa na základe získaných údajov vykoná séria testov:

Kontrola prítomnosti figúrok: určenie, či sa v počiatočnej a konečnej pozícii nachádzajú figúrky. To zahŕňa načítanie príslušných obrázkov a ich klasifikáciu pomocou prvého modelu.

Typ figúrky: Ak je na počiatočnej pozícii prítomná figúrka, určí sa jej typ pomocou druhého modelu. To umožňuje nastaviť možné ťahy pre túto figúrku.

Overenie ťahu: Vyhodnotí, či je ťah platný, pričom zohľadní pravidlá pohybu figúrok a prítomnosť iných figúrok na ceste. To zahŕňa kontrolu cesty medzi počiatočným a koncovým bodom na prítomnosť prekážok.

Výsledky testu: Výstupné informácie o tom, či bol ťah povolený, a ak nie, z akého dôvodu.



Obr. 15 Obrázok, na ktorom sa vykonajú testy

Na ukážku si vezmime jeden z obrázkov, na ktorom boli vykonané testy. Teraz vidíme dosku v obrátenom stave, t. j. z strany čiernych. Vykonajme testy na figúrke veže, ktorá stojí v bode b2.

6.4.1 Test 1

Skúsme vykonať ťah b2 - b5

Reakcia programu - {"valid": True, "move_type": "move", "captured": None}

Táto figúrka môže urobiť tento ťah za predpokladu, že je to ťah čiernych. Nedošlo k zničeniu súperových figúrok

6.4.2 Test 2

Skúsme vykonať ťah b2 – b6

Reakcia programu - {"valid": True, "move_type": "capture", "captured": "p_white"}

Táto figúrka môže urobiť tento ťah za predpokladu, že teraz je to ťah čierneho. Zničená figúrka: biely pešiak

6.4.3 Test 3

Skúsme vykonať ťah b2 – b7

Reakcia programu - {'valid': False, 'reason': 'Path is not clear'}

Tento ťah je nemožný, pretože veža nemôže preskočiť pešiaka, ktorý jej stojí v ceste.

6.4.4 Test 4

Skúsme vykonať ťah b2 – h2

Reakcia programu - {"valid": False, "reason": "Cannot capture own piece"}

Chyba nastala, pretože vo štvorci h2 sa nachádza naša figúrka

6.5 Analýzy chýb

Po testovaní môžeme konštatovať, že oba modely fungujú perfektne, spĺňajú naše požiadavky a plnia naše úlohy.

Výsledok však, žiaľ, stále nie je stopercentný. V skutočnosti bude s najväčšou pravdepodobnosťou väčšina chýb v druhej fáze spôsobená tým, že sme v prvej fáze urobili chybu, a to takú, že prvý model označil pole ako vyplnené, ale v skutočnosti tam nič nie je.

V budúcnosti sa plánujeme tejto chyby zbaviť a ešte viac zvýšiť presnosť modelov. Urobíme to tak, že získame obrázky, na ktorých prvý model urobil chybu, a pridáme tieto obrázky do súboru údajov na trénovanie druhého modelu. Toto riešenie výrazne neovplyvní štruktúru druhého modelu, ale poskytne mu možnosť pochopiť, že pole je prázdne a teoreticky bude mať druhý model šancu opraviť chybu prvého.

Údaje, ktoré sa pridajú do druhého súboru údajov, nebudú vybrané náhodne, musia to byť údaje, na ktorých prvý model urobil chybu. Ak porovnáme 2 typy obrázkov s označením - prázdny. Napríklad jeden obrázok je úplne prázdny, nie je na ňom vôbec nič, len prázdna tabuľa. Na takomto type obrázkov má 1 model 100 % presnosť, je to príliš jednoduchá úloha a nemá zmysel ukazovať tieto obrázky druhému modelu. A vezmime si napríklad druhý typ prázdnych obrázkov, ten najťažší. Sú to tie obrázky, kde je presne tá bunka, ktorú potrebujeme, prázdna, ale okolo nej stojí veľa susedných kusov a na obrázku vidíme 5-7 kusov iných kusov, ktoré čiastočne zakrývajú toto prázdne pole, práve na takýchto obrázkoch sa prvý model najčastejšie mýlil a práve tieto obrázky pridáme do súboru údajov druhého modelu.

6.6 Porovnanie s konkurentmi.

Chessboard and chess piece recognition with the support of neural networks (Czyzewski et al. 2020).

Článok autorov Czyzewski, M.A., Laskowski, A. a Wasik, S., publikovaný v roku 2020 v časopise Foundations of Computing and Decision Sciences, sa venuje rozpoznávaniu šachovnice a šachových figúr pomocou neurónových sietí. Práca analyzuje použitie rôznych architektúr hlbokého učenia na automatické zisťovanie polohy šachových figúrok na šachovnici a identifikáciu typov figúrok. Autori článku sa zamerali na návrh a optimalizáciu algoritmov strojového videnia, ktoré dokážu efektívne spracovať obrázky šachovnice na presné

rozpoznávanie a klasifikáciu šachových figúrok. Výsledky tejto štúdie ukázali výrazné zlepšenie presnosti rozpoznávania pomocou konvolučných neurónových sietí a navrhli spôsoby ďalšieho zlepšenia systému prostredníctvom zlepšenia predspracovania údajov a ladenia parametrov siete.

V tejto práci boli vykonané testy na 10 rôznych fotografiách šachovníc, pričom celková presnosť klasifikácie figúr bola 94 %. Čas spracovania nie je známy.

Model optimization for chess pieces classification (Vilà Sánchez 2023).

Diplomová práca, ktorú Vila Sánchez vypracoval na Universite Politecnica de Catalunya v roku 2023, sa zaoberá optimalizáciou modelov na klasifikáciu šachových figúrok. Cieľom práce je skúmať a zlepšovať algoritmy strojového učenia špeciálne prispôsobené na presnú identifikáciu typov šachových figúrok. V práci sa analyzujú rôzne prístupy k optimalizácii architektúr neurónových sietí vrátane ladenia hyperparametrov a použitia techník redukcie nadmerného prispôsobenia. Hlavným cieľom práce je vyvinúť efektívnejší a rýchlejší klasifikačný systém, ktorý možno integrovať do šachových analytických nástrojov alebo reálnych herných aplikácií.

Ako sa nám však zdá, výsledky získané v tejto práci nie sú použiteľné v reálnych aplikáciách, autori tejto práce sa snažili optimalizovať čas vykonávania, pričom zabudli na presnosť.

Tu je to, čo o tom hovorí sám autor:

Pozrieme sa na konkrétnejšie výsledky, aby sme v predpovediach hľadali určité vzory a našli nedostatky modelov. Na tento účel získame predpovede figúr a presnosť šachovnice v hre, aby sme zistili, či modely majú problémy s predpovedaním niektorých figúrok alebo niektorých častí šachovnice. Prvým pozorovaním je, že veľký počet prázdnych políčok je predpovedaných tak, že majú figúrku. Rozpoznávame rovnakú chybnú klasifikáciu ako výsledky validácie: veľký počet prázdnych políčok je klasifikovaný ako pešiaci a ďalší vysoký počet čiernych pešiakov je klasifikovaný ako bieli pešiaci. Prázdne políčka sú ktoré sú predpovedané najhoršie. Aby sme získali väčší prehľad, vypočítali sme aj priemernú presnosť modelov na predpovedanie jednotlivých figúrky. Vidíme, že prázdne políčka nemajú najhoršiu presnosť dielov. Keďže však prázdnych políčok je minimálne 32, ak máme presnosť 63 % pri predpovedaní prázdnych štvorčekov, globálna presnosť klesne. Figúrky, ktoré modely majú najväčšie problémy s predpovedaním, sú králi: v troch modeloch nie je predpovedaný biely kráľ správne ani raz a pokiaľ ide o čierneho kráľa, tri zo štyroch modelov majú predpovede nižšie ako 1%.

Najlepšia presnosť: ResNet50 50,79 %

Čas pre celú šachovnicu: 8.17 sekundy

LiveChess2FEN: A framework for classifying chess pieces based on CNNs (Quintana et al. 2020).

V článku Quintanu, Garcíu a Matíasa, ktorý bol predložený v roku 2020 a uverejnený ako predbežný výtlačok na arXiv, sa vyvíja rámec LiveChess2FEN založený na konvolučných neurónových sieťach (CNN) na klasifikáciu šachových figúr. Výskum navrhuje metodiku, ktorá konvertuje vizuálne údaje zo šachovnice do formátu FEN (Forsyth-Edwards Notation), ktorý opisuje aktuálny stav hry. Dosahuje sa to trénovaním CNN na rozpoznávanie a klasifikáciu šachových figúrok v obrazoch v reálnom čase, čo umožňuje automatizáciu sledovania ťahov a podporuje digitálne prehrávanie partií. V štúdii sa zdôrazňuje presnosť a efektívnosť navrhovaného systému, vďaka čomu predstavuje významný príspevok do oblasti automatizácie šachovej technológie.

Najlepšia presnosť: Xception 94 %

Čas pre celú šachovnicu: 5,62 sekundy

Determining chess game state from an image (WÖLFLEIN 2021).

Túto prácu vytvoril Georg Wolflein. Tento človek vytvoril súbor údajov ChessCog, ktorý sme použili v našej práci, takže porovnanie s výsledkami tejto práce bude najspravodlivejšie.

V práci pána Wolfleina bolo najväčším problémom to, že jeho modely niekedy nesprávne klasifikovali prázdne polia a označovali ich ako pešiakov, čomu sa nám podarilo vyhnúť, pretože náš prvý model mal 99 % presnosť na testovacej množine.

Celková presnosť klasifikácie v tejto práci bola 93,86 %.

Presnosť klasifikátora, ktorý určuje typ figúrky, pričom ignoruje prázdne polia, bola 99,7 %, teda rovnaká ako naša.

Naše riešenie.

V tejto bakalárskej práci sme najväčšiu pozornosť zamerali na algoritmus počítačového videnia, ktorý nám bude musieť nájsť políčka šachových figúrok, a potom sme postupne použili 2 modely na určenie typu figúrky. Ak porovnáme naše riešenie s ostatnými tu prezentovanými, aj keď predpokladáme, že v tejto verzii sa niekedy druhý model pomýli, pretože prvý model bol nesprávny, celková presnosť je stále vyššia ako vo všetkých vyššie uvedených prácach.

Presnosť prvého modelu: 99.0%

Presnosť druhého modelu: 99.9%

Konečná presnosť: 0.98901%

Čas: 23.41 sekundy

Bohužiaľ, ak je prvý model nesprávny, druhý model automaticky poskytne nesprávnu odpoveď. Preto sme na výpočet celkovej pravdepodobnosti predikcie presnosti modelu použili vzorec celkovej pravdepodobnosti z teórie pravdepodobnosti.

Výpočet pravdepodobnosti výskytu udalosti B za predpokladu, že je možná až po výskyte udalosti A, sa uskutočnil podľa vzorca celkovej pravdepodobnosti výskytu udalosti B, pričom sa zohľadnila podmienka výskytu udalosti A:

Vzorec 5 "Celková pravdepodobnos"

$$P(B) = P(A) \times P(B|A)$$

Zdroj: wikipedia, Probability theory

P(A) - presnosť prvého modelu

P(B) - presnosť druhého modelu

P(B|A) - celková presnosť dvoch modelov za predpokladu, že výsledok druhého modelu závisí od prvého.

Tento vzorec je vhodný, pretože udalosť B môže nastať len vtedy, ak nastala udalosť A. Skutočná pravdepodobnosť udalosti B sa teda rovná pravdepodobnosti, že najprv nastane udalosť A a potom udalosť B. Ide o súčin pravdepodobností dvoch závislých udalostí, pričom udalosť B závisí od udalosti A.

ZÁVER

V priebehu práce sa riešil problém vývoja a trénovania modelov strojového učenia na rozpoznávanie šachových figúr na obrázkoch šachovnice. Boli vytvorené dva modely založené na predtrénovaných architektúrach VGG16 a ResNet101, ktoré preukázali vysokú presnosť a schopnosť zovšeobecnenia na nových údajoch.

Ako vidno z porovnania, najbližším konkurentom je autor súboru údajov ChessCog, 94% oproti našim 98,9% (WÖLFLEIN 2021), všetky ostatné riešenia majú oveľa horšiu presnosť a vyžadujú si zlepšenie.

V priebehu spracovania fotografií bol vytvorený vlastný algoritmus počítačového videnia na spracovanie fotografií šachovnice a ďalšie rozdelenie tejto fotografie na malé časti a výsledok je prezentovaný v štýle knižnice na spracovanie súboru údajov ChessCog. Funkcie, ktoré vytvoril pán Wolflein (2021), boli síce použité ako základ, ale konečný vzhľad je vlastným návrhom.

Práca bola organizovaná v niekoľkých kľúčových krokoch: predbežné spracovanie a príprava údajov, vytvorenie a trénovanie modelov, testovanie a analýza chýb. Osobitná pozornosť sa venovala rozšíreniu údajov a optimalizácii bootstrappingu s cieľom zlepšiť trénovanie modelov a ich výkonnosť v podmienkach reálneho času.

Model založený na VGG16 preukázal pôsobivú presnosť viac ako 98 % s vysokými hodnotami metrík, ako sú presnosť a odvolanie. Model založený na sieti ResNet101 dosiahol ešte lepšie výsledky a priblížil sa k dokonalej klasifikácii s presnosťou približne 99 %. Tieto výsledky zdôrazňujú účinnosť zvolených architektúr a metód trénovania.

V procese sa vyskytli určité ťažkosti súvisiace s počiatočným spracovaním údajov a výberom vhodných parametrov modelu. Starostlivým ladením a optimalizáciou sa však podarilo tieto prekážky prekonať a dosiahnuť dobré výsledky.

Obzvlášť úspešné bolo rozhodnutie použiť techniku jemného doladenia po predtrénovaní modelov. Výrazne sa tým zlepšila ich schopnosť rozpoznávať šachové figúrky na rôznych šachovniciach a v rôznych svetelných podmienkach.

Práca dosiahla svoje ciele a preukázala vysokú účinnosť vyvinutých metód a prístupov pri riešení problému rozpoznávania šachových figúrok. Výsledky štúdie sa dajú využiť na vytvorenie systémov, ktoré automatizujú proces analýzy šachových partií a tréningu šachistov.

V budúcnosti plánujeme rozšíriť funkčnosť tejto práce a odstrániť niektoré drobné problémy. Napríklad plánujeme dotrénovať druhý model na základe fotografií, na ktorých prvý model urobil chybu. Tým sa ešte viac zvýši presnosť predpovedí a minimalizuje pravdepodobnosť výskytu chýb.

Táto práca tiež potvrdzuje význam aplikácie hlbokého učenia a počítačového videnia pri riešení zložitých problémov spracovania obrazu.

ZOZNAM POUŽITEJ LITERATÚRY

1. BA, J. a KINGMA, D. P. 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.
2. BENGIO, Y., LIPSON, H., CLUNE, J. a YOSINSKI, J. 2014. How transferable are features in deep neural networks? Advances in Neural Information Processing Systems, 27.
3. CHOLLET, Francois. 2017. Deep Learning with Python. Manning Publications. ISBN 978-1617294433.
4. Czyzewski, M.A., Laskowski, A. and Wasik, S., 2020. Chessboard and chess piece recognition with the support of neural networks. Foundations of Computing and Decision Sciences, 45(4), pp.257-280.
5. HE, K., REN, S., ZHANG, X. a SUN, J. 2016. Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
6. KERAS. 2022. Transfer learning and fine-tuning. [online]. Dostupné z: https://keras.io/guides/transfer_learning/ [Accessed 28 April 2024].
7. NIKOLENKO, Sergei I., KADURIN, Artur a ARKHANGELSKAYA, Elena. 2018. Deep Learning. Saint Petersburg: Peter. ISBN 978-5-4461-0705-0.
8. MCCULLOCH, Warren S. a Walter PITS. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115–133.
9. Quintana, D.M., García, A.A.D.B. and Matías, M.P., 2020. LiveChess2FEN: A framework for classifying chess pieces based on CNNs. arXiv preprint arXiv:2012.06858.
10. SCIKIT-LEARN. 2024. Model evaluation: quantifying the quality of predictions. [online]. Dostupné z: https://scikit-learn.org/stable/modules/model_evaluation.html [Accessed 1 May 2024].
11. SIMONYAN, K. a ZISSERMAN, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556.
12. SMITH, L. N. a TOPIN, N. 2017. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications.

13. STEPIK. 2022. Úvod do dátovej vedy a strojového učenia. [online]. Dostupné z: https://stepik.org/course/4852/syllabus [Accessed 1 May 2024].
14. Vilà Sánchez, D., 2023. Model optimization for chess pieces classification (Bachelor's thesis, Universitat Politècnica de Catalunya).
15. WÖLFLEIN, Georg. 2021. Determining Chess Game State from an Image. Journal of Imaging, [online] 2021. Dostupné z: https://www.mdpi.com/2313-433X/7/6/94.

PRÍLOHY

Príloha č. 1 Súbory a priečinky

V prílohe nájdete 3 zložky s fotografiami dva súbory .keras a 4 súbory s python kódom.
Zoznam priečinkov s fotografiami:

- app_data. V tomto priečinku nájdete 3 obrázky šachovnice a 3 súbory json popisujúce pozíciu na šachovnici. Tieto fotografie sa použijú na testovanie celého programu.
- app_data_split. V tomto priečinku sú rovnaké 3 obrázky už rozdelené do 64 buniek a súbory json pre každý obrázok.
- app_data_test. V tomto priečinku je fotografia číslo 1406 rozdelená do formátu, ktorý vyžaduje funkcia image_dataset_from_directory na vytvorenie objektu tf.DataSet, ktorý bude potrebný, ak bude chcieť niekto model priamo otestovať.

Zoznam python súborov:

- corner_and_sides_detection. Modul s funkciami na hľadanie rohov šachovnice, hľadanie štvorcov a delenie fotografie.
- creating_and_configuring_neural_networks. Modul s funkciami na vytváranie a trénovanie alebo sťahovanie hotových modelov. Na konci tohto súboru nájdete aj komentáre s príkladmi testov.
- preprocessing_data. Modul s funkciami na vytváranie súborov údajov
- __init__. Python súbor, ktorý je potrebný na to, aby sa všetky vyššie uvedené súbory považovali za balík.

Zoznam python súborov:

- model1.keras. Trénovaný model pripravený na načítanie a používanie.
- model2.keras. Trénovaný model pripravený na načítanie a používanie.

Príloha č. 2 Návod na používanie kódu

Ako príloha k tejto práci prichádza archív so python súbormi, zložkami s fotografiami a dva súbory s modelmi keras. model1.keras, model2.keras

Na úspešné spustenie programu je potrebné mať nainštalované: Python vo verzii 3.11 a vyššej, knižnice: cv2, sklearn, recap,tensorflow, shutil, PIL a niektoré ďalšie, úplný zoznam knižníc je uvedený v súboroch pri importovaní.

Python súbory sú v štýle knižnice, každá funkcia má riadok s dokumentáciou a popisom, preto ich tu nebudeme opisovať.

Na rozdelenie fotografie na 64 menších častí je potrebné použiť súbor corner_and_sides_detection.py, posledná funkcia opísaná v tomto súbore (create_variables_for_png_files) po vyvolaní spustí všetky potrebné procesy a kompletne spracuje fotografiu do podoby, ktorú potrebujeme na testy

Ak chcete vytvoriť vlastný súbor údajov z fotografií a potom na ňom trénovať modely, musíte použiť súbor preprocessing_data.py. Opisuje funkcie, ktoré prevedú naše údaje do formátu potrebného na vytvorenie súboru údajov, vytvoria všetky potrebné adresáre priečinkov a naplnia ich

Na vytváranie modelov a ich testovanie používame súbor creating_and_configuring_neural_networks.py. Tento súbor obsahuje funkcie, ktoré môžeme použiť na vytvorenie a trénovanie našich modelov opäť za predpokladu, že máme k dispozícii sadu údajov, na ktorých sa budú trénovať, a tiež sa na konci tohto súboru nachádza niekoľko testov, ktoré nám pomôžu používať hotové modely.
Najprv musíme načítať oba modely:

```
model1 = load_keras_model("first_model.keras")
model2 = load_keras_model("second_model.keras")
```

Test 1/2 - použitie funkcie predict_image_class(). Táto funkcia predpovedá pre obrázok, čo sa na ňom nachádza. Ak hovoríme o modeli 1, potom 0 je prázdny, 1 je niečo. Ak hovoríme o modeli 2, funkcia vráti číslo triedy. To možno použiť ako index pre zoznam class_names a získať textovú odpoveď.
Test 3 je použitie funkcie compare_predictions_with_ground_truth(). Táto funkcia berie obrázok celej šachovnice a vracia súbor údajov, v ktorom porovnáva predpoklad modelu a skutočnú odpoveď. Počas tejto funkcie sa tiež meria čas, aby sme zistili, koľko času strávime spracovaním celej dosky.Test 4 je funkcia is_valid_chess_move(), ktorú sme použili pri praktických testoch v časti 6.4 - praktický test