Python-based deep learning for detecting ditches from elevation data

Holger Virro & Wai Tik Chan

University of Tartu

Nordic-RSE conference

May 31, 2024

Where we come from

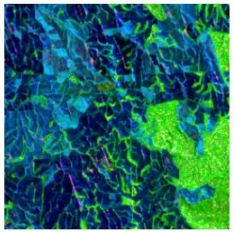
Home institution: Department of Geography, **University of Tartu**

Our workgroup: **Landscape Geoinformatics Lab**, led by **Prof Evelyn Uuemaa**

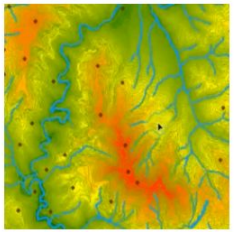
https://landscape-geoinformatics.ut.ee/

https://github.com/LandscapeGeoinformatics

Spatial Analysis and Spatial Machine Learning



Environmental Remote Sensing



Environmental Modelling

Who we are

Holger Virro

- Research Fellow in Geoinformatics
- PhD in Geoinformatics, University of Tartu

Wai Tik Chan

- Scientific Programmer
- MSc in Computer Science, University of Tartu

My perspective

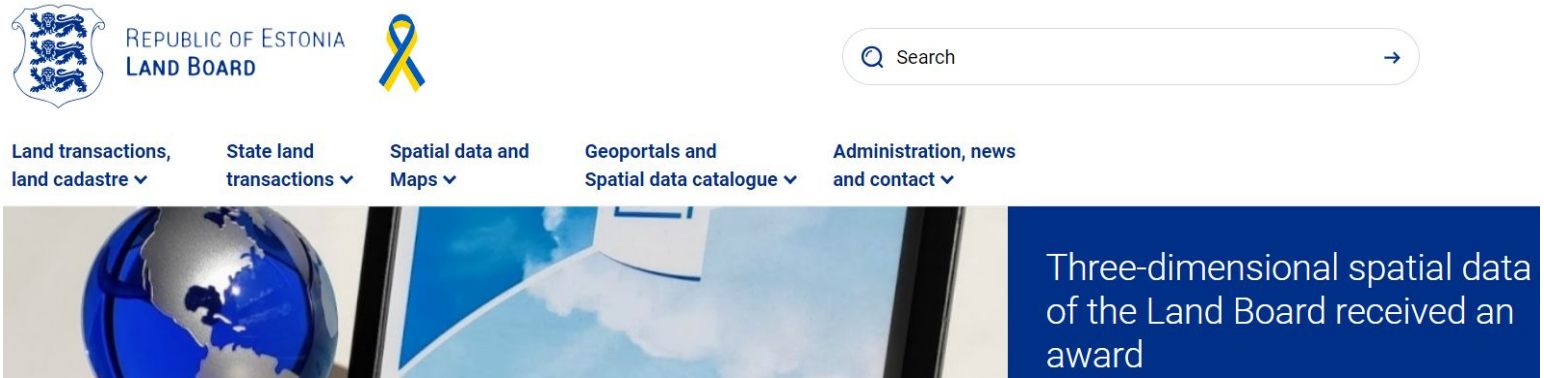
From a researcher point of view

- No experience in writing “production” software
- No previous experience with DL
- What should I handle on my own?
- What should I let the RSE handle?

General background

Estonian Land Board (Maa-amet)

- Manages Estonian geospatial datasets
- Interested in leveraging AI to make mapping more efficient
- Inspired by Finnish National Land Survey



Project background

Why **ditches**?

- Estonia has a dense drainage ditch network
- Many ditches not mapped as they are hard to detect from orthophotos
- Accurate ditch data required for modeling hydrology, carbon fluxes etc.

Perhaps **deep learning** (DL) can be used to map them?

Starting point

Found an existing ditch detection model from Sweden (**Lidberg et al., 2023**)

- **U-Net** architecture
- Input: digital elevation model (DEM) images processed with high-pass median filter (**HPMF**)
- **TensorFlow 2.6**
- Packaged in a **Docker** image
- https://doi.org/10.1061/JIDEDH.IRENG-9796

Issues with trying to run existing model

No experience with Docker

- Took some much time to set it up

Still did not run properly in the end

- “Graphics card and GPU difference” issue according to author 😞

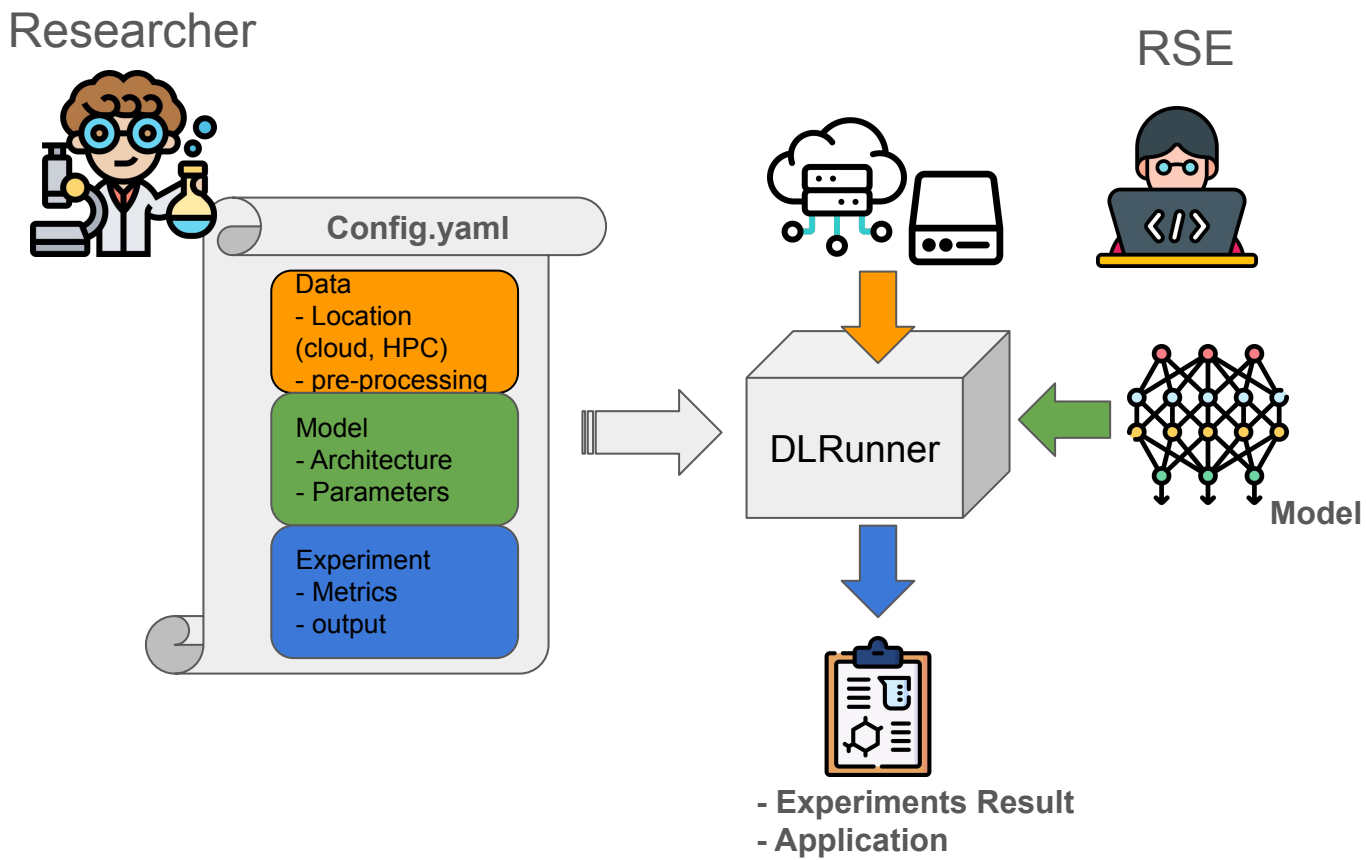
General vision for the project

Develop a DL workflow for the team

- Should be based on **PyTorch** to align with specific geospatial packages (TorchGeo, Xarray)
- Allows to test different parameter **configurations** (kernel size, batch size etc)
- Data processing pipeline (image sliding, cloud etc.)
- Can be repurposed for other projects
- Can be used on our local machine as well as **HPC**

Workflow components

1. **U-Net class** for the model architecture
   - Should handle different kernel sizes
2. **Data loader** for reading and preprocessing images
   - Optional data augmentation
3. **DL “runner”** to glue components together, train and test the model
   - Should handle loading existing weights for finetuning purposes
4. **Configuration** file to serve as runner input
   - Specifies kernel, batch size, padding, epochs etc.



Icons:
https://www.flaticon.com/

Config file example

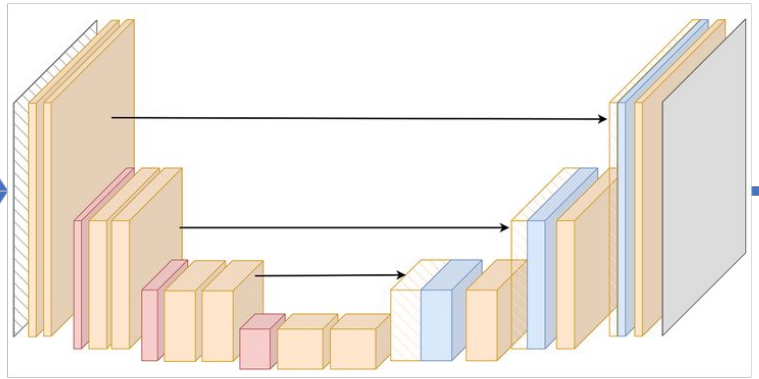
```yaml
experimentname: &experimentname 'unet_ksize_3x3_relu_sweden_unaug'
experimentpath: &experimentpath './experiments'
model:
    unet.UNET:
      pooling: 'max'
      in_channel: 1
      hiddenactivation: 'ReLU'
      encoder_ksize: [3, 3]
      decoder_ksize: [3, 3]
      encoder_padding: 1
      decoder_padding: 1
    loss:
      BCEWithLogitsLoss:
      #MSELoss:
      #HuberLoss:
    #weight:
    opt:
      Adam:
        lr: 0.0001
    epoch: 100
    dataset: 'TrainTestDataSet'
    datasetloader:
      batch_size: 16
      shuffle: True
    accuracymetric: # sklearn.metrics
      threshold: 0.5
      BinaryF1Score:
      # you can add parameter for the function if need
      BinaryAccuracy:
      BinaryPrecision:
      BinaryRecall:
    weightexport:
      path: [ *experimentpath, *experimentname, 'weight' ]
      selection: 'best' # support best or all
      additional_criterion: 'BinaryF1Score'
    experimentresult:
      path: [ *experimentpath, *experimentname, 'result' ]
      exportSample: False
data:
  loader: 'SKImgDataLoader'
  filetype: 'tif'
  training:
    data: '../deep_learning/data/pretraining/training/hpmf'
    label: '../deep_learning/data/pretraining/training/labels'
```

HPC script example

```
#!/bin/bash

#SBATCH -J finetuned_aug_estonia_relu_ks7x7_500ep_bs16_from_train_unaug_sweden_test_unaug_sweden
#SBATCH -N 1
#SBATCH --cpus-per-task=4
#SBATCH -t 72:00:00
#SBATCH --mem=8G
#SBATCH --partition=gpu
#SBATCH --gres=gpu:tesla:1
#SBATCH -o /gpfs/space/home/holgerv/Ditches/working/pytorch_unet/log/%j_%x.out

# Experiment name
experiment=finetuned_aug_estonia_relu_ks7x7_500ep_bs16_from_train_unaug_sweden_test_unaug_sweden

cd ~/Ditches/working/pytorch_unet

rm -rf ./experiments/$experiment/result
rm -rf ./experiments/$experiment/weight

module load python/3.10.10

python DLRunner.py -config ./experiments/$experiment/config.yml
```

Thanks for listening!

Deep learning model



PyTorch

scikit-image
image processing in python