12/6/2024

Dukát Nándor, YZV8QV

Gaszner Ádám, BIA823

Házi Feladat Dokumentáció

Karakterfelismerés

Tartalomjegyzék

[Bevezetés 1](#_Toc183602359)

[Feladat és adathalmaz ismertetése 3](#_Toc183602360)

[Megvalósítás lépései 4](#_Toc183602361)

[Adatok előkészítése és beolvasása 4](#_Toc183602362)

[Konvolúciós neurális háló 5](#_Toc183602363)

[Modell felépítése 5](#_Toc183602364)

[Konvolúciós blokkok felépítése 6](#_Toc183602365)

[Flatten és Dense rétegek 6](#_Toc183602366)

[Modell fordítása 6](#_Toc183602367)

[Eredmények 7](#_Toc183602368)

[Konklúzió 8](#_Toc183602369)

[Kontribúció eloszlása 9](#_Toc183602370)

# 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 rérteget alkalmat, 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.

A képen szöveg, Betűtípus, szám, óra látható

Automatikusan generált leírás

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.

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.

A képen szöveg, Betűtípus, szám, óra látható

Automatikusan generált leírás

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.

# Eredmények

# Konklúzió

# Kontribúció eloszlása