Képi Karakterfelismerés Konvolúciós Hálók használatával

Adatelemzés Mélytanulási Módszerekkel (BME VITMAC15) - nagyházi feladat

Szerzők: Cardinael Jan Maerten, Király Bálint
Dátum: 2024. Dec. 4.

Abstract

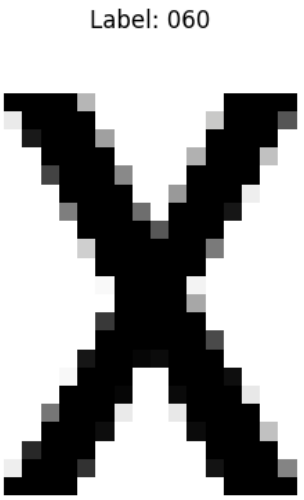
A képi karakterfelismerés a mesterséges intelligencia és deep learning egyik fontos alkalmazási területe, amelyben a cél egy olyan modell fejlesztése, amely képes a bemeneti képeken szereplő karaktereket magas pontossággal azonosítani. Ebben a feladatban konvolúciós neurális hálók (CNN) alapú architektúrát alkalmazunk, amelyet egy karakterképekből álló, előre feldolgozott adathalmazon tanítunk. Az architektúra paramétereinek finomhangolását a validation accuracy függvényében végezzük, különös figyelmet fordítva az overfitting minimalizálására.

Assignment Description

Az adathalmaz kis méretű, fekete-fehér képeket tartalmaz, amelyek egyértelmű karakterazonosítóval vannak ellátva. A cél egy olyan mélytanulási osztályozó modell kifejlesztése, amely automatikusan, manuális beavatkozás nélkül képes az ismeretlen képeken szereplő karaktereket azonosítani (az emberi szemmel történő segítség alkalmazása nem megengedett). Az osztályok halmaza a következő: 0–9 számjegyek, a–z kisbetűk és A–Z nagybetűk.

Data Preparation

Az adatelőkészítés során az OpenCV könyvtár segítségével olvastuk be és dolgoztuk fel a képeket. Az eredeti képek mérete 128x128 pixel volt, amelyet 28x28 pixelre redukáltunk, hogy csökkentsük a futási időt és memóriaigényt. A képeket szürkeárnyalatos formátumban olvastuk be, majd normalizáltuk a pixelek értékeit a [0,1] intervallumba.



Mivel a CNN bemeneteknek egy extra csatornadimenzióra van szüksége, a képeket átalakítottuk egy 4D-s tensorba, ahol a shape (példányszám, 28, 28, 1). A címkék hozzárendelésére az sklearn könyvtár LabelEncoder osztályát használtuk, amely a képek mappanevei alapján számszerű címkéket generált. Az előfeldolgozás végén a címkéket és képeket numpy tömbökké alakítottuk, amelyek már készen álltak a modell betanítására.

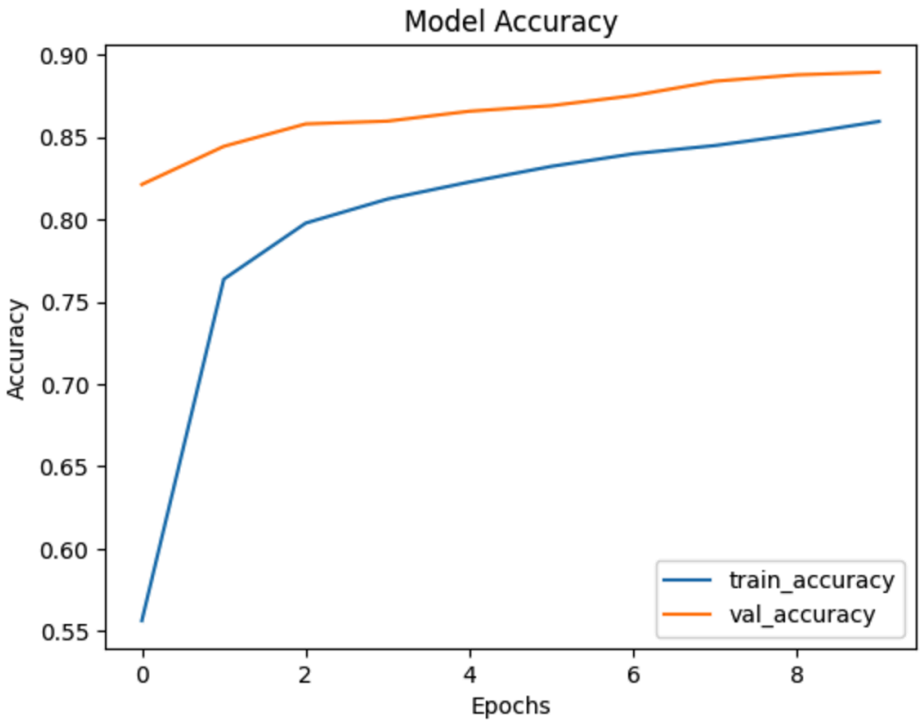
Baseline model

Első lépésként felépítettünk egy Baseline modellt, amelyet a kiselőadás keretében bemutattunk. Ennél a modellnél még nem alkalmaztunk keresztvalidációt, és a hiperparamétereket sem optimalizáltuk; ezek értékeit a tanultak alapján, önkényesen határoztuk meg.

A CNN architektúra a következőképpen épült fel:
Két konvolúciós réteg szerepel a hálózatban, az elsőben 32, a másodikban 64 szűrővel, mindkettő 3x3-as kernelmérettel. Az aktivációs függvény mindkét rétegben ReLU. A rétegek után MaxPooling rétegeket alkalmaztunk 2x2-es mérettel a dimenziók csökkentésére. Az adatok lapítása (Flatten réteg) után egy 128 neuronból álló Dense réteg következik, amely szintén ReLU aktivációt használ. Ezt követően Dropout réteget alkalmaztunk (50%-os arány), hogy csökkentsük az overfitting kockázatát. A modell kimenete egy 62 neuronos Dense réteg, amely softmax aktivációval osztályozza a képeket a lehetséges karakterek között.

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_8 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_9 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_9 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_4 (Flatten)	(None, 1600)	0
dense_8 (Dense)	(None, 128)	204,928
dropout_4 (Dropout)	(None, 128)	0
dense_9 (Dense)	(None, 62)	7,998

Baseline CNN



A modell tanítása során 0.884 validation accuracy értéket értünk el, ami azt mutatta, hogy a hálózat képes jól általánosítani az adathalmazra.

Parameter Tuning

A paraméterek optimalizálása során több módszert is kipróbáltunk, hogy megtaláljuk a modellünk számára legmegfelelőbb beállításokat. Ezek közé tartozott a GridSearch, a Hyperband, valamint végül a Bayes-optimalizáló, amelyet a végső beadáshoz választottunk.

GridSearch

A GridSearch módszerrel egy előre definiált paraméterhalmazt próbáltunk ki, ahol minden néhány hiperparaméter összeállítást teszteltünk. A keresés során a következő értékeket használtuk:

```
param_grid = {
    'model__dropout_rate': [0.3, 0.5, 0.7],
    'batch_size': [32, 64],
    'epochs': [10, 15]
}
```

Eredmény:
Best: 0.8941360114345897 using {'batch_size': 32, 'epochs': 15, 'model__dropout_rate': 0.3}

Azonban hamar rá kellett jönnünk, hogy bár a GridSearch biztosítja, hogy minden kombináció tesztelésre kerül, a módszer nem skálázódik jól nagy paraméterterek esetén. A

futási idő exponenciálisan nő a paraméterek számával és értékeinek lehetséges tartományával, ami miatt ezt a megközelítés elvetettük a további alkalmazásra.

Hyperband

A Hyperband egy modern hiperparaméter-optimalizálási módszer, amely adaptív erőforrás-allokációt alkalmaz. Ez a módszer gyorsabban képes azonosítani a jó paraméterkombinációkat, mint a GridSearch. Az optimalizáló 5 iteráció és 10-es epoch szám mellett az ábrán látható paraméter értékeket találta legjobban teljesítőnek:

```
{'conv_1_filters': 96,
 'conv_1_kernel_size': 3,
 'conv_2_filters': 128,
 'conv_2_kernel_size': 3,
 'dense_units': 256,
 'dropout_rate': 0.2,
 'learning_rate': 0.001,
 'tuner/epochs': 10,
 'tuner/initial_epoch': 4,
 'tuner/bracket': 1,
 'tuner/round': 1,
 'tuner/trial_id': '0110'}
```

A Hyperband optimalizálás során a modell **0.9134** validation accuracy-t ért el, amely jelentős javulást jelentett a korábbi eredményekhez képest.

Bayes-optimalizálás

Végül a Bayes-optimalizálást alkalmaztuk, amely iteratív módon halad a valószínűleg jó paraméterek irányába a korábbi futások eredményei alapján. A módszer működését az alábbi pszeudo-kód szemlélteti:

Algorithm 1 Basic pseudo-code for Bayesian optimization

Place a Gaussian process prior on f
Observe f at n_0 points according to an initial space-filling experimental design. Set $n = n_0$.
while $n \leq N$ **do**
  Update the posterior probability distribution on f using all available data
  Let x_n be a maximizer of the acquisition function over x, where the acquisition function is computed using the current posterior distribution.
  Observe $y_n = f(x_n)$.
  Increment n
end while
Return a solution: either the point evaluated with the largest $f(x)$, or the point with the largest posterior mean.

Bayes-optimalizáló pszeudo-kódja[1]

A módszer előnye, hogy nem próbál végig minden kombinációt, ehelyett a már ismert eredmények alapján választja ki a következő tesztelendő paramétereket. A kísérletek során azokat a paramétereket, melyek a korábbi tapasztalatok alapján biztosnak tűntek, rögzítettük,

[1] arXiv:1807.02811

hogy az optimalizáció az instabil paraméterekre fókuszálhasson. Így változtak a tesztelt változók és ezeknek optimális értékei:

```
Validation Accuracy: 0.9039022326469421
Optimal Hyperparameters:
filters_1: 64
kernel_1: 3
filters_2: 192
kernel_2: 3
dense_units: 512
optimizer: adam
learning_rate: 0.0004828117543315003
batch_size: 128
```

1. kísérlet

```
Validation Accuracy: 0.9084156155586243
Optimal Hyperparameters:
filters_1: 32
kernel_1: 3
filters_2: 128
kernel_2: 5
dense_units: 512
dropout_rate: 0.4
optimizer: adam
learning_rate: 0.00027099568345591985
batch_size: 96
```

2. kísérlet

```
Validation Accuracy: 0.9094499349594116
Optimal Hyperparameters:
filters_1: 64
filters_2: 160
dense_units: 384
dropout_rate: 0.24714106985172238
learning_rate: 0.00011648878629496095
batch_size: 128
```

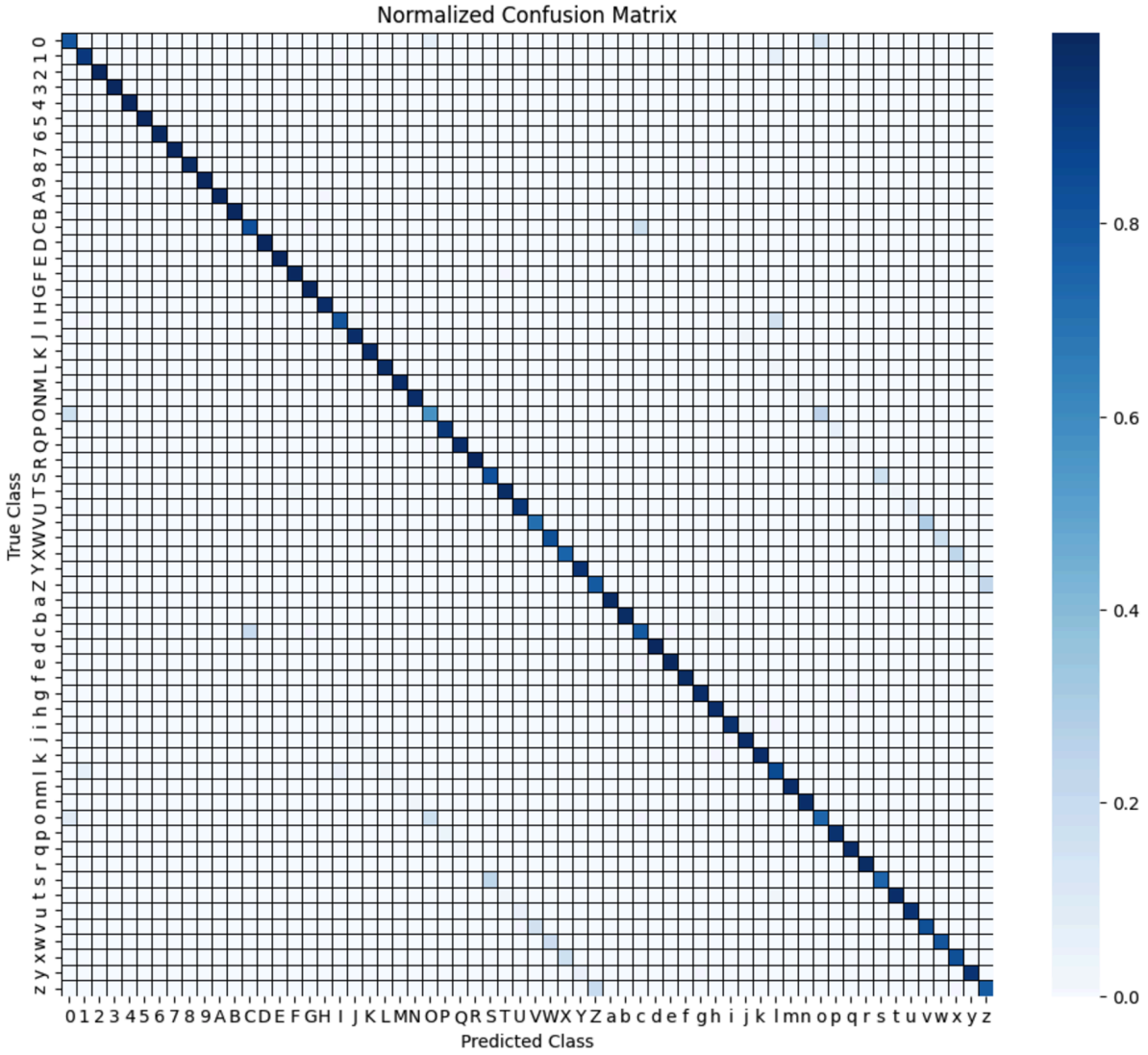
3. kísérlet

```
Validation Accuracy: 0.911894679069519
Optimal Hyperparameters:
filters_2: 223
dense_units: 501
dropout_rate: 0.251338450091664
learning_rate: 0.00013428186382661135
batch_size: 97
```

4. kísérlet

Az optimális epoch szám kiválasztásához 7-es vágású keresztvalidációt futtattunk a modellel. A keresztvalidáció során minden fold-ban alkalmaztunk early stoppingot, amely megakadályozta a overfittinget azáltal, hogy a tanítást leállította, amikor a validation accuracy három epoch-on keresztül nem javult tovább. Az optimális epoch szám a mérések alapján 16-18 közé esett, és az átlagos validation accuracy **0.9211** volt.

Konfúziós Mátrix



Konfúziós mátrix

A jobb oldali színsáv az accuracy-t reprezentálja, az ábrán pedig jól látható az egyes betűknél, hogy mennyire volt képes eltalálni a helyes osztályt a modell. A legkevésbé az O betűnél tudta, itt a 0-val keverte sokszor. Illetve a két párhuzamos egyenes arra utal következtetni, hogy a kis- és nagybetűket téveszti össze gyakran a modell jellemzően az abc második felében.

Training

A végső modell a következő felépítést követte:

- A bemenet shape-je a korábban meghatározottak szerint 28x28.
- Konvolúciós réteg 64 filterrel, (3, 3)-as kernel size-al és relu aktivációval.
- Max pooling réteg (2,2)-es mérettel.
- Második konvolúciós réteg 192 filterrel és szintén (3,3)-as kernel size-al és relu aktivációs függvénnyel.
- Második max pooling réteg ugyanolyan paraméterekkel.
- Flatten réteg.
- Dense réteg 512 dimenziójú outputtal és relu aktivációval.
- Dropout réteg 0,25-ös rátával.
- Végül még egy dense réteg 62 dimenziós outputtal, mert ennyi osztály van és softmax aktivációs függvénnyel.

Ezekkel a paraméterekkel a felépített modell validation accuracy-je **0,92**.

Prediction

A predikció során a test adathalmazban lévő képeket a traininghez használt képekhez hasonlóan átalakítottuk, majd a felépített modellünket használtuk és egy txt fájlba a megadott formátum szerint kiírtuk a predikció eredményét.
Az elvárt formátum a következő volt:
class;TestImage
19;Test0001.png
19;Test0002.png
45;Test0003.png
. . .

Assignment Evaluation

A feladat kiértékelése során a macro-average 0,786 lett, az accuracy: 0.818. Ez sajnos a saját adathalmazunkon kapott értékekhez képest alacsonyabb lett, ennek lehetséges oka a túltanulás vagy az adathalmazok vártnál nagyobb eltérése.

Conclusion

A feladat elvégzése során magabiztos tudást szereztünk a konvolúciós neurális hálók alkalmazásáról, valamint többféle paraméter optimalizálóval is mélyebben megismerkedtünk. Felépítettünk egy jó pontosságú modellt. Ami kissé meglepett minket, hogy a különböző optimalizálók hatékonyságában a vártnál kevesebb volt a különbség. Fejlődési lehetőség egy olyan modell építése, ami kis- és nagybetűket is képes elkülöníteni egymástól.