12/6/2024

Házi Feladat Dokumentáció

Karakterfelismerés

Dukát Nándor, YZV8QV
Gaszner Ádám, BIA823

Tartalomjegyzék

Bevezetés

Ez a dokumentáció egy mélytanulási osztályozó modell implementációját, tanítását, valamint a kapott eredmények értékelését ismerteti. Az osztályozási feladat során a karakterfelismerés problémájával foglalkoztunk, amelyhez kisméretű fekete-fehér képekből álló adathalmazt használtunk. A képekhez karakterazonosítók tartoznak, és a cél az ismeretlen tesztképek karakterének (0-9, a-z, A-Z) felismerése manuális beavatkozás nélkül, mélytanulási osztályozó segítségével. A megoldás során nagy hangsúlyt fektettünk az eredményességre és a modell teljesítményének objektív értékelésére.

A projekt elején egyszerűbb osztályozó algoritmusokkal kezdtünk, például a RandomForestClassifier használatával, amely előtt PCA használatával dimenziócsökkentést végeztünk az adatokon. Ez lehetővé tette a gyorsabb kezdeti kísérletezést, miközben alapvető működési tapasztalatokat szereztünk, azonban volt egy határ, ahonnan már nem tudtuk tovább fejleszteni ezzel a módszerrel a megoldásunkat. Ahogy haladtunk előre a tananyagban, és mélyebb megértést szereztünk a mélytanulásról, áttértünk bonyolultabb modellek, például neurális hálózatok implementálására, hogy tovább javítsuk a modell teljesítményét.

A képek preprocesszálásához a cv2 könyvtárat használtuk, különös tekintettel az élkiemelési technikákra, amelyek segítették a modell számára releváns jellemzők kiemelését. Az adathalmaz feldolgozásakor a képeket az eredeti méretükről 32x32 pixeles méretre csökkentettük, mivel ez kiegyensúlyozta a számítási teljesítményt és a modell hatékonyságát. Megpróbáltuk nagyobb méretben is feldolgozni a képeket, azonban a számítási idő exponenciálisan megnőtt, és nem értünk el jobb eredményeket.

Az osztályozó modellünk mély neurális hálózatként több konvolúciós rétteget alkalmaz, amelyet magas számú epoch mellett tanítottunk, miközben early stopping és dropout mechanizmust használtunk a túlilleszkedés elkerülése érdekében.

A projekt lépései közé tartozott a modell megtervezése és implementálása, a tanulási folyamat dokumentálása, a tesztadathalmazra történő predikciók előállítása. A dokumentáció kitér a tervezési döntésekre, amelyek meghatározták a végső megoldást, és részletezi a résztvevők közötti munkamegosztást.

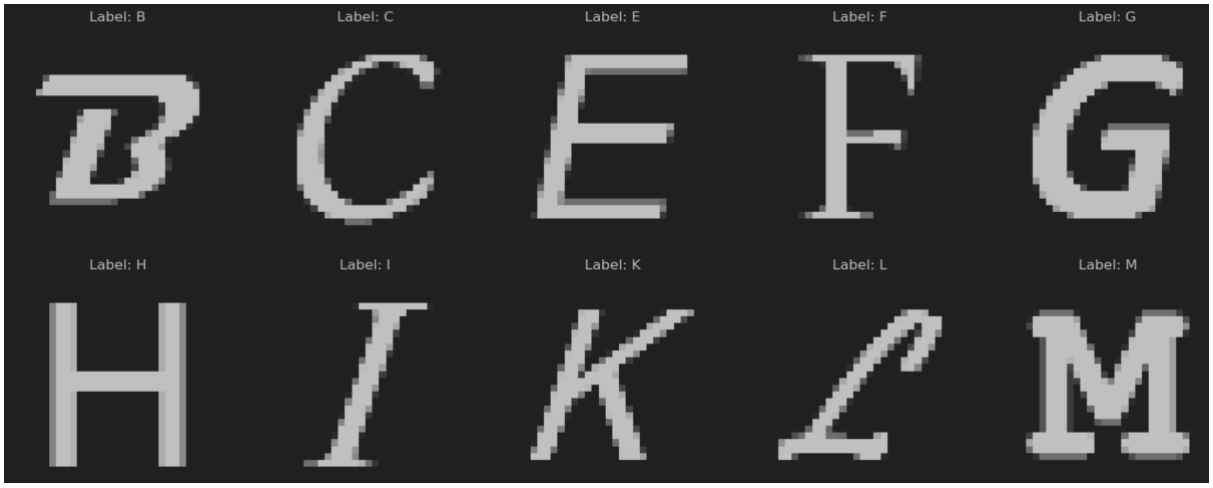
Feladat és adathalmaz ismertetése

Az adathalmaz kisméretű fekete-fehér képekből áll, amelyek az angol ábécé kis- és nagybetűit, valamint a számjegyeket (0-9) reprezentálják. Minden karakterhez egy dedikált mappa tartozik, amelyben 800-900 tanítóelem található. Az összes képet egységesen 32x32 pixel méretre skáláztuk, majd a megfelelő címkékkel láttuk el a további feldolgozás érdekében.

A tanító adatok összeállításakor megfigyeltük, hogy a képek minden karakter esetében széles körű varianciát mutatnak, így biztosítva a modell számára a megfelelő generalizációs képességet. Ez lehetővé tette, hogy a különböző típusú bemeneti karakterek felismerése hatékonyan megtörténjen.

Az adathalmaz kizárólag fekete-fehér színskálát használ, amely jelentősen egyszerűsíti a preprocesszálási folyamatokat. Az adatok előfeldolgozásához a cv2 könyvtárat alkalmaztuk, különös hangsúlyt fektetve arra, hogy a modell a vékonyabb éleket is észlelni tudja, ezért a beolvasásnál Gauss blurt használtunk, ezen kívül implementáltunk Laplace, Canny, illetve Sobel féle éldetekciós módszereket, de ezeket a végső megoldásunkban nem alkalmaztuk. Ez az eljárás kiemelte a képek jellegzetességeit, ami a modell tanulási folyamatának hatékonyságát nagymértékben növelte.

Az adathalmaz megfelelően kiegyensúlyozott, az osztályok címkéinek eloszlása egyenletes, így az osztályozó modell tanítása során nem lépett fel osztálytorzítás. Az adatokat tanulási és tesztelési részhalmazokra osztottuk, biztosítva ezzel a modell fejlesztésének és objektív kiértékelésének alapfeltételeit.



Képek az adathalmazból

Megvalósítás lépései

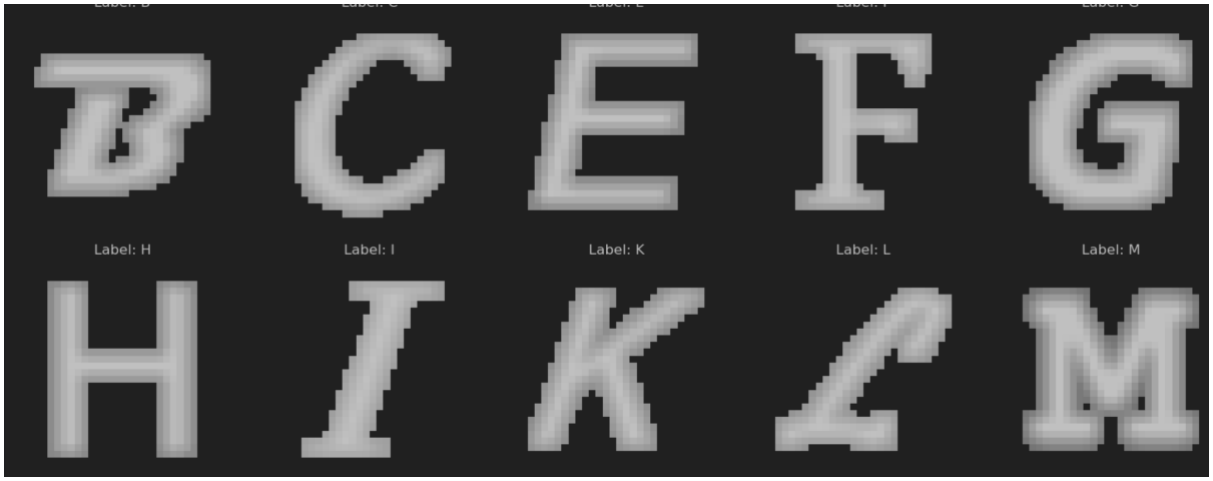
Adatok előkészítése és beolvasása

A képek feldolgozását egy függvény segítségével valósítottuk meg, amely iteratívan beolvasta az adatokat, és előkészítette azokat a modell számára. Az importált os, cv2, és pandas könyvtárakat használtuk az adatfeldolgozás különböző lépéseiben: az os a mappák és fájlok kezeléséhez, a cv2 a képfeldolgozási feladatokhoz, például a méretezéshez és éldetektáláshoz, míg a pandas az adatok strukturált tárolásához és manipulálásához kellett.

Az összes mappát végigjárva, az egyes mappákhoz hozzárendeltük a megfelelő célváltozót, amely kategóriaként jelölte a karaktertípusokat. A mappákon belül iteráltunk a képeken, és minden feldolgozott képet egy közös adathalmazhoz fűztünk hozzá.

A képek méretét 32x32 pixelre csökkentettük, mivel a kísérletek során ez bizonyult a leghatékonyabb kompromisszumnak a számítási teljesítmény és a modell pontossága között. Az előkészítés során az éldetektálás érdekében a Gauss blur módszert alkalmaztuk a cv2 segítségével, amely kiemelte a karakterek éleit és segítette a releváns jellemzők kinyerését. A detektált élek képpontértékeit 8 bites formátumba konvertáltuk, normalizáltuk, és laposított formában tároltuk az adathalmazban, hogy megfeleljenek a modell bemeneti követelményeinek.

Ezt a teljes feldolgozási folyamatot egy paraméterezhető függvényben valósítottuk meg, amely egységes módon kezelte a tanító és teszt adatok előkészítését. A függvény az iteratív beolvasástól kezdve a méretezésen, éldetektáláson és normalizáláson át a végső adatstruktúra előállításáig minden lépést automatikusan elvégzett, jelentősen megkönnyítve a további feldolgozást és elemzést.



Képek a feldolgozás után

Konvolúciós neurális háló

A feladat megvalósításhoz konvolúciós neurális hálót alkalmaztunk, amely a neurális hálók egy típusa, amelyet legtöbbször képi adatok feldolgozására használnak. A CNN-ek alapvető jellemzője, hogy a bemeneti adatok térbeli (vagy időbeli) struktúráját kihasználva képesek automatikusan megtanulni a releváns jellemzőket, ezzel minimálisra csökkentve a manuális jellemzőkivonás szükségességét. A CNN-ek egyik kulcseleme a konvolúciós réteg, amely szűrők (filterek) segítségével képes kiemelni a bemeneti adatok különböző jellemzőit, például a képek éleit, textúráit vagy mintáit. A konvolúciós művelet során a szűrő egy "ablakként" mozog a bemenet fölött, és az adott területen alkalmazott számítások eredményeit egy új, jellemzőket tartalmazó térképen tárolja. Ez a megközelítés rendkívül hatékony, mivel megőrzi a térbeli kapcsolódásokat, miközben csökkenti a feldolgozandó adatok mennyiségét.

A CNN megvalósításához a tensorflow keras könyvtárát használtuk, és egy függvény segítségével hozzuk létre, amelynek megadható a bemenet és a kimenet mérete.

Modell felépítése

```
from tensorflow.keras import layers, models

def build_conv_pool_model(input_size, length):
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_size,
padding="valid"))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(64, (2, 2), activation='relu', padding="valid"))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(128, (2, 2), activation='relu', padding="valid"))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(length, activation='softmax'))

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Konvolúciós blokkok felépítése

- Conv2D: Konvolúciós réteget használunk a bemeneti képek jellemzőinek kinyeréséhez. A rétegek sorban 32, 64, illetve 128 szűrőt használnak, az első rétegben 3x3, a többi rétekben 2x2 méretűek, a relu aktivációs függvény segítségével növeljük a modell nemlinearitását. Az egyre nagyobb szűrőszámok lehetővé teszik, hogy a modell összetettebb mintákat is felismerjen.
    - input_shape: Az első konvolúciós réteg megkapja a bemeneti adatok méretét.
    - padding="valid": Az eredeti kép széleit nem tölti fel nullákkal, így a kimeneti méret csökken.
- BatchNormalization: Normalizálja a konvolúciós réteg kimenetét, gyorsítva a tanulást és csökkentve a túlilleszkedés esélyét.
- MaxPooling2D: Csökkenti a kimeneti mátrix dimenzióját, kiemelve a legfontosabb jellemzőket. A (2, 2) pooling ablak felezi a térbeli méretet.

Flatten és Dense rétegek

- Flatten: A 2D-s kimenetet egy egydimenziós vektorrá alakítja, amely alkalmas a teljesen kapcsolt rétegekhez.
- Dense(256, activation='relu'): Teljesen kapcsolt réteg, amely 256 neuronnal dolgozik. A relu aktiváció a nemlinearitást biztosítja.
- Dropout(0.5): Csökkenti a túlilleszkedés esélyét úgy, hogy minden iterációban véletlenszerűen nullázza a neuronok 50%-át.
- Dense(length, activation='softmax'): A kimeneti réteg a softmax aktivációt használja, amely a bemenetek valószínűség-eloszlását adja vissza az osztályszám szerint.

Modell fordítása

- Az Adam optimizert használtuk, ez gyors és hatékony gradiensalapú tanulást biztosít.
- Veszteség függvényként a sparse categorical crossentropyt használtuk, mivel a címkék integer formában vannak megadva.
- Az accuracy metrika nyomon követi a modell teljesítményét a tanulás és validáció során.

Modell tanítása

A feldolgozott adatokat először a train_test_split függvény segítségével osztottuk fel tanító- és validációs halmazra, amely az sklearn könyvtárból érhető el. A validációs halmaz mérete a teljes adathalmaz 10%-át tette ki. Az elosztás során a stratify paraméter használatával biztosítottuk, hogy az osztályok aránya mindkét halmazban az eredeti adatkészletnek megfelelő legyen. Emellett a random_state paraméter segítségével garantáltuk, hogy az osztás reprodukálható legyen, vagyis ugyanazokat az adathalmazokat kapjuk minden futtatásnál.

A modell tanítását két callback segítségével optimalizáltuk. Az egyik a korai leállás (EarlyStopping), amely megakadályozza a modell túlilleszkedését. Ez a mechanizmus figyeli a validációs halmaz veszteségét (val_loss), és ha az öt egymást követő epoch során nem tapasztal javulást, a tanítási folyamat leáll. A callback ezen felül visszaállítja a modell súlyait arra az állapotra, amikor a legjobb eredményt érte el.

A másik callback a TensorBoard, amely lehetővé teszi az eredmények vizualizációját, például az tanítási és validációs veszteség, valamint az egyéb metrikák alakulásának nyomon követését. Ehhez egy naplófájlt hozunk létre, amely tartalmazza a tanulási folyamat során gyűjtött információkat, így azok a TensorBoard segítségével könnyen elemezhetők. Ezáltal a tanítási folyamat átláthatóbbá és nyomon követhetőbbé válik.

Ezután következett a modell illesztése az alábbi módon:

```
history = model.fit(X_train, y_train,
              epochs=50,
              batch_size=256,
              callbacks=[early_stopping, tensorboard_callback],
              validation_data=(X_val, y_val),
              verbose=1)
```

Eredmények

A projektet a félév elején kezdtük el, kezdetben az akkori tudásunk és a tananyagban addig elsajátított algoritmusok alkalmazásával. Az adatok előfeldolgozása során dimenziócsökkentéshez PCA-t használtunk, majd a redukált dimenziójú adatokon RandomForestClassifier segítségével végeztük az osztályozást. Ezzel a megközelítéssel körülbelül 80%-os pontosságot értünk el.

Ahogy a tananyaggal haladtunk előre, az alkalmazott módszerek és az eredményeink is folyamatosan fejlődtek. A projekt végső verziójában három konvolúciós rétegből álló neurális háló segítségével sikerült elérnünk a legjobb teljesítményt. Ezzel a modellel a validációs halmazon elért pontosságunk körülbelül 92,5% lett.

A modellek teljesítményének értékelése során a teszt halmazt is használtunk. Minden modellmódosítás után külön elemeztük az eredményeket, hogy feltárjuk a modell aktuális gyengeségeit és erősségeit. Ezeket az elemzéseket figyelembe vettük a további optimalizálás során, ami hozzájárult a végső modell teljesítményének növeléséhez.

Próbálkozások a modell javítására

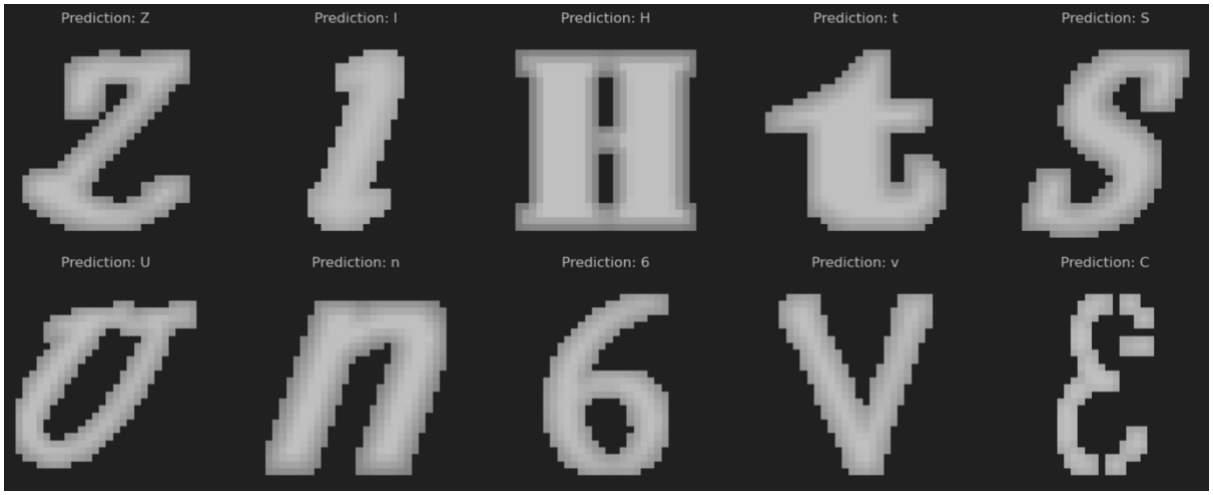
Az adatbeolvasás során az Image Data Generator használátával is próbálkoztunk, hogy javítsuk a modell teljesítményét és általánosítási képességét azáltal, hogy a tanítás során az adatokból nagyobb változatosságot hozzunk létre. Azonban ez nem segített a modellünk teljesítményét, gyakori volt az osztályok felcserélése.

Modellünk egyik fő gyengesége a vékony élekből álló karakterek, különösen a számok detektálása volt, amit többféle megközelítéssel próbáltunk orvosolni. Ennek érdekében három különböző éldetektáló algoritmust teszteltünk: Sobel, Laplace és Canny. Ezeket abban a reményben alkalmaztuk, hogy az éldetektálás segíthet a modell számára jobban felismerni az ilyen típusú karaktereket az előfeldolgozás során.

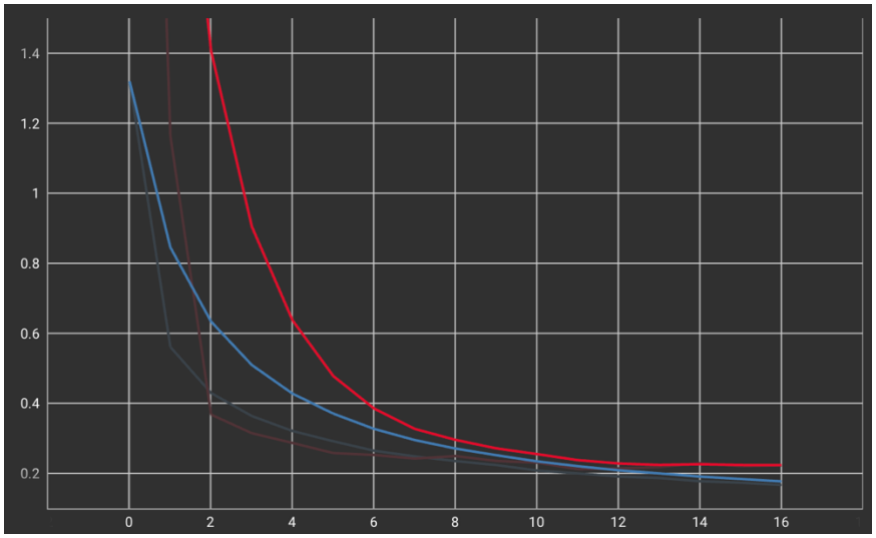
Azonban az éldetektálás bevezetése nem hozott javulást. A tesztek során a teszt halmazon egy vékony vonalú „7”-es karaktert használtunk referenciaként, de az alkalmazott éldetektáló módszerek következetesen több darabra szabdalták a karakter vonalait. Ez a torzítás miatt a modell gyakran inkább „1”-esként azonosította a „7”-eseket az előfeldolgozott képeken.

Ezen problémák hátterében az állt, hogy az éldetektálási algoritmusok túlérzékenyek voltak a vékony vonalakra, és hajlamosak voltak a finomabb részletek feldarabolására. Bár az algoritmusok paramétereit igyekeztünk optimalizálni, ez sem hozott számottevő javulást. Végül arra a következtetésre jutottunk, hogy az éldetektálás önmagában nem megfelelő módszer a vékony élekből álló karakterek felismerésének javítására. Ezért a végső megoldásban a GaussianBlur alkalmaztuk 3-as kernel mérettel, ez bizonyult a leghatékonyabbnak, ekkor a már említett “7”-es referencia karaktert is felismerte a modellünk, és a teszt halmazon is jelentős javulást láttunk.

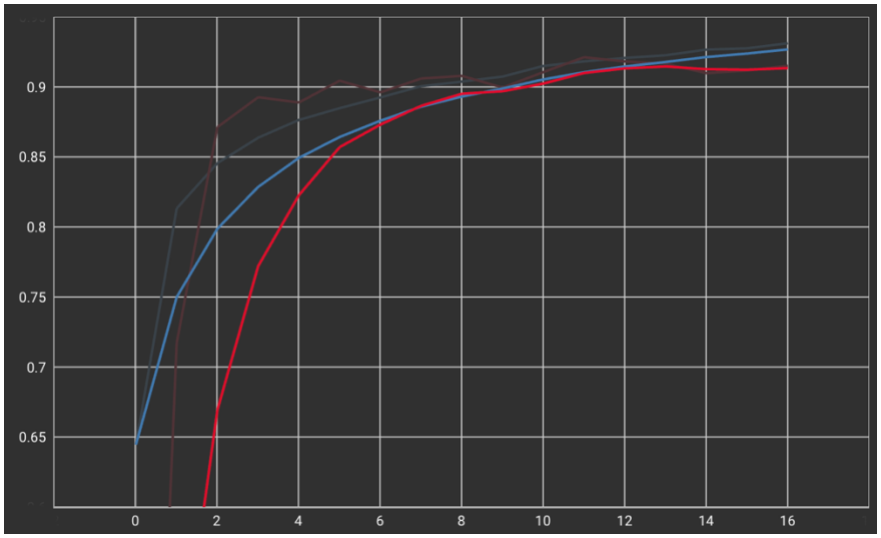
Véletlenszerűen Kiválasztott Képek és Predikciók Vizualizációja



Epochonkénti veszteség alakulása a végső modellen



Epochonkénti pontosság alakulása a végső modellen



A teszthalmazon elért eredmények

A modellünk a teszthalmazon 82,2%-os pontosságot ért el, míg az F1 eredmény 78,1%-os lett. A teszthalmazon elért 10%-kal rosszabb eredmény mögött az állhat, hogy olyan képek is szerepeltek benne, amelyekhez hasonlót a modell korábban nem látott, és ezek felismerése jelentősen nehezebb volt, szinte szabad szemmel is alig felismerhetőek. Mint például az alábbi kép:

Konklúzió

A mélytanulási modellek, különösen a konvolúciós neurális hálók, hatékonyabban alkalmazhatók képi adatok osztályozására, mint a gépi tanulási modellek.

Az adatok előfeldolgozásának minősége és a modell felépítése kulcsfontosságú a jó teljesítmény eléréséhez.

A túlilleszkedés elkerülése érdekében a megfelelő technikák, mint a Dropout és a BatchNormalization, alkalmazása elengedhetetlen.

Fontos a megfelelő EarlyStopping használata, mivel a teszthalmazon elért eredmények után egy magasabb „patience” értékkel is megpróbálkoztunk. Ez jelentős javulást hozott a validációs adathalmazon, ami azt sugallja, hogy valószínűleg a teszt halmazon is jobb eredményeket érhettünk volna el. Ebből levonható a következtetés, hogy a modellünk alul tanult volt.

Kontribúció eloszlása

A munkát közösen kezdtük el még a konvolúciós neurális hálók implementálása előtt, amikor az addig tanult osztályozó algoritmusokat teszteltük. Miután a tananyagban eljutottunk a neurális hálók témaköréhez, folyamatos egyeztetés mellett külön-külön dolgoztunk, amíg el nem értünk egy stabilan működő neurális háló kialakításáig. Ezt követően ismét közös erőfeszítésekkel optimalizáltuk a modellt, így érve el a végső eredményt.

A kommunikáció elsősorban Discordon, időnként személyesen zajlott. A közös munkához a Git és GitHub platformokat használtuk, amelyek mindkettőnk számára hozzáférést biztosítottak a projekthez.

A munkához egyenlő arányban, 50-50%-ban járultunk hozzá, biztosítva a közös felelősségvállalást és eredményességet.

Irodalomjegyzék

- https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
- https://scikit-learn.org/stable/
- https://www.tensorflow.org/guide/keras
- https://opencv.org