Beszámoló szakmai gyakorlatról

Kórád György
ZF440N
korad.gy@gmail.com

Intézmény: Miskolci Egyetem, Matematikai Intézet

Instruktor: Piller Imre
egyetemi tanársegéd

2023

Tartalom

1 Bevezetés

A szakmai gyakorlatot a Miskolci egyetemen töltöttem, Piller Imre tanár úr felügyeletében. Az egyetem különböző karokból és intézetekből áll, és széleskörű tudományos kutatásokat és oktatást kínál többféle területen. Az intézmény története 1949-ig nyúlik vissza, és azóta is folyamatosan fejlődik. Az egyetemnek aktív diákközössége van, és rendszeresen szervez kulturális és sporteseményeket. Továbbá, az egyetem fontos szerepet játszik a régió oktatási és tudományos életében, valamint különböző kutatási projektekben és együttműködésekben vesz részt más intézményekkel és vállalatokkal.

A cél egy olyan program elkészítése volt, ami kézzel írott dokumentumok digitalizálását hajtja végre mesterséges intelligencia segítségével. Az elkészült programrészek python nyelven íródtak jupyter munkafüzetekben. A python a mesterséges intelligencia fejlesztéséhez kiválóan alkalmas, mert könnyen tanulható és rendkívül rugalmas programozási nyelv. Egyszerű és olvasható szintaxisa gyors prototípusok és algoritmusok készítését teszi lehetővé, ami különösen hasznos az MI tervezési folyamatának kezdeti szakaszában. Számos népszerű és erőteljes könyvtár és keretrendszer érhető el Pythonban, mint például a TensorFlow és a PyTorch, amelyek segítik a modellek fejlesztését és kísérletezését. Könnyen integrálható más technológiákkal és platformokkal, lehetővé téve az MI alkalmazásoknak az együttműködést más rendszerekkel. Alkalmas adatfeldolgozásra és vizualizációra is, ami elengedhetetlen a projektek során gyűjtött adatok elemzéséhez és értelmezéséhez. Összességében Python egy erőteljes és sokoldalú eszköz a mesterséges intelligencia területén, amely lehetővé teszi a fejlesztők számára, hogy hatékonyan dolgozzanak projekteken.

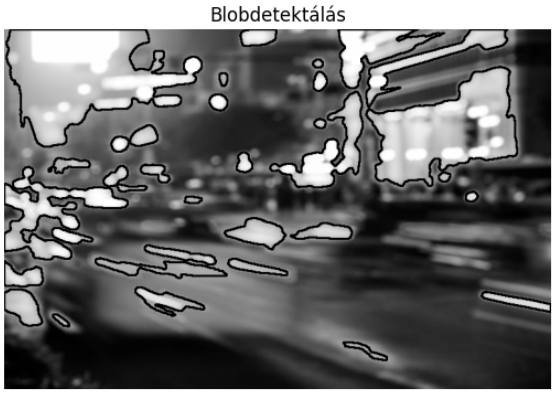
A python kiváló választás a kézírás felismeréshez is. A programozási nyelv rendkívül rugalmas és sokoldalú, ami lehetővé teszi a fejlesztőknek, hogy könnyedén kifejlesszék a kézírásos karakterek vagy szavak felismerését segítő alkalmazásokat. Python egyszerű és olvasható szintaxisa elősegíti a prototípusok gyors létrehozását és a kísérletezést különböző kézírásfelismerési algoritmusokkal. Emellett a nyílt forráskódú könyvtárak és keretrendszerek, például a OpenCV és a Tesseract, segítik a kézírásos szövegek felismerését és azoknak az alkalmazásoknak a fejlesztését, amelyeknek kéziráselemzésre van szükségük. A python közössége aktív és támogatja az ilyen projekteket, így könnyű hozzáférhetőséget és erőforrásokat biztosít a fejlesztőknek.

2 Képfeldolgozás

A szakmai gyakorlat elején képmanipulációs algoritmusokkal foglalkoztam, mivel előzetes képmanipuláció szükséges volt a képek tisztításához és javításához.

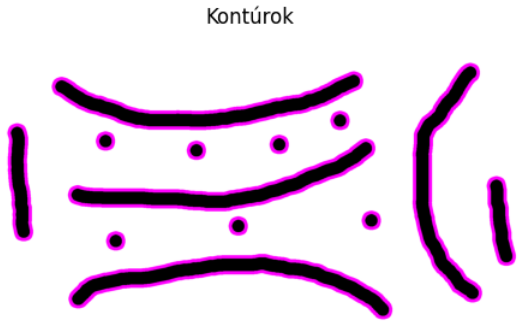
- **Képminőség javítása:** Bemeneti képek gyakran tartalmaznak zajt vagy lehetnek rossz minőségűek. A képmanipulációs technikák segítségével csökkenthetjük a zajt, javíthatjuk a kontrasztot és növelhetjük a kép tisztaságát, ami növeli a kézírás felismerési pontosságot.
- **Képek egységesítése:** A képek lehetnek különböző méretűek és felbontásúak, ami hátráltathatja az automatikus felismerési folyamatot. A képmanipuláció segít abban, hogy az összes képet azonos formátumba hozzuk, ami könnyebbé teszi az algoritmusok számára az azonosítást.
- **Jellemzők kiemelése:** Az előfeldolgozási lépések során kiemelhetünk olyan jellemzőket a képeken, amelyek segítenek azonosítani a kézírásos karaktereket vagy szavakat. Például élesíthetjük a karakterek kontúrjait vagy eltávolíthatjuk a háttér zaját.
- **Hiba javítása:** Ha a képeken hibák vagy torzítások vannak, a képmanipulációs lépésekkel ezeket kijavíthatjuk. Ezáltal növelhetjük a felismerési pontosságot és a rendszer megbízhatóságát.

Összességében a képmanipuláció előkészíti a bemeneti képeket az automatikus kézírás felismerési folyamathoz, javítja a felismerés pontosságát és hatékonyságát, és hozzájárul a projekt sikeréhez.



Ebben a részben a kontúrkövetés nagy kihívást jelentett számomra. A kontúrkeresés egy képfeldolgozási technika, amelyet a gépi látásban alkalmaznak az objektumok vagy alakzatok külső határainak meghatározására. Az éldetektálás olyan pontokat vagy vonalakat talál, ahol a kép intenzitása hirtelen változik, például egy objektum határain. A kontúrkeresési algoritmus ezután ezeket az éleket összekapcsolja, hogy megtalálja az élek által határolt összefüggő területeket. Ezek az összefüggő területek jelentik az objektumok vagy alakzatok külső határait. A kontúrkeresés azért hasznos a gépi látásban, mert lehetővé teszi az objektumok határainak és alakjainak meghatározását a képeken. Ezenkívül segít kinyerni fontos jellemzőket, például az objektumok méretét vagy területét. A kontúrkeresés segíthet az objektumok elkülönítésében a háttértől vagy egymástól, és csökkentheti a kép zaját, ami javítja a gépi látás algoritmusok teljesítményét.

2.1 Kontúrkövetés

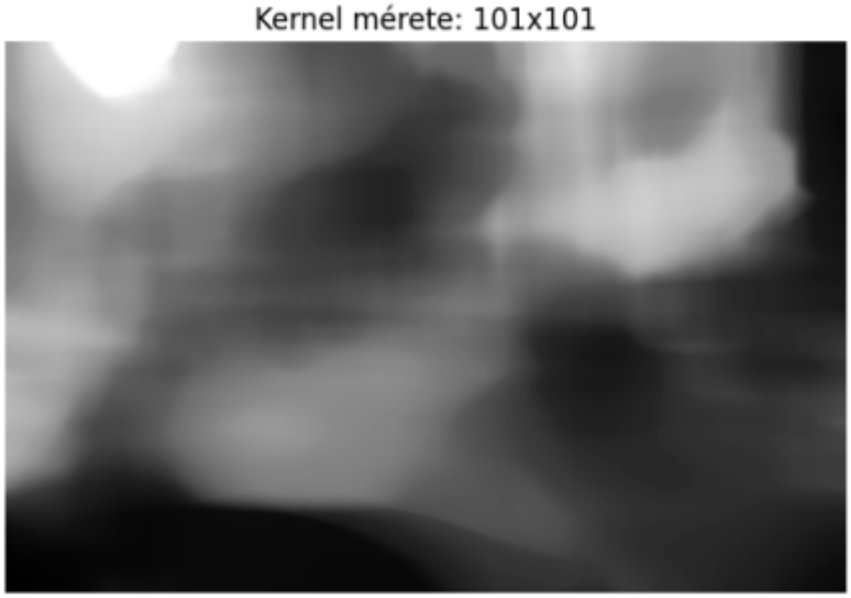


Canny éldetektálás

A fenti példában a kontúrokat canny éldetektáló algoritmussal emeltem ki. A Canny éldetektor egy képfeldolgozási algoritmus, amely az élek megtalálására szolgál a képeken. Működése során először kisimítja a képet, majd kiszűri az éleket a kép gradiensével. Végül kiválasztja a legkiemelkedőbb éleket és összekapcsolja azokat, hogy folytonos vonalakat hozzon létre. Ezt az algoritmust zajérzékenység csökkentésére és élek pontos meghatározására használják a gépi látásban és képfeldolgozásban.

2.2 Mediánszűrő

A zajszűrés mediánszűrővel egy gyakran használt módszer a képfeldolgozásban és a jelanalízisben, amely segít eltávolítani a képeken vagy jeleken lévő zajt vagy zavarokat. A mediánszűrő egy olyan lineáris szűrő, amely a kép vagy jel kis részein egy meghatározott ablakot használ, és az ablakban lévő értékeket rendezve a középső értéket választja ki. A mediánszűrő alkalmazása a zajszűrésre azért hatékony, mert a medián értéke kevésbé érzékeny az extrém értékekre vagy zajokra, mint például az átlagolás vagy a Gauss-szűrés. A mediánszűrő általában jól teljesít olyan képeken vagy jeleken, ahol a zaj vagy a zavar impulzusok vagy hirtelen ugrások formájában jelenik meg.

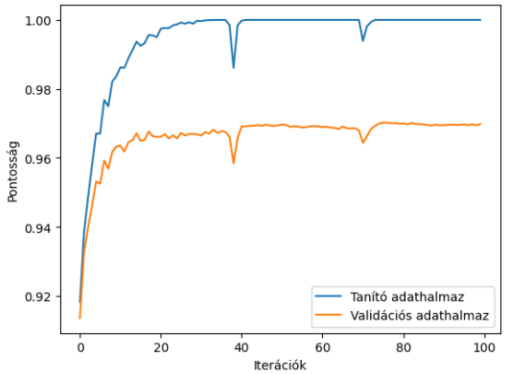


A mediánszűrő egy olyan képfeldolgozási technika, amely a kép zajának eltávolítására szolgál. Ennek a műveletnek a lényege az, hogy egy kis ablakot (ez az ablak a "kernel" vagy "szűrő") mozgat a képen, és minden ablakban található értékeket rendezve kiválasztja a középső értéket (azaz a mediánt). Ez a medián érték lesz a szűrt kép pixelének új értéke.

3 Kézi számjegyek felismerése

A képfeldolgozó algoritmusok után számjegy felismeréssel foglalakoztam. Ehhez az mnist tanító adathalmazt használtam és a scikit-learn könyvtár függvényeit. A scikit-learn könyvtárban megtalálhatók a különböző gépi tanulási feladatokhoz szükséges osztályok és függvények, például osztályozás, regresszió, klaszterezés, dimenziócsökkentés. Emellett könnyen integrálható más Python könyvtárakkal és eszközökkel, így kiváló választás az adatok feldolgozásához és gépi tanulási modellek kifejlesztéséhez.

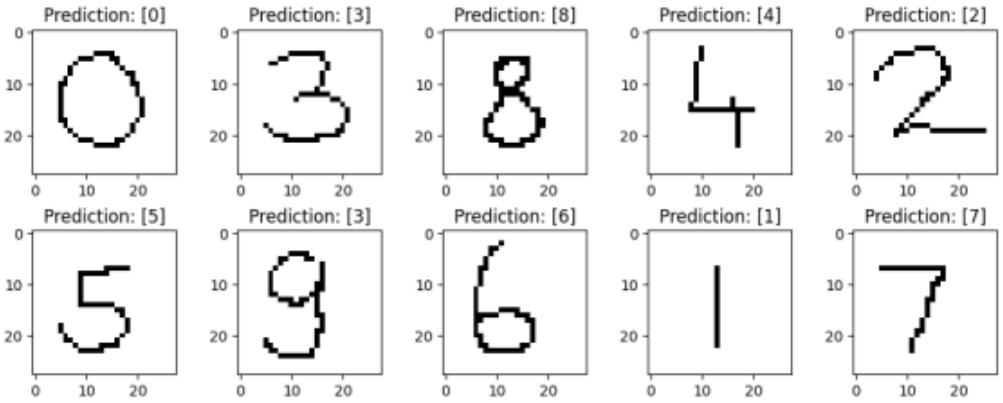
```
# Validációs adathalmaz létrehozása
xtrain, xval, ytrain, yval = train_test_split(xtrain, ytrain, test_size=0.5,␣
↪random_state=42, stratify=ytrain)

# MLP osztály inicializálása
clf = MLPClassifier(solver='adam', activation='relu', hidden_layer_sizes=(64,␣
↪64), max_iter=1)

# Pontosságok tárolása
train_accuracy_list = []
val_accuracy_list = []

# Lépésenkénti tanítás és pontosságok nyomon követése
for i in range(100):
    #print(f'{i + 1} / 100')
    clf.partial_fit(xtrain, ytrain, classes=np.unique(ytrain))

    # Pontosság mérése a tanító adathalmazon
    ytrain_pred = clf.predict(xtrain)
    train_accuracy = accuracy_score(ytrain, ytrain_pred)
    train_accuracy_list.append(train_accuracy)

    # Pontosság mérése a validációs adathalmazon
    yval_pred = clf.predict(xval)
    val_accuracy = accuracy_score(yval, yval_pred)
    val_accuracy_list.append(val_accuracy)

# Pontosságok megjelenítése
plt.plot(train_accuracy_list, label='Tanító adathalmaz')
plt.plot(val_accuracy_list, label='Validációs adathalmaz')
plt.xlabel('Iterációk')
plt.ylabel('Pontosság')
plt.legend()
plt.show()
```



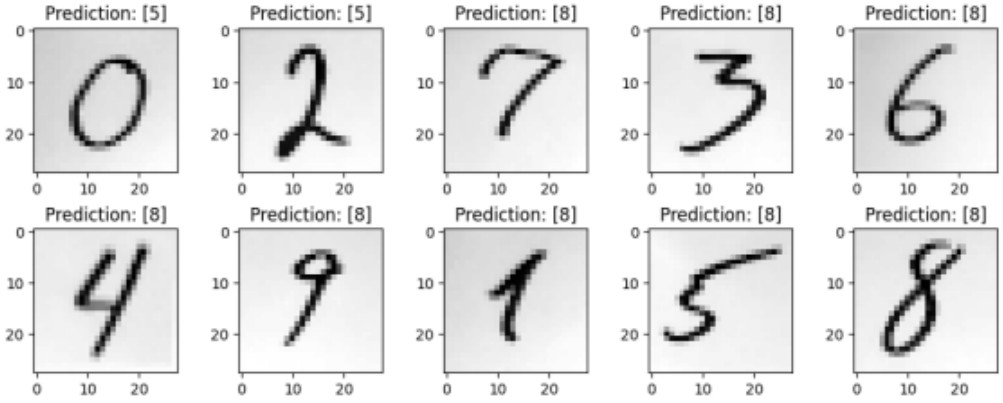
Modell tanítás algoritmusa és pontossága

A fenti példában MLPClassifierrel dolgoztam a teljes adathalmaz felhasználásával. Az MLPClassifier (Multi-Layer Perceptron Classifier) egy olyan osztály a scikit-learn könyvtárban, amely egy mesterséges neurális hálót (MLP) használó osztályozót implementál. Az MLPClassifier lehetővé teszi a gépi tanulási modellek létrehozását és betanítását osztályozási feladatokhoz. A hálózat több rétegből áll, és képes tanulni a bemeneti adatok alapján.

Ebben a feladatrészben kézzel illetve digitálisan írt számjegyekkel folytettam vizsgálatokat. A digitális számjegyek sokkal pontosabb eredményeket adtak, a kézzel írott számjegyeknél szükség volt a fenti képfeldolgozó algoritmusokra, így ezek a vizsgálatok is pontosabb eredményekhez vezettek.



Digitális számjegyek



Kézi számjegyek

Ezek után a kézi számjegyekkel foglalkoztam tovább, viszont ezzel a módszerrel sem kaptam tökéletes eredményeket. Tanár úr javaslatára megpróbáltam a mintaszám csökkentésével és a tanító addathalmaz zajosításával foglalkozni, és az így kapott modellt vizsgálni. A tanító adathalmaz zajosítása az optikai karakterfelismerő (OCR) rendszerek fejlesztésében és tanításában hasznos lehet több okból.

- **Robosztusság növelése:** Ha az OCR rendszer tanító adathalmaza zajjal van ellátva, a modellnek jobban meg kell tanulnia kezelni a valós életből származó zajokat, például rossz minőségű szkenneléseket, elmosódást, zajos háttérrel rendelkező képeket stb. A zaj hozzáadása segíthet abban, hogy az OCR rendszer robosztusabbá váljon és jobb teljesítményt nyújtson valós környezetben.
- **Túltanulás megelőzése:** Ha a tanító adathalmaz kifogástalanul tiszta és zajmentes, az OCR modell hajlamos lehet túltanulásra. Ez azt jelenti, hogy a modell túlságosan illeszkedik a tanító adathalmazhoz, és nehezebben kezeli az olyan variációkat, amelyek nem szerepelnek a tanító adathalmazban. A zaj hozzáadása segíthet ezt a problémát enyhíteni, mivel a modellnek általánosabb mintázatokat kell tanulnia.
- **Korlátozott adatok esetén:** Ha korlátozott mennyiségű tanító adat áll rendelkezésre, a zajosítás segíthet növelni az adathalmaz méretét anélkül, hogy újabb tiszta adatokat kellene begyűjteni. Ezzel többféle variációt és kontextust lehet bevezetni a tanító adathalmazba.
- **Praktikusabb tesztelés:** Ha a rendszer a valóságban zajos képeket dolgoz fel, akkor a tesztelés során is olyan zajos adatokat kell használni, amelyek hasonlóak az élő alkalmazásokban előforduló adatokhoz. A zajosított tanító adathalmazok használata lehetővé teszi, hogy a tesztelés valósághűbb legyen.

Fontos azonban megjegyezni, hogy a zajosítást megfelelően kell beállítani, hogy a zaj ne váljon túlzottá, ami ronthatja a tanító adathalmaz minőségét és a modell teljesítményét. A zaj mennyiségét és típusát gondosan kell kiválasztani, hogy megfeleljen a céljainknak, és segítse a rendszerünk hatékonyabb és robusztusabb kialakítását. Ez a módszer sem adott tökéletes eredményeket, viszont sokat javított a korábbiakhoz képest.

4 Kézírás felismerése

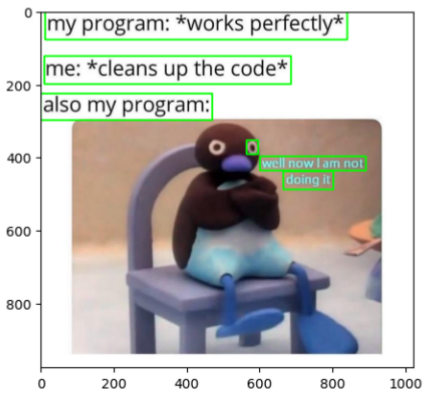
4.1 EasyOCR és Pytesseract

Végül a gyakorlat legnehezebb része következett, a szkennelt szövegek digitalizálása. Itt először az EasyOCR és Pytesseract könyvtárakkal foglalkoztam. Az EasyOCR és a Pytesseract két olyan eszköz, amelyekkel gépi módon lehet szövegeket felismerni képeken vagy szkennelt dokumentumokon.
EasyOCR: Egyszerűen használható Python könyvtár a karakterfelismeréshez, több nyelvet támogat, és előre tanított modellekkel rendelkezik.
Pytesseract: Python interfész a Tesseract OCR motorhoz, ami egy erőteljes nyílt forráskódú karakterfelismerő motor. Mindkét eszközt szövegfelismeréshez használják, de a választás attól függ, hogy milyen egyszerűséget vagy testreszabhatóságot preferál az adott feladatban.
A vizsgálataim alapján a tesseract jobb eredményeket adott kézírás felismerésére, ezért ezzel haladtam tovább. Viszont érdemes megemlíteni az EasyOCR-t is, mivel ezzel nagyon egyszerűen lehet gépelt szöveget kiemelni képekről.

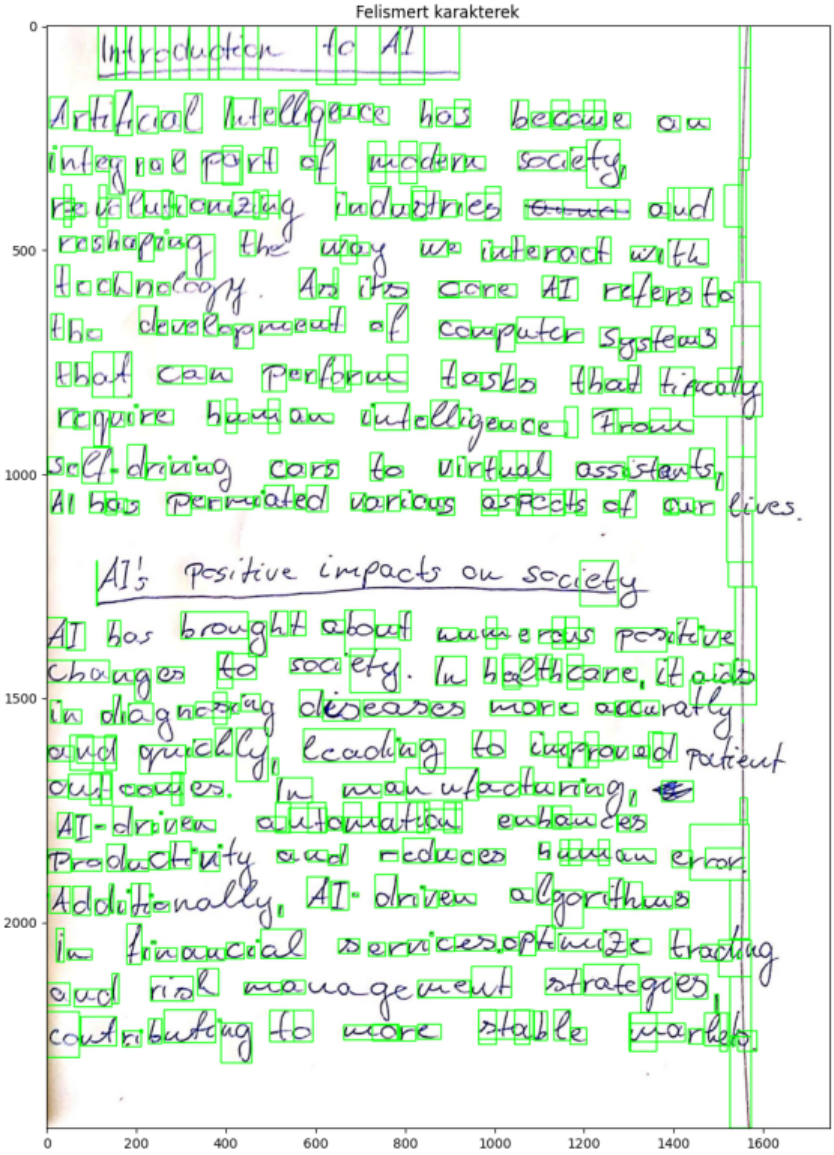
```
img2 = cv2.imread(image2)
spacer = 100
for detection in result2:
    top_left = tuple(detection[0][0])
    bottom_right = tuple(detection[0][2])
    text = detection[1]
    img2 = cv2.rectangle(img2,top_left,bottom_right,(0,255,0),3)
    spacer+=15
    print(text)
```

```
plt.imshow(img2)
plt.show()
```

```
my program: *works perfectly*
me: *cleans up the code*
also my program:
0
well now Iam not
doing it
```



EasyOCR működés közben

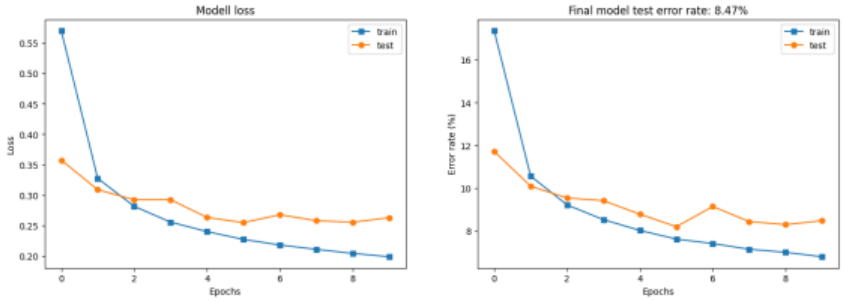


Pytesseract szegmentálása

A Tesseract OCR karakterfelismerő motorja egy olyan szoftver, amely képeken vagy szkennelt dokumentumokon található szövegeket azonosít és kinyer. Ehhez szövegmezők megtalálását, karakterek szegmentálását és felismerését végzi el, majd a végeredményt szöveggé összeállítja. A karakterfelismerést algoritmusok és statisztikai modellek segítik. Sajnos ezzel a módszerrel sem tudtam tökéletesen detektálni minden karaktert, ezért további megoldasokat kerestem. Így találtam rá a PyTorch-ra.

4.2 Torchvision

A torchvision egy python könyvtár, amely része a PyTorch ökoszisztémának, és képfeldolgozási és gépi tanulás feladatokhoz kínál eszközöket és előfeldolgozó függvényeket.
Képfeldolgozás: A torchvision lehetővé teszi képek betöltését, előfeldolgozását és transzformálását, például méretezést, vágást és normalizálást. Ezek a funkciók különösen hasznosak, ha képfeldolgozási feladatokkal foglalkozunk, például képek osztályozásával vagy szegmentálásával.
Datasets és DataLoaders: A torchvision tartalmaz olyan előre definiált adathalmazokat, amelyeket könnyen lehet használni a gépi tanulási modellek tanításához és teszteléséhez. Ezenkívül lehetőséget kínál a saját adathalmazok betöltésére is.
Modellarchitektúrák: A torchvision részei olyan előre tanított mély tanulási modellarchitektúrák, mint a VGG, ResNet és Inception. Ezek a modellek könnyen alkalmazhatók és finomhangolhatók különböző képfelismerési feladatokhoz.
GPU-támogatás: A torchvision támogatja a GPU-kat, így lehetővé teszi a gyors gépi tanulási műveletek végrehajtását, különösen akkor, ha mély tanulási modellarchitektúrákat használunk.



A tanulási folyamat számos lépésből áll. Először inicializáljuk a modellt, a veszteségfüggvényt és az optimalizátort. A tanítás során több epochon keresztül iterálunk, és minden epochban a tanítóadathalmazon tanítjuk a modellt. A veszteség (loss) és a hiba (error) értékeket rögzítjük minden epoch után.

A modell az epochok végén kiértékelődik a tesztadathalmazon, és a tesztelési veszteség és hiba értékeit rögzítjük.

Végül visszaadjuk a képzett modellt, amelyet később felhasználhatunk további feladatokhoz, például osztályozáshoz vagy predikciókhoz.

Torchvision modell pontossága

Torchvision-t használva két modellt készítettem, egy az EMNIST adathalmmazal, ami a MNIST kibővítése írott karakterekkel, illetve a másik az I AM WORDS adathalmaz, amely számos kézzel írt szót tartalmaz. Ezzel a két modellel készült két grafikus felületű program, egy, ami betűket és egy, ami szavakat próbál felismerni, amiket a felhasználó kézzel tud a grafikus felületen beírni. Sajnos időhiány miatt ezek kissé pontatlanok maradtak, de szeretném egyszer tökéletesíteni őket.

5 Összegzés

A szakmai gyakorlat során hatalmas mennyiségű értékes tapasztalatot szereztem a kézírás felismerés területén, és mélyebben megértettem a gépi tanulás és képfeldolgozás kulcsfontosságú elveit. Az első lépéseken keresztül, amikor a képfeldolgozással és zajosítással foglalkoztam, felfedeztem a képek előfeldolgozásának és javításának fontosságát. Megértettem, hogy a képek minőségének és tisztaságának javítása jelentősen javíthatja a kézírás felismerési rendszerek pontosságát. A gépi tanulási modellekkel való munka során, különösen az MLPClassifierrel, mélyebben megismertem a gépi tanulás alapelveit és az osztályozók működését. A tanító adathalmaz zajosításának fontosságát is felfedeztem, és tapasztaltam, hogyan segíthet a modelleknek megbirkózni a valós környezetből származó zajokkal. Köszönettel tartozom az egyetemnek és Piller Imre tanár úrnak a lehetőségért, hogy részt vehettem ezen a projekten, és bővíthettem ismereteimet ezen a területen.