Útmutató a keretrendszer használatához

Követelmények

- Python ^3.7
- TensorFlow 2.0
- Numpy ^1.17

Felépítés

A fő modulok a `vse_training_fw` gyökérkönyvtárban találhatóak.

- `train.py` Tanítás.
- `evaluate.py` Kiértékelés VAL és FAR metrikákra.
- `embed.py` Képek beágyazása vektortérbe.
- `test.py` Megadott adathalmaz képeihez tartozó bálnák azonosítása.
- `training_session.py` Több tanítási ciklus egymás utáni automatikus futtatása.

Segédprogramok a `tools` könyvtáron belül. Részletekért lásd a forráskód dokumentációt.

Modellek létrehozására a `model` könyvtáron belül találhatóak önállóan futtatható modulok. A futtatást követően a szkript felépít és lemezre menti a kódban meghatározott konvolúciós neurális hálót. További információ a forráskódban.

A `config` mappában található konfigurációs fájlokról bővebben lásd a *Konfigurációs fájlok* részt.

Adathalmaz könyvtárstruktúra

Egy adathalmazt egy könyvtár elérési úttal kell megadni. A megadott könyvtárnak tartalmazni kell egy `images` mappát, benne ömlesztve a képekkel. Az `images` mappa mellett egy `labels.csv` fájlnak kell tartalmaznia a bálna azonosító adatokat a következő formában:

```
Image,Id
0000e88ab.jpg,w_f48451c
0001f9222.jpg,w_c3d896a
00029d126.jpg,w_20df2c5
...
```

Az első oszlop a kép fájl neve, a második a képhez köthető bálna azonosítója.

Például legyen a tanuló adathalmaz elérési útvonala

`C:\my_data\training_data`.

Ekkor a belső struktúrája:

```
training_data\
|--- labels.csv
|--- images\
    |--- 0000e88ab.jpg
    |--- 0001f9222.jpg
    |--- 00029d126.jpg
    |--- ...
```

Konfigurációs fájlok

Adat elérési útvonalakat, mentési könyvtárakat és hiperparamétereket konfigurációs fájlokban lehet meghatározni.

A `config` mappának tartalmaznia kell egy `dir_config.ini` nevű fájlt, ami a fő elérési útvonalakat tartalmazza fájlok mentéséhez és olvasásához. A fájlban lévő paraméterek értékeit a felhasználó saját maga adhatja meg az alapján, hogy hol szeretné ezeket az adatokat tárolni. További részletek a `dir_config.ini` fájlban.

A futtatási konfigurációs fájl a fő modulok számára tartalmaz az adott műveletre (tanítás, kiértékelés, stb...) vonatkozó paramétereket, illetve a `dir_config.ini` fájlban található útvonalakhoz képesti relatív útvonalakat. Egy minta erre `config` mappában található `run_config_example.ini` fájl. Ez a fájl tetszőleges számban másolható és tetszőleges néven tárolható. A paraméter értékek a kívánt módon változtathatóak. További részletek a `run_config_example.ini` fájlban.

Funkciók használata

A bemutatott modulok és segédprogramok a paranccsorból indíthatóak. Ehhez ajánlott a keretrendszer gyökérkönyvtárába navigálni és onnan futtatni a fájlokat, különben néhány program nem fog működni.

A fő modulok mindegyike egyetlen argumentumot vár, ami egy futtatási konfigurációs fájl elérési útvonala. A konfigurációs fájlra látható példa a `config` mappában a `run_config_example.ini`.

Példa adathalmaz és beállítások

Az `example` mappában elhelyeztem a kisebb próba adathalmazt és könyvtárstruktúrát az eredmények és a modellek tárolására. A `config` mappában található `dir_config.ini` ennek a használatára van konfigurálva, de ez természetesen tetszőlegesen megváltoztatható.

Az adathalmazt a `dataset` mappában két részre osztottam. A `train` a tanuló adathalmaz és a `validation` a kiértékelő vagy validációs adatokat tartalmazza. A `logs` mappa tárolja a metrika értékeket, az `embeddings` a beágyazott vektorokat és a hozzájuk tartozó meta adatokat (ez esetben a bálnák azonosítója), a `models` mappában pedig elhelyeztem egy próba modellt `my_resnet.h5` néven.

A próba modellen lefutattam egy 6 epoch hosszúságú tanítást a próba adathalmazon, aminek az eredményei a fent felsorolt mappákban találhatóak.

A `test.py` modul a kNN algoritmus eredményeit az `embeddings` mappába menti le, a beágyazott vektorok mellé. Ezek az eredmények

- `eval.txt` A sikeresen azonosított bálnák százalékos aránya.
- `submission.csv` A kNN algoritmus által kiválasztott *k* legközelebbi szomszéd azonosítója a validációs halmazban lévő minden kép esetén. Itt látható, hogy milyen bálna azonosítókat rendelt hozzá a rendszer az egyes képekhez. Az első oszlop az azonosítani kívánt kép neve (zárójelben pedig a hozzá tartozó **tényleges** bálna azonosító). A második oszlop pedig a kNN algoritmus által legközelebbinek talált bálna azonosítóknak a felsorolása. Ha a tényleges azonosító köztük van, akkor az egy sikeres azonosítás.

Példa a `submission.csv` fájl tartalmára:

```
Image,Id
0052ce2f5.jpg(w_2365d55),w_fccccec w_f48451c w_20950a9 w_990921b
009687166.jpg(w_778e474),w_efbdcbc w_8a1b71c w_60cf87c w_08630fd
00d07df97.jpg(w_5650932),w_f765256 w_c1d2fbe w_e07b79f w_955bfe2
...
```

A `config\run_config_example.ini` azokat a konfigurációs beállításokat tartalmazza, amelyeket a példa létrehozásához használtam.

Új modell létrehozása

```
vse_training_fw> python model/resnet.py <konfigurációs_fájl>
```

A konfigurációs fájlban a `model_name` paraméter értéke lesz az új modell neve, illetve relatív elérési útvonala. Tartalmaznia kell a `.h5` kiterjesztést is. A modell bemeneti rétege az `input_shape` paraméterben megadott értéket fogja felvenni.

Példa modell név megadására: `model_name = new_resnet.h5` vagy új alkönyvtár megadásával `model_name = resnets/new_resnet.h5`. A teljes útvonal a `dir_config.ini` fájlban található `models = ../example/models` kombinálva `../example/models/new_resnet.h5` illetve `../example/models/restnets/new_resnet.h5`.

Tanítás

```
vse_training_fw> python train.py <konfigurációs_fájl>
```

A konfigurációs fájl `model_name` paraméterben megadott modellt fogja tanítani a `[train]` kategórián belül megadott paramétereknek megfelelően. A `log` paraméterben megadott relatív könyvtárba fog kerülni a tanítás alatt mért veszteségek értékei. Ha a `log = resnet_train` és a `dir_config.ini` fájlban a `logs_root = ..\example\log` akkor a létrejövő .csv fájlok a `..\example\log\resnet_train\` mappába fognak kerülni.

Vektortérbe ágyazás (Vector-Space Embedding)

```
vse_training_fw> python embed.py <konfigurációs_fájl>
```

A beágyazott jellemzővektorok a `dir_config.ini` fájlban beállított `embeddings_root` és a paraméterül átadott konfigurációs fájlban lévő `embeddings` relatív útvonal kombinációja által megjelölt útvnonalon lesznek tárolva. Például ha az `embeddings_root = ..\example\embeddings` és az `embeddings = resnet`, akkor a `vecs.tsv` és `meta.tsv` fájlok a `..\example\embeddings\resnet\` mappában lesznek lementve. A beágyazott adatokat a `dataset_dirs` alatti `train` paraméter által megadott adathalmazból képzi a `model_name` modell.

Bálnák azonosítása

```
vse_training_fw> python test.py <konfigurációs_fájl>
```

Ehhez a művelethez szükség van, hogy az `embeddings` által megadott helyen már legyen egy `vecs.tsv` és `meta.tsv` fájl. A `dataset_dirs` alatti `validation` útvonalon lévő adathalmazból képzi az azonosítandó jellemzővektorokat. A kNN algoritmus a legközelebbi szomszédokat az említett `vecs.tsv` fájlban elmentett vektorok közül keresi meg és látja el a `meta.tsv` fájlban lévő címkékkel. Az eredményeket az `embeddings` relatív útvonalra menti. A validációs halmaz méretétől és a `vecs.tsv` által tárolt vektorok mennyiségétől függően ez akár több tíz percig is eltarthat!

Modell kiértékelés ROC görbével

```
vse_training_fw> python evaluate.py <konfigurációs_fájl>
```

A `model_name` paraméterben megadott modell kiértékelése a `dataset_dirs` alatti `validation` halmazon. Az eredmény a `log` relatív útvonalon megadott mappába kerül `val_far.csv` fájlban. A vizsgált `treshold` pontok mennyiségét és eloszlását a forráskódban lehet átírni. Részletekért lásd a `evaluate.py` kódot.

Training session

```
vse_training_fw> python training_session.py <konfigurációs_fájl>
```

Lehetővé teszi több tanítási ciklus automatikus lefuttatását. Például 8 ciklus, amelynek mindegyike 10 epoch hosszan tanítja a megadott modellt. Az egyes ciklusok között snapshot mentéseket készít a modellről a beállított gyakoriságban.

Ha a `snapshot_point = 1`, akkor minden ciklus végén elmenti a `model_name` paraméterben megadott modellt `model_name_<ciklus száma>.h5` néven. A `snapshot_point = 2` érték esetén minden második ciklusban ment és így tovább. Példa: `model_name = resnet.h5` és `snapshot_point = 2`, akkor a második ciklusban készít egy `resnet_2.h5` mentést, a negyedikben `resnet_4.h5` és így tovább. A ciklusok között lehetőség van a tanulási ráta ütemezett változtatására az elérhető `decay` metódusokkal. Részletekért lásd a `training_session.py` forráskódot.