



Mesterséges intelligencia

Pytorch Image Classification

Deep Learning-el

Foodifier

Ételtípusok felismerése képről a Food 101 dataset alapján

Kálmán Dávid

LNF8KS

2023/2024/2

Tartalom

A probléma és a projekt leírása.....	3
Használt technológiák.....	4
Image 101 Dataset.....	4
PyTorch.....	4
fast.ai.....	4
ResNet.....	4
Használati útmutató.....	5
Github Repository.....	5
Telepítési útmutató.....	5
Használata fájlból.....	5
A program bemutatása.....	6
Szerkezeti felépítés.....	6
A model elkészítése.....	6
A model finomítása.....	8
A model használata.....	8
Irodalomjegyzék.....	9

A probléma és a projekt leírása

A projektem célja az volt, hogy egy mesterséges intelligencia alapú (jelen esetben deep learninget alkalmazó) programot készítsek, amely alkalmas megállapítani azt, hogy egy, a programba feltöltött képen milyen típusú étel szerepel.

Azért választottam ezt a témát, mert mindig is érdekelt a képfelismerés és az ehhez tartozó technológiai megoldások. A fejlesztés során a PyTorch gépi tanulási könyvtárat használtam, a modell képzése pedig ResNet-tel történt (több verziójával is próbáltam).

A témaválasztásomban leírt 4 db ételtípus helyett a végső verzió 48 darab típust tud felismerni, 0.33-as hibaértékkel. Ez a gyakorlatban azt jelenti, hogy az esetek 67%-ban sikeresen meg tudja mondani, hogy ténylegesen milyen étel szerepel a képen.

Természetesen ennek alap feltétele az, hogy a tesztelt fotón egy olyan ételtípus legyen, amely benne van a 48 darab betanított típusban, különben a program a hozzá leginkább hasonlító formát fogja megadni eredményként.

Model betöltése

```
1 learn_inf = load_learner('export.pkl')
2
3
4 local_image_path = '../test_images/stek.jpg'
5
6 img = PILImage.create(local_image_path)
7 img.show()
8
9 label, _, probs = learn_inf.predict(img)
10 label_index = probs.argmax() # A legnagyobb valószínűségű index
11
12 print(f"This should be a(n) {label}.")
13 print(f"Probability: {probs[label_index]*100:.2f}%")
```

[✓] 03s


...

...

...

This should be a(n) steak.
Probability: 98.12%

...



A model használata egyedi képpel és a kiírt eredmény

Használt technológiák

Image 101 Dataset

Természetesen mivel képfelismerésről van szó, így a program egyik legfontosabb része maga a tanításhoz használt dataset. Az én választásom a Food 101 datasetre esett, ami Kaggle-ön elérhető bárki számára. Ez az adathalmaz 100.000 darab képet tartalmaz összesen 101 féle ételtípusról. A programomnak ebből 48 darab típus lett megtanítva a végső verzióban.

PyTorch

A fejlesztés során a PyTorch gépi tanulási könyvtárra épült fel a programom. Azért választottam inkább ezt a TensorFlow-val szemben, mert képfelismeréshez sokkal jobban használható, több extension és library épült rá, amely ezzel foglalkozik (ilyen például a lentebb említett fast.ai).

fast.ai

A fast.ai egy, a PyTorch-ra épülő gépi tanulási könyvtár, amelyet a gyors és hatékony mélytanulási modellfejlesztés támogatására terveztek. Ez a keretrendszer lehetővé teszi a gyors prototípuskészítést és a mélytanulási modellképzést minimális kódsorok felhasználásával.

ResNet

A ResNet egy hatékony mély neurális hálózat architektúra. Előnyei közé tartozik a könnyebb tanulás és a mélyebb hálózatok építésének lehetősége.

A fast.ai használatával létrehozott modellem az előre tanított `resnet34` architektúrát alkalmazza, ami segít az általános képfelismerési feladatok hatékony megoldásában.

Használati útmutató

Github Repository

A projekt során használt GitHub repository linkje:

https://github.com/davidkalmn/image_classification_food_ML.git

Telepítési útmutató

0. Telepítsük a számítógépre a Python 3.11-es verzióját (a PyTorch Windowson csak a 3.8 és 3.11 közötti verziókat támogatja, így célszerű ezek közül a legfrissebbet használni a hibák elkerülése végett).
1. Klónozzuk a GitHub repository-t a kívánt célmappába terminálon keresztül:
 - `git clone https://github.com/davidkalmn/image_classification_food_ML.git`
2. Telepítsük a szükséges csomagokat terminálon keresztül:
 - `pip install torch torchvision torchaudio`
 - `pip install -Uqq fastai`

Használata fájlból

Ha lokálisan, fájlból szeretnénk használni a Foodifier-t:

1. A klónozott git mappán belül keressük meg a **`/model/use_food_model.ipynb`** fájlt, majd nyissuk meg Visual Studio Code-al, vagy valamilyen ehhez hasonló fejlesztői környezettel.
2. A megnyitott Jupiter Notebook kiterjesztésű fájlban futtassuk le az első 2 kódblokkot, amelyek az Előkészületekhez tartoznak.
3. Az utolsó kódblokkban, a 3. sorban adjuk meg a **`local_image_path`** változó értékeként a tesztelésre szánt kép elérési útvonalát, majd futtassuk le a kódblokkot.
4. A program végül megmutatja az általunk használt képet, valamint kiírja, hogy ez szerint milyen típusú étel, illetve, hogy mekkora eséllyel „tippeli” a típusát.

A program bemutatása

Szerkezeti felépítés

A program szerkezetét 3 nagy csoportra lehet bontani:

- **Model:** 3 külön fájlban, a model létrehozása, további optimalizálása, valamint a model használata. Ez a három fájl a **/model** mappán belül található.
- **Assets:** Ide tartozik maga a dataset, amelyet a model létrehozásánál és optimalizálásánál használunk. Ezt automatikusan feltelepíti a program a gépünkre, amikor legelőször elindítjuk.
- **Test images:** ezek azok a képek, amelyeket nem a model training-elésére használunk, hanem a tesztelésére, hogy ténylegesen jól működik-e.

A model elkészítése

A model elkészítése a **/model/create_food_model.ipynb** fájljal történik, azon belül is 4 lépésben:

1. **Load and prepare data:** az itt megtalálható kód fogja importálni a fast.ai-t a projektbe, hogy később használni tudjuk a parancsait, valamint ez telepíti fel a dataset-et is a gépre (végül egy tesztet is futtat, hogy sikeres volt-e a telepítés).

```
Load and prepare data

#!pip install -Uqq fastai
from fastai.vision.all import *

[1]

foodPath = untar_data(URLs.FOOD)

[2]

get_files(foodPath)

[3]

... (#48018) [Path('C:/Users/mauzi/.fastai/data/food-101/classes.txt'),Path('C:/Users/mauzi/.fastai/data/food-101/ex

len(get_image_files(foodPath))

[4]

... 48000
```

2. **Training:** itt a program először létrehozza a label-öket (azaz az ételtípusokat), majd a fast.ai rövidített parancsai által létrehozza a model első változatát (jelen esetben a 48000 kép 80%-át használtam trainingre, a maradék 20%-ot tesztelésre). A fejlesztés során kipróbáltam több ResNet típust is (ResNet50 és ResNet100), minél magasabb a szintje, minél nagyobb a modelszám, annál pontosabb a végeredmény, viszont drasztikusan tovább is tart a létrehozás (már az 50-es verziónál is több mint 20 perc volt csak az első training 5-10 perc helyett).

```
[12] learn = cnn_learner(dls, resnet34, metrics=error_rate, pretrained=True)

[13] learn.fine_tune(epochs=1)

...

... epoch train_loss valid_loss error_rate time
      0 3.892819 3.492297 0.863125 04:56

...

... epoch train_loss valid_loss error_rate time
      0 3.197243 2.951655 0.768958 11:38
```

3. **Verify model:** ez a hitelesítési folyamat, itt használhatjuk a dataset maradék 20%-át, vagy pedig megadhatunk saját képeket is, ezzel is ellenőrizve, hogy biztosan jól működik-e a program.
4. **Deploy & export:** az utolsó lépésben exportáljuk a létrehozott és letesztelt modelt, hogy a továbbiakban másik fájlokban is tudjuk akár tovább finomítani vagy tesztelni.

```
Deploy & export

[18] learn.export()

[19] modelPath = get_files(foodPath, '.pkl')[0]
      modelPath

... Path('C:/Users/mauzi/.fastai/data/food-101/export.pkl')
```

A model finomítása

A model finomítása a **/model/train_food_model.ipynb** fájlban történik viszonylag egyszerűen. Beimportáljuk a már meglévő, elmentett modelünket, betöltjük a kívánt tanítási, finomítási metódust, megadjuk az epoch-ok számát (hogy hányszor, hány körben csinálja újra az optimalizálást, minél több, annál pontosabb lesz), majd elindítjuk a finomítást.

Ez rengeteg időt vehet igénybe, amely függ a használt képek méretétől, mennyiségétől, valamint a hardver erősségétől. Az általam használt mód 5 ismétléses volt, ismétlésenként tartott kb. 18-19 percbe (összesen közel 1,5 óra). Ennek a lényege az, hogy minden ismétléssel, minden új finomítási folyamattal egyre pontosabb és pontosabb lesz a model (volt olyan teszt modellem, amely az eredeti 0.86-os hibarártáról ment le 0.36-ra közel fél napnyi folyamatos training után).

```
#item_tfms=Resize(380)
dls_fine_tune = ImageDataLoaders.from_name_func(
    foodPath, get_image_files(foodPath), valid_pct=0.1, seed=42,
    label_func=GetLabel, item_tfms=Resize(100))

learn = load_learner('export.pkl')

learn.dls = dls_fine_tune

1 learn.lr_find()
2 suggested_lr = learn.lr_find().valley

Outputs are collapsed ...

#lr_max=slice(1e-7, 1e-6), base_lr=suggested_lr
learn.fine_tune(epochs=5)

...
...
epoch  train loss  valid loss  error rate  time
0      1.321031   1.434408   0.388750   14:00
...
...
epoch  train loss  valid loss  error rate  time
0      1.322745   1.410115   0.386042   18:30
1      1.197774   1.352705   0.366667   18:31
2      0.766090   1.352503   0.350417   18:38
3      0.362634   1.381094   0.332083   18:32
4      0.189382   1.399408   0.333750   18:29

learn.export()
```

A model használata

A model használata a **/model/use_food_model.ipynb** fájljal történik. 3 rövid kódblokkból áll, az első importálja a fast.ai-t, a következő létrehozza az ételkategóriákat, az utolsó pedig betölti a modelt, majd használja az általunk megadott képen. Eredményül látunk egy javaslatot, hogy a program szerint milyen étel van a képen, illetve egy hozzá tartozó valószínűséget.

Irodalomjegyzék

Dataset:

- <https://www.kaggle.com/datasets/dansbecker/food-101>

PyTorch:

- <https://pytorch.org/tutorials/>
- <https://www.learnpytorch.io/>

fast.ai:

- <https://www.fast.ai/>
- <https://course.fast.ai/>

Resnet:

- <https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/>
- <https://medium.com/@ibtadaazeem/understanding-resnet-architecture-a-deep-dive-into-residual-neural-network-2c792e6537a9>
- <https://roboflow.com/model/resnet-34>