Circle Finder Marathon Challenge

Matthieu Zins (sowlosc)

Solution

My solution is built on Mask R-CNN/Faster R-CNN, two neural networks for instance segmentation and object detection.

Image pre-processing

In order to get information both from the panchromatic images and the multispectral images, I used pan-sharpening. I kept only the Red, Green and Blue channels. Note that I also tested to keep the 8 channels of the image and add some convolutional layers at the beginning of my network, but this did not reach the same level of accuracy. Probably because the backbone of the network is initialized with weights pretrained on ImageNet which has only natural RGB images.
This image pre-processing has to be done both on training images and testing images.

Object detection

My solution treats the problem as object detection in the form of bounding boxes. Then, the inscribed circles are extracted from the detection boxes. I used Mask R-CNN provided in Detectron2 (https://github.com/facebookresearch/detectron2). This network is actually designed for instance segmentation, but I found that training both for detection and segmentation was slightly better than only detection (with Faster R-CNN).

It is possible to use Faster R-CNN for inference, with the trained weights of Mask R-CNN. Both networks have the same architecture, except the segmentation branch. As we do not use segmentation during inference, removing this branch goes slightly faster.

Processing requirements

My solution assumes that a folder called */wdata* is available with write access.

The following commands need to be executed in the */code* folder.

Test: *sh test.sh /data/test /wdata/solution*
Train: *sh train.sh /data/train*

The docker has to be started by:
docker run -v <local_data_path>:/data:ro -v <local_writable_area_path>:/wdata -it <id>

(By default, the entry point calls: *sh test.sh /data/test /wdata/solution*)

The training script saves the trained model into */wdata/checkpoint/model_final.pth.*

- By default (if no */wdata/checkpoint/model_final.pth* exists), the testing script downloads my home-trained weights, so that it is possible to test without training.
- If training has been run and */wdata/checkpoint/model_final.pth* generated, this file will be used instead.

Home-trained weights:
https://www.dropbox.com/s/ham5s9kq4yzpxx2/model_final_20000_clean_train_all.pth?dl=0

The *.geojson* files are generated by *test.sh* into the folder passed as a second argument (here */wdata/solution*).

A CSV file (*/wdata/split_file.csv*) is also generated by *test.sh*. It is the file that I used to run the scorer tool locally on my computer. It contains the name of all the predicted geojson files with the indication “provisional”, but ,if I understood well, you will use your own split file.

I limited my training to 20000 iterations, which runs in ~1.5 days on the cpu of my computer. It should respect the 2 days limits on AWS (even if it is hard to be completely sure because the hardware is not exactly the same). This maximum number of iterations is just a variable in the code and could be reduced if the training is really too long (already at 8000 iterations the results are not too bad). If more training time is available, increasing the number of iterations could also improve the results.

Other details

- Language of code:
  - Python
- Map projection necessary to run code:
  - I use only the projection information provided in the TIF images.
- Source(s) of unclassified remote sensing imagery
  - I use only the training images provided in this challenge.

Licence