Sakkfigurák osztályozása

Image classification "in-the-wild"

Neurosünök
Oroszki Marietta, CZYYIF
Wiederschitz Diána, E0YKJT

2024.12

Bevezetés

Feladatleírás

A választott feladatkörünk az Image classification "in-the-wild" nevet viseli, mely során 3 különböző sakkfiguráról, a bástyáról, a huszárról és a futóról készítettünk összesen 150 db saját készítésű képet, hogy egy olyan osztályozót fejlesszünk, amely érdemben fel tudja ismerni a részletekre való rátanulással, hogy egy adott képen melyik sakkbábú látható.

Adathalmaz

Mivel a feladat része volt egy saját adathalmaz összeállítása, minden figuratípushoz 50 darab fotó készült más-más szögekből és megvilágítással, a világos és sötét bábuk felhasználásával egyaránt. Mindezt pedig összesen 5 különféle sakkészlet segítségével (üveg/márvány/fa), valamint 4 egyértelműen eltérő kör- nyezeti háttérrel. A hátterek, illetve a világos-sötét figurák megoszlása az adathalmazban nagyjából egyenlő. A képek két különböző készülékkel lettek előállítva, mindegyik négyzetes, azonban különböző felbontásúak. A képek véletlenszerű sorba rendezését követően utólag elkészítettük a címkéket tartalmazó két oszlopos táblázatot is, melyben a képek neveihez rendeltünk egy számot 0 és 2 között. (0: bástya, 1: huszár, 2: futó) Az adatok elérhetőek a github repoban lévő drive linken.

Kapcsolódó munkák

A [1] cikk az egyik, melyből igyekeztünk mi is inspirációt gyűjteni, egy módszert mutat be, amely konvolúciós neurális hálózatokat alkalmaz a sakktáblán lévő bábuk azonosítására és osztályozására. A cikk egy valamivel komplexebb feladatra keresi a választ, mint mi tettük, de bizonyos elemei számunkra is hasznosak voltak.

A sakktábla képeit feldolgozva a rendszer azonosítja a táblán lévő mezőket. A CNN-ek segítségével az egyes mezőkről pedig először eldönti, hogy üresek-e, majd ha nem üresek, megpróbálja kitalálni, hogy melyik bábu áll rajtuk. A modell tanításához kiterjedt adatkészleteket használtak, köztük szintetikusan generált adatokat is, hogy ellensúlyozzák a valós, címkézett képadatok hiányát. A cikkben vázolt módszer pontosan osztályozta a sakktábla mezőit, ez a megközelítés pedig hasznos lehet sakkos alkalmazások, például valós idejű játék- és edzésrendszerek számára is.

A feladat megoldása során számos más, az interneten fellelhető releváns munkát alapul vettünk, a hozzájuk tartozó linkeket a dokumentum alján fel- tüntettük, de segítségül vettünk néhány korábbi, gyakorlatokon rendelkezésünkre bocsátott notebookokat is.

Adathalmaz előkészítése

A ChessDataset osztály a sakkfigurák képeinek és címkéinek kezelésére szolgál a PyTorch Dataset osztályának kiterjesztéseként. Az osztály betölti a képeket egy megadott mappából az Excel fájlban tárolt címkék alapján RGB formátumban, méretüket pedig egységesíti $512x512$-es formára.

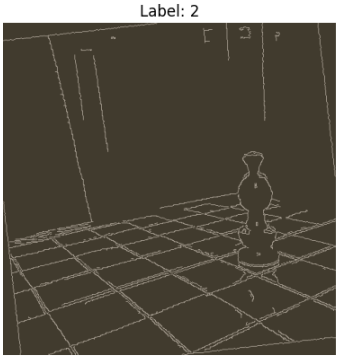
A `create_data_loaders` függvény a sakkadatok előkészítésére és betöltésére szolgál három adathalmazba: tanuló, validációs és teszthalmaz 70-15-15% arány- ban. Az tanítóhalmaz keverve van, hogy a tanulás diverz legyen. Az adatok transzformációjához egy egyedi átalakítási láncot használ, amely magába foglal véletlen- szerű átméretezést, forgatást, vízszintes tükrözést és éldetektálást a `ToEdgeDetection` osztály segítségével.

Mivel csak kis méretű adathalmaz áll rendelkezésünkre kritikus fontosságú volt, hogy a lehető legtöbb információt nyerjük ki a képekből. A figurák felismerése során a legrelevánsabb információ a sakkbábuk alakja, a színek például nem igazán bírnak megkülönböztető jelleggel, ezen információ kinyerése érdeké-

ben pedig számos módszert kipróbáltunk. A feldolgozási folyamat során például kísérletet tettünk a képek két színre való konvertálására, hogy egyszerűsítsük az adatokat. Azonban ez a megközelítés túl sok vizuális részlet elvesztésével járt, ami hátrányosan befolyásolta a modellezés pontosságát, így elvetettük ezt a módszert.

Ennél valamivel célravezetőbb módszernek bizonyult a képeken lévő objektumok kontúrvonalainak kinyerése. A `ToEdgeDetection` a képeket szürkeárnyalatossá alakítja, majd a Canny algoritmussal detektálja az éleket, és az eredményt PyTorch tensor formátumban adja vissza.

A kontúrfelismerés során a threshold értékek határozzák meg, hogy a képen milyen intenzitáskülönbségeket tekintünk élnek. A lower threshold az alacsonyabb határ, amely felett a gyengébb éleket még figyelembe vesszük, míg a higher threshold az erősebb élek kiválasztását segíti. A két érték közötti tartományban lévő pixelek csak akkor tekinthetők élnek, ha összefüggnek egy erősebb éllel, így a küszöbértékek megválasztása kritikus fontosságú volt az információkinyerés szempontjából. Számos kombinációt kipróbálva, és az ered- ményeket kiértékelve a $(60, 120)$-as beállítás egy megfelelő konfigurációnak bizonyult, így ezzel a beállítással dolgoztunk a későbbiek során is. A módszerrel a következő kép mintájára transzformáltuk a fotóinkat:



1. ábra. Példa

Modellezés

Modell összeállítása során elsősorban az órán szerzett tudásra és tapasztalatokra szerettünk volna építkezni, így az egyik gyakorlat alkalmával használt kódot

keztdük el tovább fejleszteni. A végső modellben először több féle konvolúciós blokkot használtunk, amelyek a következőkből tevődtek össze:

- Konvolúciós réteg (33-as szűrőkkel), amely a bemeneti képek jellemzőit vonja ki.
- Batch normalizáció a tanulás stabilizálására.
- ReLU aktivációs függvény, amely nemlineáris transzformációt biztosít.
- Max pooling réteg (22-es ablak), amely csökkenti a térbeli méreteket, miközben megőrzi a legfontosabb jellemzőket.

Utána adaptív átlag pooling-ot használtunk: AdaptiveAvgPool2d réteg fix méretű (44) térbeli reprezentációt állít elő, függetlenül a bemeneti kép méretétől. A modell utolsó részében használunk egy Flatten réteget, melynek segítségével a Dense réteg számára értelmezhető állapotba transzformáljuk a konvolúciós rétegek kimenetét. Egy teljesen összekötött réteg (linear layer) átalakítja a jellemzőket egy 512 dimenziós térbe. Végül ReLU aktiváció és Dropout (50%-os arány) javítja a generalizációs képességet. Egy másik lineáris réteg végzi az osztályozást a num_classes kimeneti osztály számára.

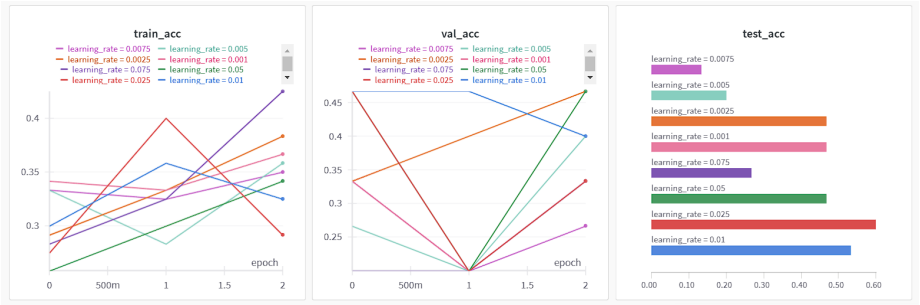
Eredmények

A fent vázolt modell mellett számos másikkal is kísérleteztünk, azonban eleinte, a legegyszerűbb megoldásokkal sajnos mindegyik hasonlóan alacsony pontosságot tudott csak elérni, ez olyan 1/3 körüli értékeket jelentett a teszthalmazon.

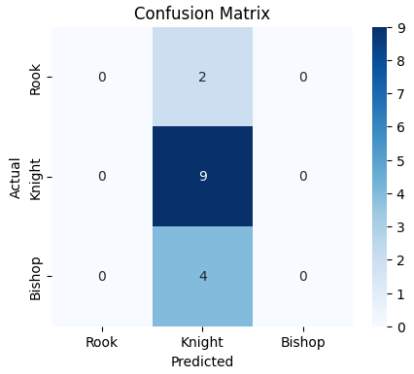
A hiperparaméterek optimalizációja során több paramétert is megváltoztattunk. Például az epocs-ok számánál egyértelműen a kevesebb volt a legcélravezetőbb, a batch méretét a szerény adatmennyiség miatt szinten kisebb értékekre állítottuk be. A tanulóráta beállítása során alapossabb monitorozást is végeztünk a wandb segítségével, ennek eredményeit 2. ábra foglalja össze, itt az epocs-ok száma 3-ra, a batch mérete 10-re lett rögzítve.

Az ábra tartalmazza az általunk elért legmagasabb pontosságot is a teszthalmazon, amely 60%. Látható, hogy ez két esetben is sikerült elérni, mi a szürke színnel jelölt, 0.0025-ös tanulórátájú modellhez készítettük el a konfúziós mátrixot, amely a 3. ábrán látható.

Látható, hogy a modell a lovag figurát nagy arányban el tudta találni, a többi helyen viszont vétett néhány hibát.



2. ábra. Elért pontosságok



3. ábra. Elért pontosságok

Fejlesztési lehetőségek

A jelenlegi modell eredményeit vizsgálva számos továbbfejlesztési lehetőséget azonosítottunk, amelyekkel a teljesítmény mindenképpen javítható lenne. Elsődlegesen a rendelkezésre álló adathalmaz méretének növelése lenne kritikus fontosságú, mivel a jelenlegi korlátozott számú képadat miatt a modell csak nagyon nehezen tudta általánosítani a különböző sakkfigurák tulajdonságait. Egyértelmű probléma volt az is, hogy a képek elkészítésekor néhol a háttér, a pozíció, vagy a fényviszonyok nem mindig voltak megfelelőek a részletek megtanulásához, bármennyire is próbáltunk figyelni rá.

A modellarchitektúra szintjén további fejlesztések közé tartozhatna egy komplexebb neurális hálózat alkalmazása, amely akár már tartalmaz előre tanult vizuális jellemzőket is. Ez nemcsak a tanítási idő csökkentéséhez, hanem a jobb pontossági eredményekhez is hozzájárulhatna kisebb finomhangolásokkal.

Az adatok előfeldolgozása terén további adataugmentációs technikák, mint például intenzívebb színtorzítás, perspektívaváltások és zajok hozzáadása, javíthatnák a modell robusztusságát. A jelenlegi éldetektálás és a beállított küszöbértékek mellett a bonyolultabb textúra- vagy mintafelismerő technikák alkalmazása is megfontolandó lenne, mivel a sakkfigurák azonosításában ezek kulcsszerepet játszhatnak.

Végül további hiperparaméter-optimalizáció, grid search vagy bayesian optimization segíthetne abban, hogy a modell finomhangolásához a lehető legjobb konfigurációkat találjuk meg. Az elért eredmények tükrében ezek a fejlesztések összességében nagy mértékben növelhetik a modell teljesítményét, különösen a teszthalmazokon való predikciók pontosságát.

Hivatkozások

[1] S. Parmar and Rahul. Chess piece classification via convolutional neural networks. In *2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, pages 36–42, Erode, India, 2023.

https://www.kaggle.com/code/evelynartoria/chess-pieces-classification-pytorch-tinyvgg

https://arxiv.org/abs/1512.03385
https://www.sciencedirect.com/science/article/pii/S0031320316303922
https://github.com/andrefakhoury/chess-board-recognition
https://chatgpt.com/
(főleg a kódban található hibák javítására használtuk és az általa adott válaszokat igyekeztünk megérteni)