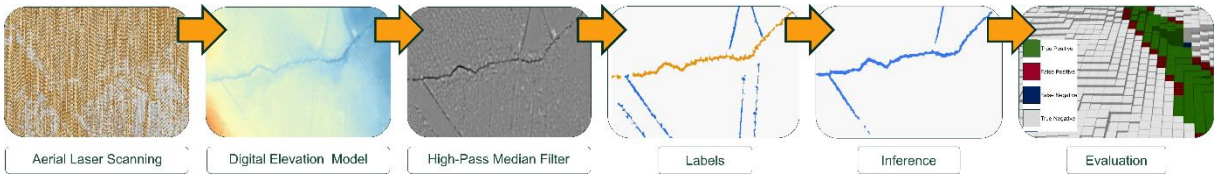
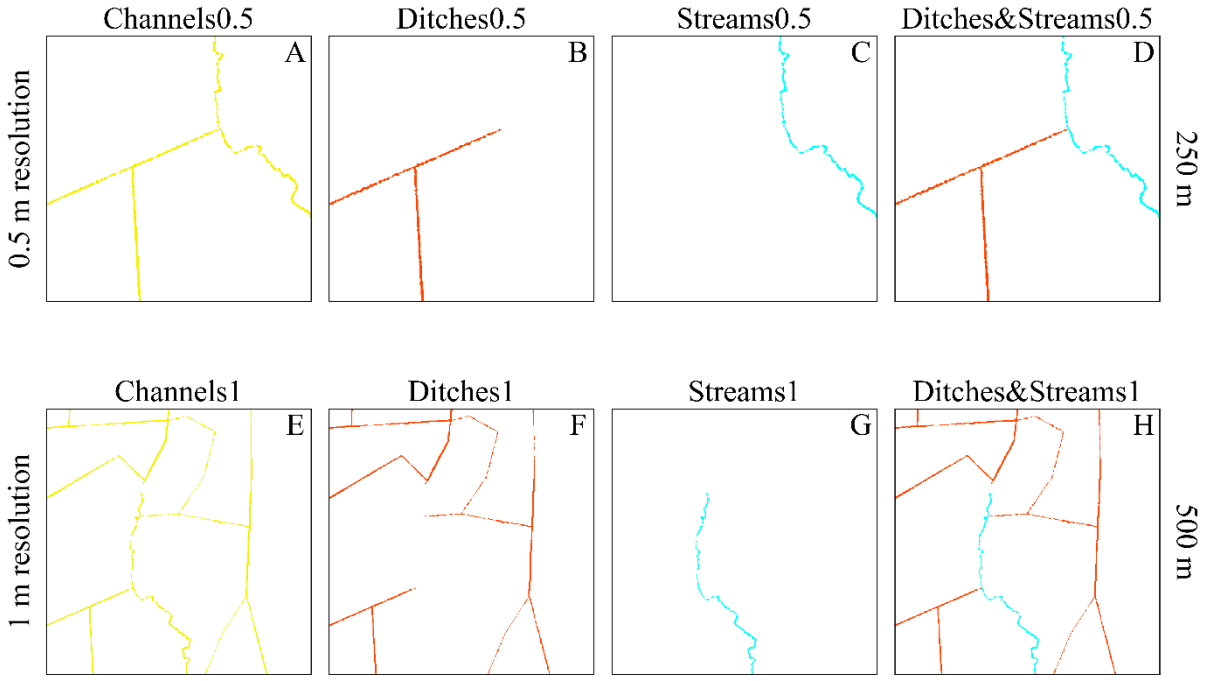
Automatic Detection of Ditches and Natural Streams from Digital Elevation Models Using Deep Learning



Using topographic indices derived from the Swedish Aerial Laser Scanning data, we have trained deep learning models to detect ditches and stream channels. The chosen architecture was UNet. The topographic indices were used individually and in combination.

This repository has scripts to create labels, calculate topographic indices, train and evaluate the models, and apply them to detect the location of channels (inference). It also contains the highest-ranking models for the 0.5 m resolution.

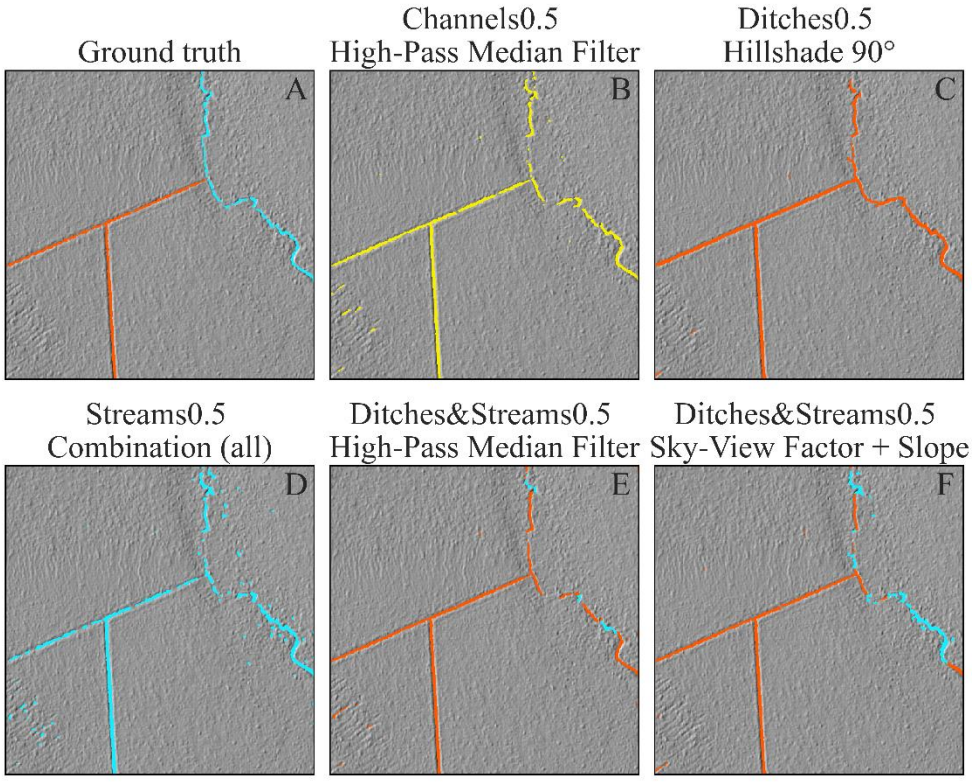
Different datasets were used to train the models and analyze the impact of different labeling methods. The dataset “Channels” combines ditches and streams under the same label. Dataset “Ditches” only has the pixels corresponding to ditches. Dataset “Streams” only contains the pixels that correspond to the stream channels. Dataset “Streams&Ditches” has ditches and streams labeled separately. All of them can be created using the code present in this repository.



This is how the models performed according to their Matthew's Correlation Coefficient (MCC):

Topographic Indices	Channels0.5	Ditches0.5	Streams0.5	Ditches&Streams0.5 (ditches)	Ditches&Streams0.5 (streams)
Combination	0.65	0.61	**0.32**	0.64	0.12
Hillshade 0°	0.63	0.68	0.31	0.57	0.11
Hillshade 45°	0.59	0.60	0.28	0.60	0.27
Hillshade 90°	0.64	**0.69**	0.30	0.65	0.22
Hillshade 135°	0.60	0.63	0.26	0.59	0.25
HPMF	**0.67**	0.67	0.25	**0.69**	0.09
Slope	0.64	0.68	0.13	0.63	**0.28**
Sky-view Factor	0.66	0.63	0.19	0.63	0.17
Sky-view Factor + Slope	-	-	-	**0.74**	**0.31**

These are examples of inference from the models with the highest MCC:

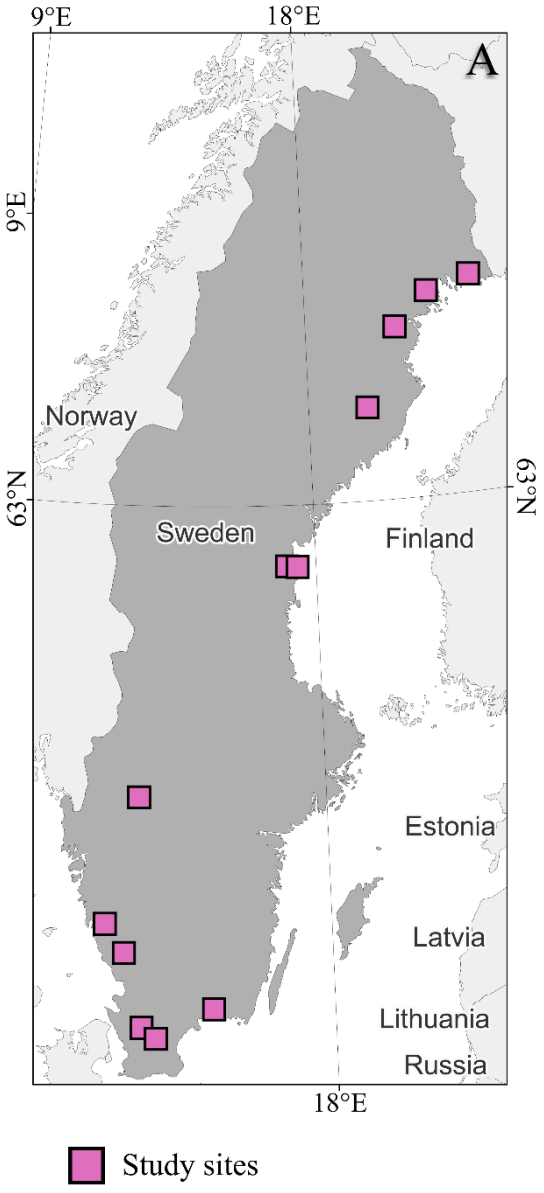


Data

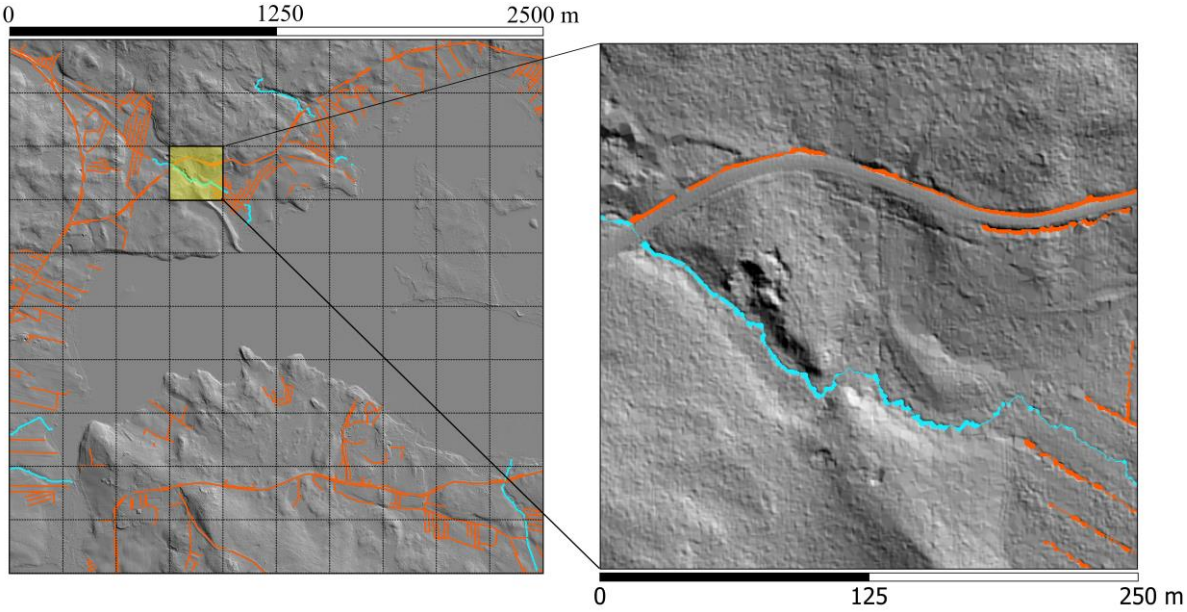
- Channel network as a polyline shapefile

- Aerial Laser Scanning data

The data for this study comes from 12 study areas spread across Sweden, with different characteristics regarding land use and forest cover, among others. The laser data comes from Lantmäteriet, and it was used to create the digital elevation models from which the topographic indices were calculated.



The data is originally organized into tiles of 2500 m x 2500 m, later being split into chips of 250 m x 250 m.



Creating the labels

---

1. laser_to_DEM.py

    Creates the digital elevation model from the aerial laser data.

2. create_rasterlines-py

    Turns the polyline channels from the shapefiles into lines in raster data. The raster pixels can be 0 (background), 1 (ditches), or 2 (streams).

3. separating_channels.py

    Splits the raster data between channel type, creating a copy with only ditches and one with only streams.

4. buffering_raster.py

    Creates a buffer of 1.5 m around each channel in the raster data, to represent the mean channel width of 3 m.

5. calculating_hpmf.py

    Calculates the High-Pass Median Filter from the digital elevation model.

6. lessthan_reclassification.py

Reclassifies the HPMF values based on the threshold of -0.075, with 0 for values above it, and 1 for those below it.

7. multiplying_rasters.py

Multiplies the reclassified rasters and the buffered ones, and outputs the pixels that are within the buffer zone.

8. majority_filtering.py

Smooths the multiplied output.

9. combining_rasters_finaloutput.py

Combines the rasters with the separated ditches and streams into a single one.

10. dataset_channels_labels.py

Creates the dataset Channels.

11. dataset_ditches_labels.py

Creates the dataset Ditches.

12. dataset_streams_labels.py

Creates the dataset Streams.

Topographic indices

- calculating_hillshade.py

Calculates the hillshade from the digital elevation model.

- calculating_slope.py

Calculates the slope from the digital elevation model.

- calculating_svf.py

Calculates the Sky-view Factor

Creating the input chips

---

1. splitting_rasters.py

   Splits the tiles into chips to be used as training data.

2. selecting_labeled_chips_by_threshold.py

   Selects the dataset chips that are over the established threshold.

3. selecting_ti_chips.py

   Selects the chips of topographic indices based on the previously selected dataset chips.

4. splitting_training_data.py

   Splits the dataset and topographic indices chips between training (80%) and testing (20%).

Semantic segmentation

- train.py
- evaluate.py
- inference_unet.py