spacenet-building-detection

Środowisko

Program jest napisany w Pythonie 3.6 i był uruchamiany na systemie Ubuntu 17.10.

Struktura projektu

- data
  - 3band - zdjęcia RGB
  - 8band - zdjęcia multispektralne
  - dist_transforms - transformaty odległościowe
  - masks - maski binarne (ground truth)
  - scaled - przeskalowane zdjęcia (256x256)
  - vectordata - oryginalne wektorowe dane testowe (wielokąty)
- image_plots - wykresy porównujące zdjęcia wyjściowe sieci z oryginalnymi danymi
- models - zapisane modele sieci
- output - testowe dane wyjściowe sieci w formacie gotowym do porównania
- results - pliki csv z danymi zawierającymi koszty dla zbioru treningowego i testowego w ilości epok
- scripts - pomocnicze skrypty w bashu
- src - kod aplikacji

Instalacja zależności

Instalacja bibliotek Pythonowych:

```
make requirements
```

Instalacja GDAL:

```
sudo apt-get install libgdal1-dev python3-gdal
export CPLUS_INCLUDE_PATH=/usr/include/gdal
export C_INCLUDE_PATH=/usr/include/gdal
pip3 install gdal
```

Pobranie pomocniczych narzędzi SpaceNet:

```
git clone https://github.com/SpaceNetChallenge/utilities.git utilities
git clone https://github.com/SpaceNetChallenge/BuildingDetectorVisualizer.git visualizer
```

Napisany program jest przystosowany do uruchamiania na GPU (wykorzystuje tensorflow-gpu). Można go uruchamiać także na CPU (wtedy należy zainstalować tensorflow zamiast tensorflow-gpu), ale w przypadku posiadania karty graficznej NVidii będzie to zdecydowanie wolniejsze. Uruchomienie Tensorflow na GPU wymaga instalacji CUDA Toolkit 8.0 oraz cuDNN v6.0 (szczegóły odnośnie wsparcia dla GPU można znaleźć na stronie https://www.tensorflow.org/install/install_linux, cały proces jest niestety dość skomplikowany).

Pobranie danych

Dane ze SpaceNet znajdują się w bucketach AWS S3. Do ich pobrania potrzebne jest konto na AWSie i skonfigurowanie AWS CLI - https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html. Następnie dane można pobrać następująco:

```
make download-data city={miasto}
```

Dostępne miasta: rio, vegas, paris, paris2, shanghai, khaortum. Program jest przystosowany do pracy na danych z Rio, użycie danych z innych miast może wymagać niewielkich zmian w kodzie (inna struktura folderów).

Preprocessing

Na proces preprocessingu składają się:

- losowe przemieszanie danych
- przeskalowanie obrazków
- generacja masek
- generacja transformat odległościowych

Całość zawiera się w skrypcie preprocess.py i można go uruchomić za pomocą make'a:

```
make preprocess
```

Uczenie

Definicja sieci znajduje się w pliku unet.py, a skrypt do trenowania i testowania sieci w pliku train.py (w tym pliku zdefiniowane są także parametry do uczenia sieci). Uczenie można uruchomieć poleceniem:

```
make train
```

Testowanie i postprocessing

Do obliczenia F1-score i wizualizacji rezultatów można użyć narzędzia SpaceNetVisualizer:

```
java -jar ./visualizer/visualizer-1.1/visualizer.jar -truth
./output/geojson/truth.csv -solution
./output/geojson/result{data_obliczen}.csv -image3-dir
./data/rio/3band -image8-dir ./data/rio/8band -band-triplets
./visualizer/visualizer-1.1/data/band-triplets.txt
```

Aby narysować wykres kosztu na zbiorze treningowym i testowym w funkcji epok można wywołać:

```
python3 ./src/visualize.py cost --file
./results/csv/{data_eksperymentu}.csv --output
{ścieżka_pliku_wynikowego} --title {tytuł_wykresu}
```