Reproducibility review of: Extraction of linear structures from digital terrain models using deep learning

Daniel Nüst and Anita Graser

2021-06-07

This report is part of the reproducibility review at the AGILE conference. For more information see https://reproducible-agile.github.io/. This document is published on OSF at https://osf.io/2sc7g/. To cite the report use

> Nüst, D., & Graser, A. (2021, April 30). Reproducibility review of: Extraction of linear structures from digital terrain models using deep learning. https://doi.org/10.17605/osf.io/2sc7g

Reviewed paper

> Satari, R., Kazimi, B., and Sester, M.: Extraction of linear structures from digital terrain models using deep learning, AGILE GIScience Ser., 2, 11, https://doi.org/10.5194/agile-giss-2-11-2021, 2021.

Summary

The provided workflow was **partially reproduced**. Based on the provided test file and instructions, I was able to recreate the computing environment and run the segmentation models. Relevant tables from the paper could be recreated. The training and validation part of the workflow is irreproducible because of proprietary data, therefore no figures could be recreated.

Reproducibility reviewer notes

The first manuscript did not include a DASA section, but the authors provided a link to the project repository https://github.com/Bashirkazimi/agile-submission-2021. The README file provides a step by step documentation.

```
git clone https://github.com/Bashirkazimi/agile-submission-2021

# first try using my local Python and GDAL, but Python 3.8 does not have tensorflow-gpu available in the required old version
gdalinfo --version
#GDAL 3.2.1, released 2020/12/29

# install python3.5, dev needed for GDAL
# sudo add-apt-repository ppa:deadsnakes/ppa
# sudo apt install python3.5 python3.5-dev

# in agile-submission-2021
mkvirtualenv agile-010 -p python3.5 -r requirements.txt
# following commands all in the venv

pip install GDAL==3.2.1
```

Next, I downloaded and unzipped the data per instructions from here, and tried to run the script.

```
python create_dataset.py
```

Leads to an error:

```
libcublas.so.9.0: cannot open shared object file: No such file or directory
```

So, let's check if I have a CUDA-compatible graphics card:

```
sudo lshw -C display

*-display
       description: VGA compatible controller
       product: UHD Graphics 620 (Whiskey Lake)
       vendor: Intel Corporation
       physical id: 2
       bus info: pci@0000:00:02.0
       version: 00
       width: 64 bits
       clock: 33MHz
       capabilities: pciexpress msi pm vga_controller bus_master cap_list rom
       configuration: driver=i915 latency=0
       resources: irq:150 memory:bb000000-bbffffff memory:60000000-7fffffff ioport:3000(size=64) memory:c0000-dffff
```

Problem: I don't have the right graphics card! The authors said it should be possible with `tensorflow-cpu`, so I create `requirements-cpu.txt` with `tensorflow-cpu` version `1.15.0`, the only one for version `1.x`.

```
pip uninstall tensorflow_gpu
pip install -r requirements-cpu.txt
```

Need to make adjustments in `utils/utils.py` for my version of GDAL, later also in `HRNetBinary../evaluate.py`

```
# instead of import gdal, gdalconst
from osgeo import gdal, gdalconst

# instead of import ogr, osr
from osgeo import ogr, osr

python create_dataset.py
# a lot of output...
```

Creates a lot of `.tif` files in `data/test`.

```
list.dirs(here::here("010/agile-submission-2021/data/"), full.names = FALSE)


## [1] ""       "dtms"   "labels" "test"   "test/x" "test/y"
```

Evaluate segmentations

Next step in the instructions, binary segmentation wit HRNet:

```
cd HRNetBinarySegmentation
python evaluate.py --evaluation_file=evaluation_file.csv
# [..]
Total params: 6,459,588
Trainable params: 6,446,018
Non-trainable params: 13,570
# [..]
```

Takes some time to run... then finishes with:

```
[...]
192/192 [==============================] - 663s 3s/step
[0.2579103708461288, 0.9069677637591589, 0.7762661981036123, 0.8187425669110917, 0.7393777735160645]
```

And the file `evaluation_file.csv` is created as mentioned in the instructions. The values in the file match the column `HRNet` of Table 1, within a level of precision to be expected from such a classification.

```
# note the file name in the instructions has different capitalisation
cd SegnetBinarySegmentation
python3 evaluate.py --evaluation_file=evaluation_file.csv
```

This finished within a minute! These values match the column `SegNet` of Table 1, within a level of precision to be expected from such a classification.

```
hrnet <- read.csv("agile-submission-2021/HRNetBinarySegmentation/files/evaluation_file.csv")
segnet <- read.csv("agile-submission-2021/SegnetBinarySegmentation//files/evaluation_file.csv")

suppressPackageStartupMessages(library("tidyverse"))
dplyr::full_join(hrnet, segnet) %>%
  knitr::kable()
```

acc	f1_m	loss	model_type	precision_m	recall_m	specific_name
0.9069678	0.7762662	0.2579104	hrnet	0.8187426	0.7393778	simple_binary
0.8787796	0.6977567	0.3094499	segnetCustomized	0.7688990	0.6399856	simple_binary

Run the next segmentation:

```
cd multiclassSegmentation
python3 evaluate.py --evaluation_file=evaluation_file.csv
```

This completes and recreates the data in Table 3 within reasonable numerical precision based on the file `multiClassEvaluation.csv`. It is unclear to me how Table 2 can be constructed from `evaluation_file.csv` of this segmentation, but I assume it can be.

```
multi <- read.csv("agile-submission-2021/multiclassSegmentation/files/multiClassEvaluation.csv")

rows <- lapply(c(0:5), function(class) {
  classValues <- multi %>%
    dplyr::select(dplyr::ends_with(as.character(class)))
  names(classValues) <- c("sparse_iou", "prediction", "recall", "f1.score", "support")
  c(`Class label` = as.character(class), classValues)
})

dplyr::bind_rows(rows) %>%
  knitr::kable()
```

Class label	sparse_iou	prediction	recall	f1.score	support
0	0.8952831	0.9270560	0.9630006	0.9446866	12342971
1	0.2073642	0.4009994	0.3745212	0.3873083	247492
2	0.0324831	0.3139039	0.0373107	0.0666941	282037
3	0.7648531	0.8797419	0.8552982	0.8673479	1961849
4	0.4453632	0.6953347	0.5580455	0.6191711	830866
5	0.1593569	0.2697686	0.2787574	0.2741893	145345

Train and make predictions

In each experiment folder:

```
python3 train_segmenter.py
python3 predict_sliding.py --output_shp_file=vector.shp --input_dtm=test_dtm.tif

Leads to an error:
Traceback (most recent call last):
  File "train_segmenter.py", line 111, in <module>
    with open(os.path.join(train_dir, 'filenames.txt'), 'r') as reader:
FileNotFoundError: [Errno 2] No such file or directory: '../data/train/filenames.txt'
```

The path seems to be wrong, there only is `filenames.txt` in `data/test`, but when using that, the code is missing the file `../data/valid/filenames.txt`. A second view and short exchange with the authors made clear that the dataset is proprietary and belongs to a collaboration partner.

The diff of all changes made to the repository is stored in the file `repro-review.diff` published with this report and also in the fork github.com/reproducible-agile/agile-submission-2021.

The workflow was partly reproduced. The data preparation (extraction with ArcGIS) is not reproducible, but the different segmentation algorithms could be executed. The actual training of the models and prediction was not reproducible because of unavailable training and validation datasets. The code seems to be complete tough and is readable, documented, and structured. No code was provided to recreate any figures.

Comments to the authors

A decent project, but there are always some things you can do better, here are some suggestions.

- Mostly a good job with the instructions in the README - please always test your own instructions - the file `test_labels.tif` was at the wrong location in the file share; ideally you store these file in a more long term location, for example by publishing your GitHub repo and the files in a Zenodo repository
- Suggest to provide a small script or CLI command to download the dataset, or mention to which directory the downloaded files should be extracted (`<project root>/data/dtms` or `<project root>/dtms`?)
- Note the code you "borrowed" from https://github.com/niecongchong/HRNet-keras-semantic-segmentation does not have a license attached to it, so I assume you asked the original author explicitly for permission to republish the code under the MIT license
- For machine learning workflows that use properitary data, please consider sharing a *synthetic dataset* so that the full workflow can be executed by others
- Please provide more explicit documentation what part of the code reflects what part of the paper (mentioning chapters in the README, mentioning scripts in the paper)