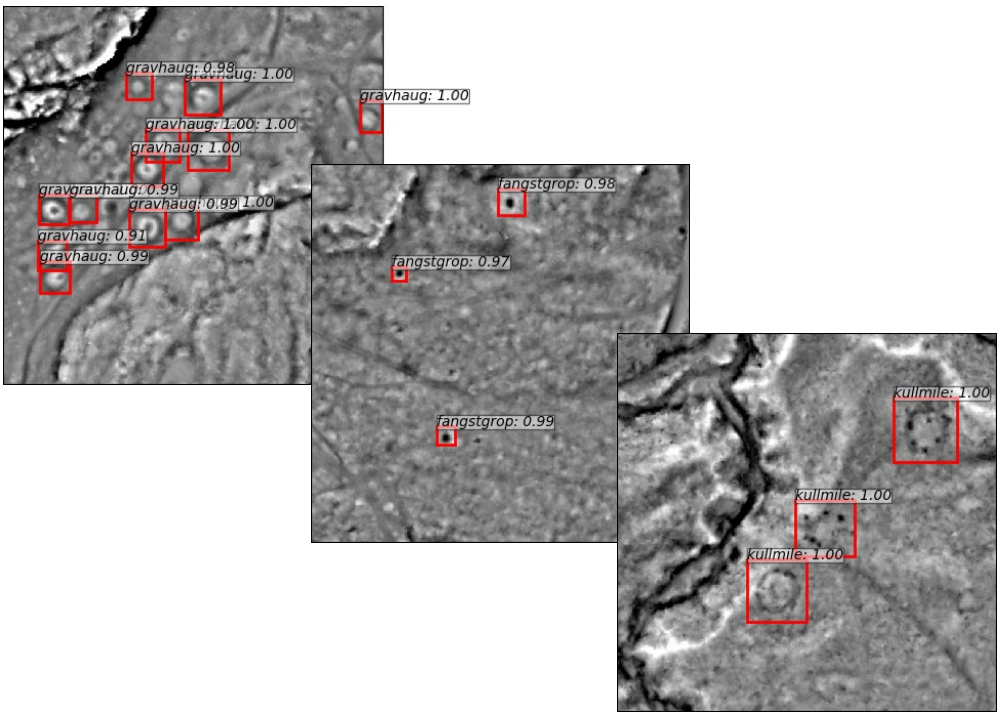
Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Note

# Automated detection of cultural heritage in airborne lidar data

## CultSearcher operationalisation





**Note no.** SAMBA/50/19

**Authors** Øivind Due Trier, Jarle Hamar Reksten

**Date** 30 December 2019

**Authors**
Øivind Due Trier is senior research scientist at the Norwegian Computing Center, Section for Earth Observation.

Jarle Hamar Reksten is senior research scientist at the Norwegian Computing Center, Section for Earth Observation.

**Norsk Regnesentral**
Norsk Regnesentral (Norwegian Computing Center, NR) is a private, independent, non-profit foundation established in 1952. NR carries out contract research and development projects in information and communication technology and applied statistical-mathematical modelling. The clients include a broad range of industrial, commercial and public service organisations in the national as well as the international market. Our scientific and technical capabilities are further developed in co-operation with The Research Council of Norway and key customers. The results of our projects may take the form of reports, software, prototypes, and short courses. A proof of the confidence and appreciation our clients have in us is given by the fact that most of our new contracts are signed with previous customers.

**Norsk Regnesentral**
Norwegian Computing Center
Postboks 114, Blindern
NO-0314 Oslo, Norway

**Besøksadresse**
*office address*
Gaustadalleen 23a
NO-0373 Oslo, Norway

**Telefon** · telephone
(+47) 22 85 25 00
**Telefaks** · telefax
(+47) 22 69 76 60

**Bankkonto** · bank account
8200.01.48888
**Org.nr.** · enterprise no.
NO 952125001 VAT

**Internett** · internet
www.nr.no
**E-post** · e-mail
nr@nr.no

| | |
|---|---|
| **Title** | **Automated detection of cultural heritage in airborne lidar data** |
| **Authors** | **Øivind Due Trier, Jarle Hamar Reksten** |
| Quality assurance | Rune Solberg |
| Date | 30 December 2019 |
| Year | 2019 |
| Publication number | SAMBA/50/19 |


**Abstract**

The goal of this research was to develop automated tools for improving the cultural heritage mapping in Norway, thus enabling detailed mapping of large areas within realistic budgets and time frames.

Preprocessing and detection methods were integrated into a python script that may be called from QGIS or started from the Linux command line. The input was a collection of LAS files, and the output was two ESRI shape files for each object type; centre points in one file and object outlines in another file. The software was installed at the Directorate for Cultural Heritage in Norway (Riksantikvaren).

The best classification performance was 86% correct classification (consumer's accuracy), i.e., how many of the true cultural heritage objects were correctly predicted by the method. This was obtained on a test set of labelled lidar data not seen during training. At the same time, the producer's accuracy was 3%, i.e., how many of the objects predicted by the method were in fact true cultural heritage objects. Thus, the main potential for improvement is in reducing the large number of false predictions, i.e., increasing the producer's accuracy. This should be the focus for future improvements of the detection method.



| | |
|---|---|
| Keywords | Airborne laser scanning, grave mounds, deer hunting systems, charcoal kilns, deep learning, artificial intelligence |
| Target group | Researchers, archaeologists |
| Availability | Open |
| Project number | 220 878 CultSearcher |
| Research field | Remote sensing, archaeology |
| Number of pages | 157 |

# Table of Content

# List of figures

# 1 Introduction

The goal of this research was to develop automated tools for improving the cultural heritage mapping in Norway, thus enabling detailed mapping of large areas within realistic budgets and time frames.

The existing cultural heritage mapping in Norway is incomplete. Some selected areas are mapped well, while the majority of areas only contain chance discoveries, often with bad positional accuracy.

The Norwegian Computing Center has previously developed automated methods for detecting some types of cultural heritage objects from airborne laser scanning (ALS) data (Trier and Pilø 2012; Trier, Zortea and Tonning 2015; Trier, Pilø and Johansen 2015; Trier, Salberg and Pilø 2018; Trier, Cowley and Waldeland, 2019). These have contributed to increasing the number of areas that are mapped well. However, the methods have a number of issues that have prevented them from being used systematically on all available ALS datasets.

All of Norway will soon be covered by ALS data for the purpose of creating a new national elevation model. The Directorate for Cultural Heritage in Norway (Riksantikvaren) wants to use this opportunity to obtain a more complete and accurate mapping of cultural heritage in the landscape. The focus is on Iron Age grave mounds and deer hunting systems, as these are automatically protected by Norwegian law due to their age. The automatic protection by law applies to such monuments even if they are not yet mapped. This is, however, at the risk of the monuments being unintentionally destroyed due to the lack of knowledge of their existence.

The following challenges were identified:

1. develop an automated processing chain,
2. reduce processing time
3. reduce the number of false positives and false negatives
4. develop detection methods that may be applied on all Norwegian landscapes.

A recent development in deep neural networks for object detection in natural images is the region-proposing convolutional neural network (R-CNN; Girshick et al., 2014), which may also be used for cultural heritage detection in ALS data. Verschoof-van der Vaart and Lambers (2019) use Faster R-CNN (Ren et al., 2017) to detect prehistoric barrows and Celtic fields in ALS data from the Netherlands.

He et al. (2017) extend Faster R-CNN into Mask R-CNN by providing, for each detected object, an object mask in addition to the bounding box provided by Faster R-CNN.

# 2 Installation at Riksantikvaren

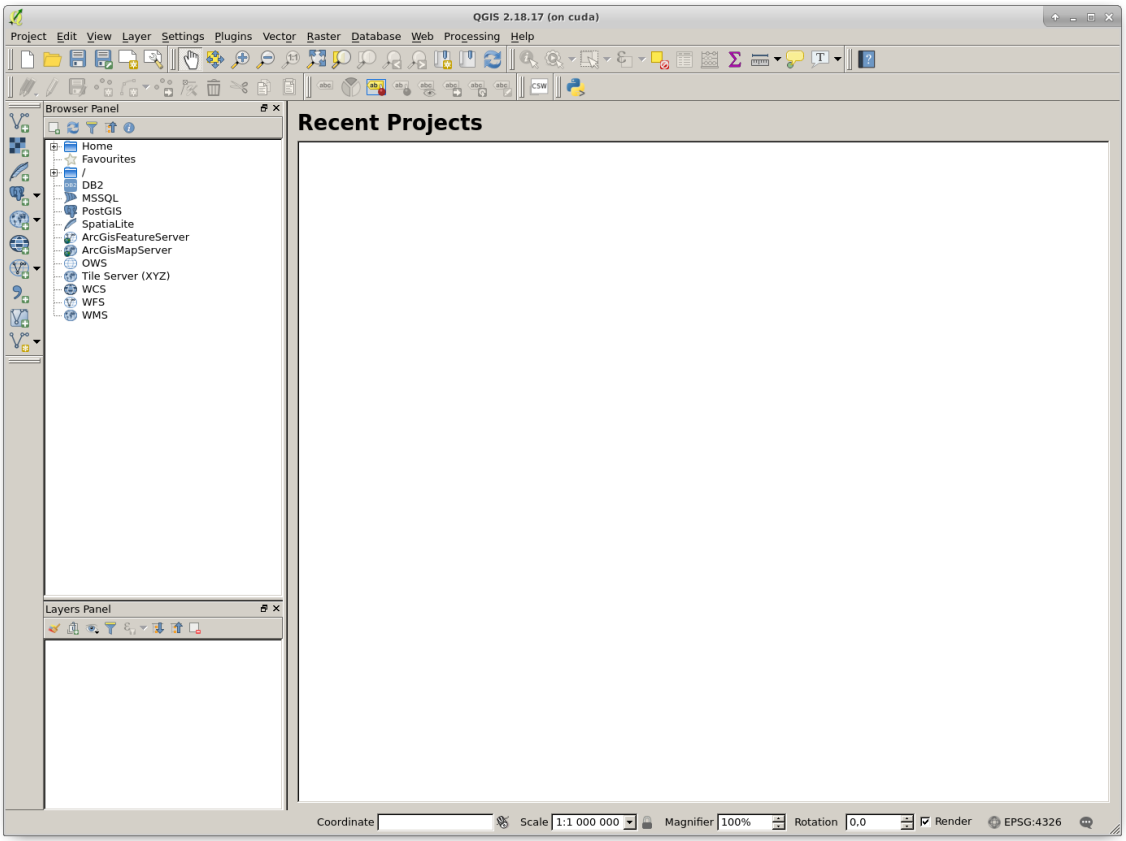
## 2.1 How to run

```
trier@cuda:~$ cd /opt/nr/cultsearcher/gui

trier@cuda:/opt/nr/cultsearcher/gui$ qgis
```

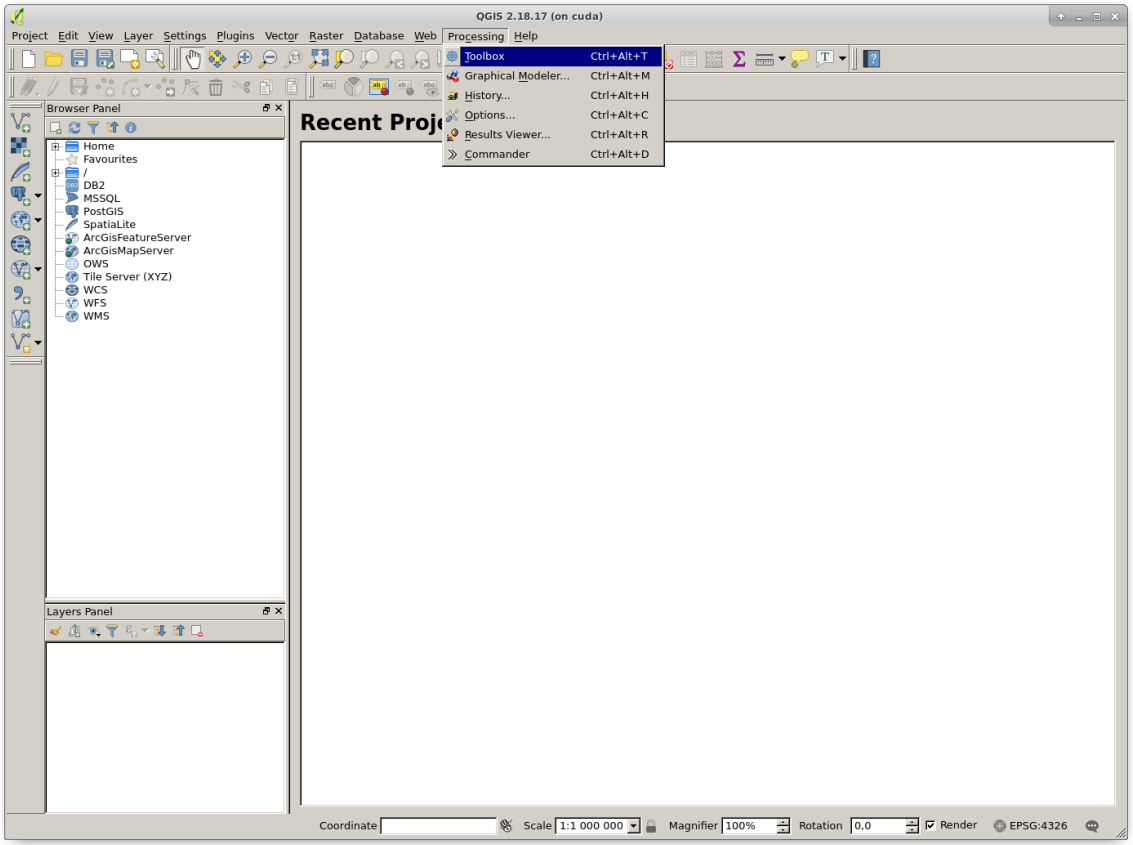


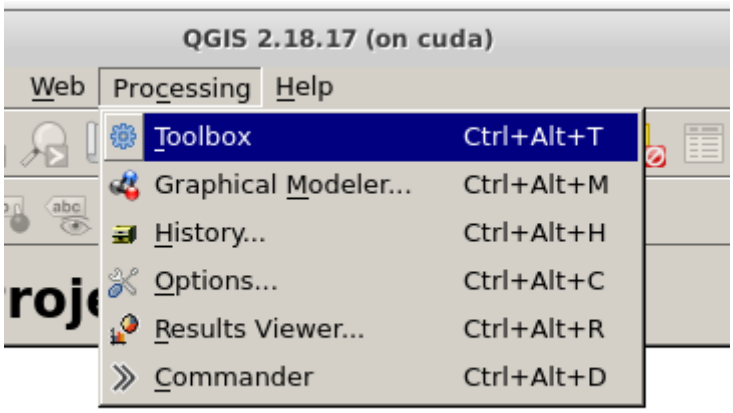
Click ‘OK’.

QGIS starts.

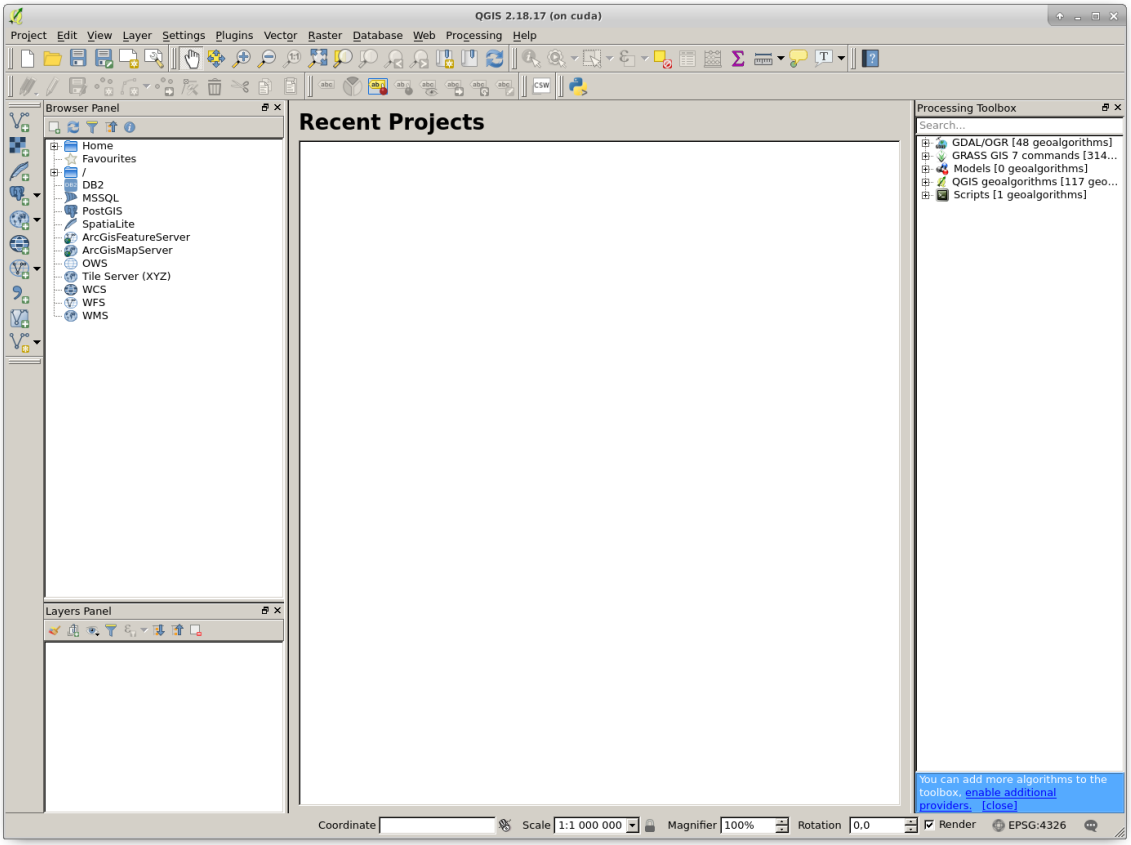

QGIS 2.18.17 (on cuda)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
Recent Projects
Coordinate
Scale 1:1 000 000
Magnifier 100%
Rotation 0,0
Render
EPSG:4326

In the menu bar, select ‘Processing’, ‘Toolbox’

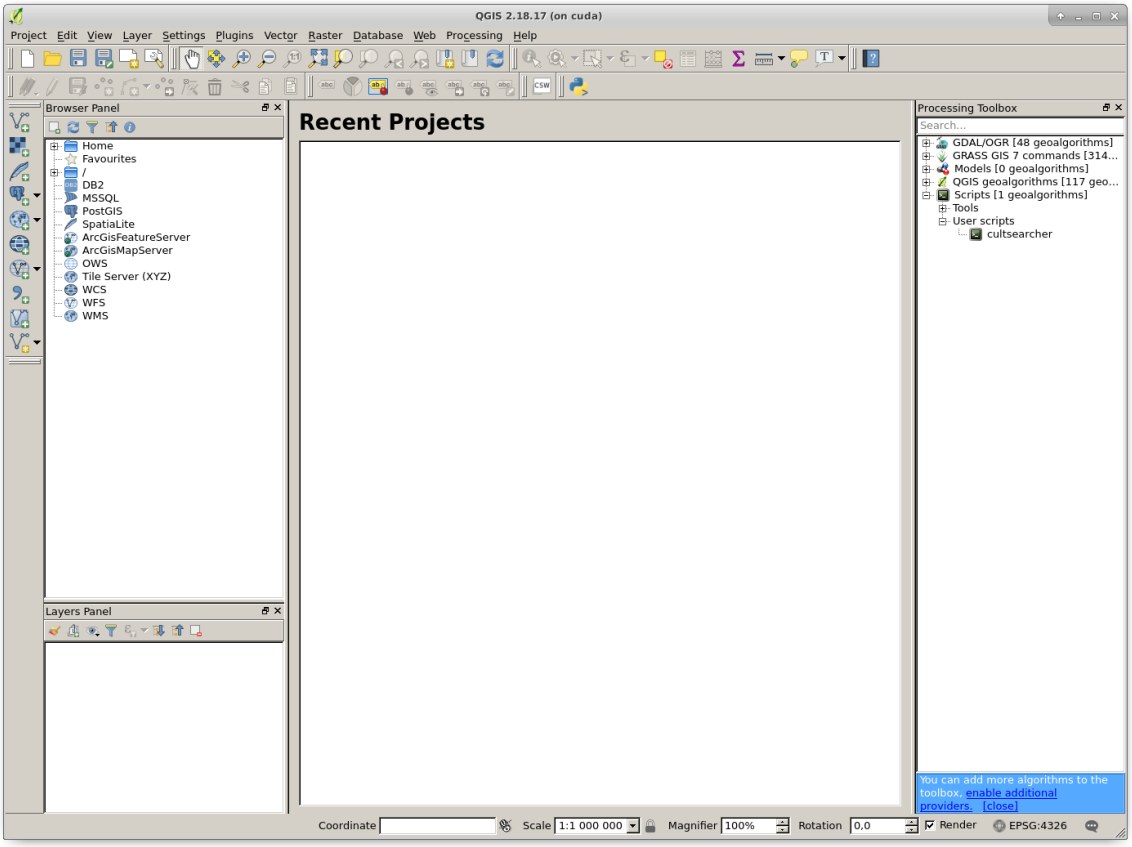


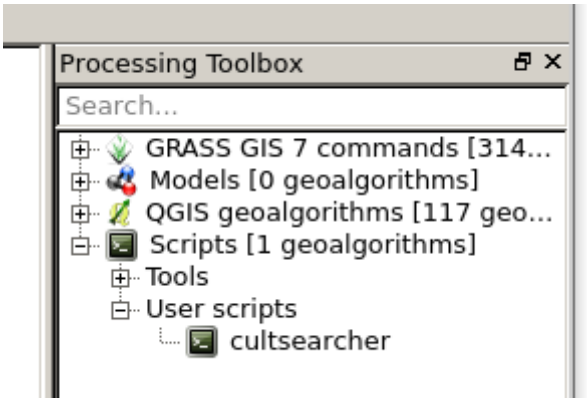



The processing toolbox is now displayed on the right hand side.




In the processing toolbox, expand ‘scripts’ by clicking on the ‘+’. Also expand ‘user scripts’. The script ‘cultsearcher’ should appear.

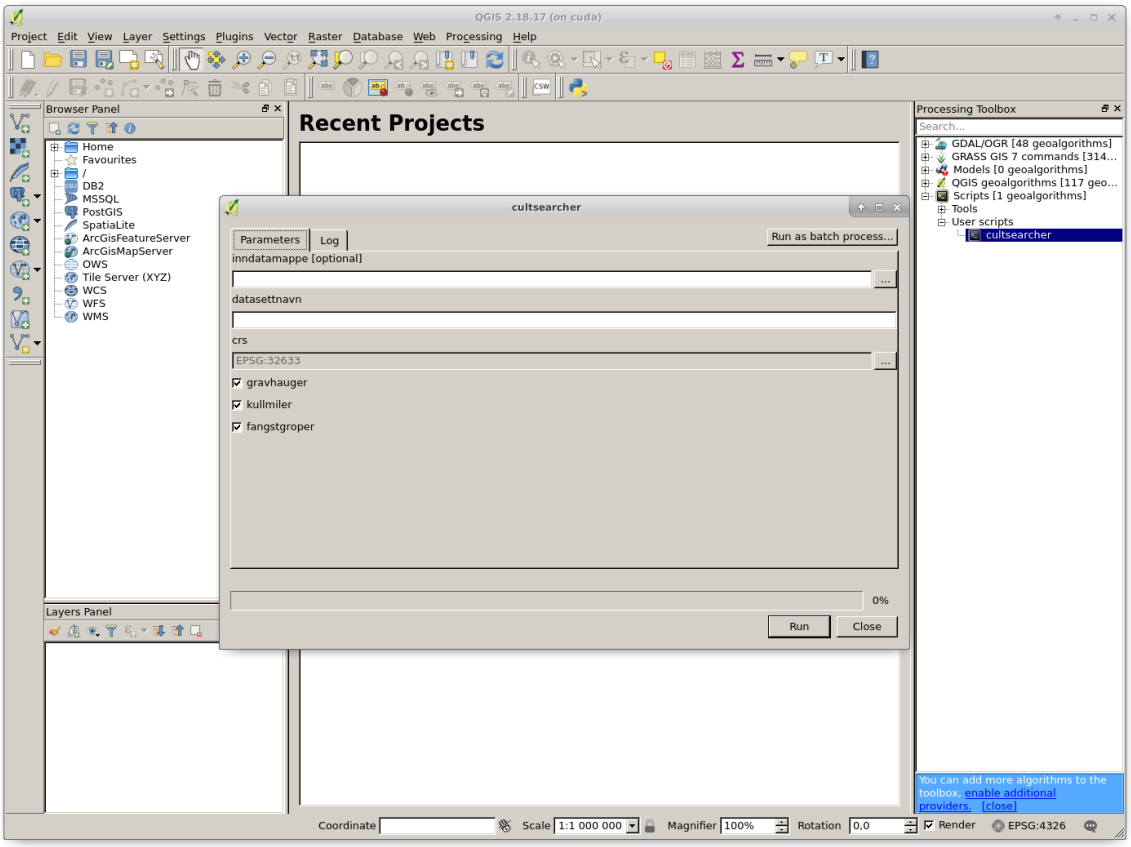



If ‘cultsearcher’ does not appear, then see section 2.3 for an explanaition on how to add ‘cultsearcher’ as a user script.

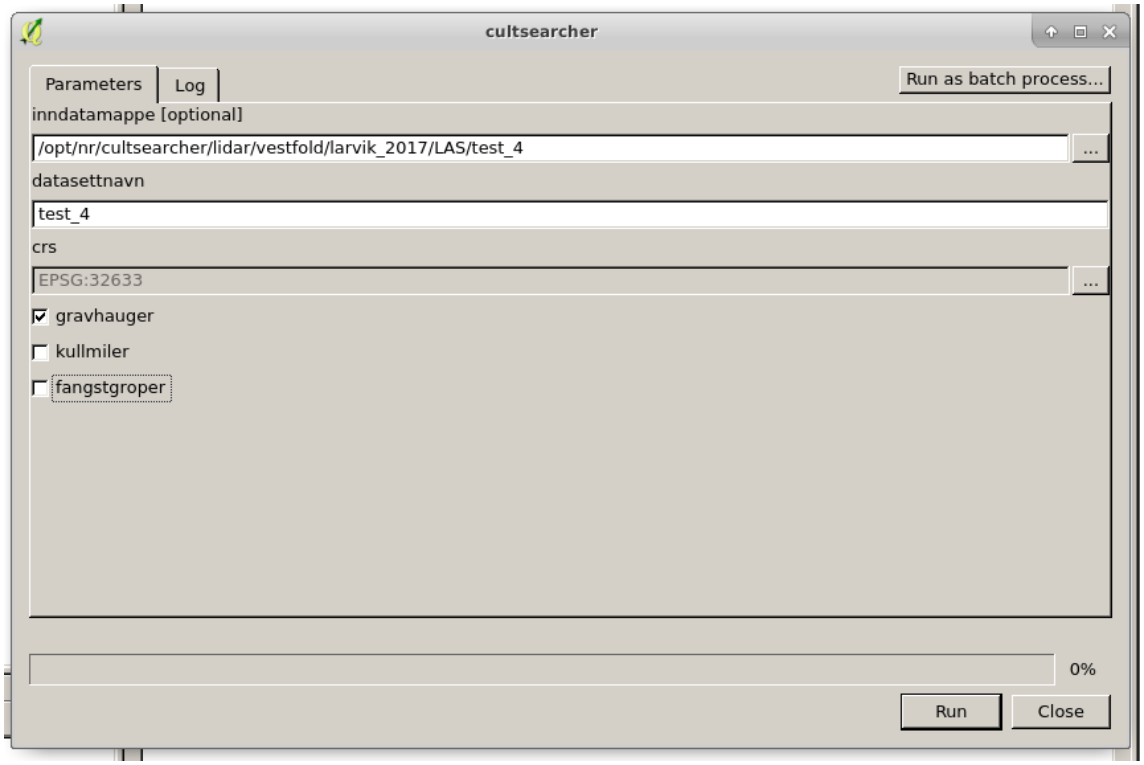
Double-click on ‘cultsearcher’.

A dialog window appears.

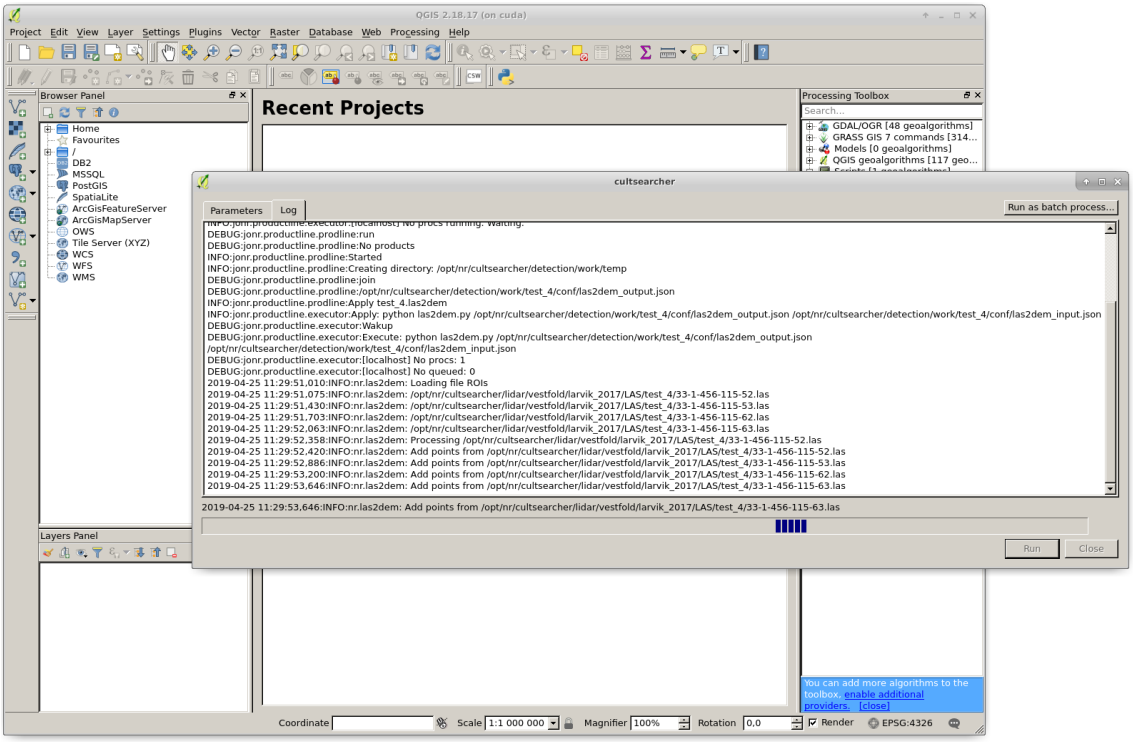



Select a folder containing las or laz files.

Also, type a name for the dataset, and select the object types to detect.

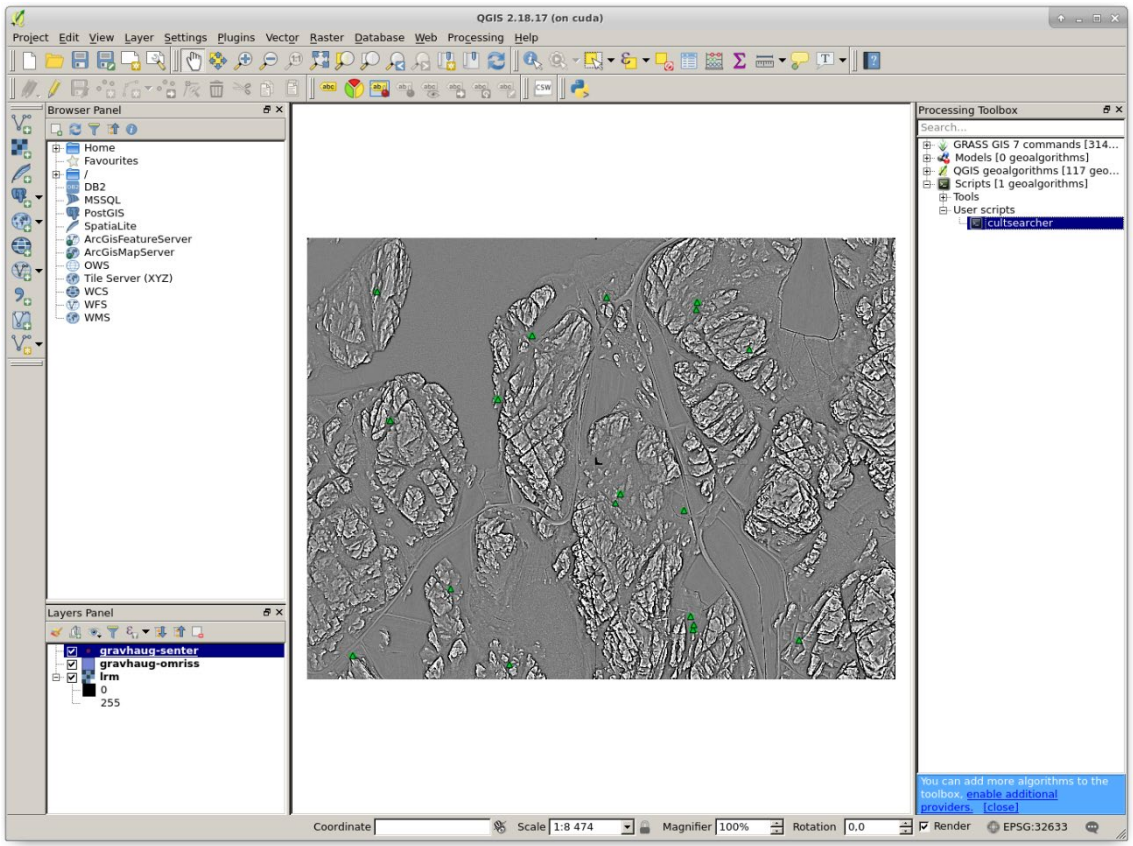


Click ‘run’. The processing starts.




The conversion from las (or laz) files to raster files is quite slow. The remaining processing steps are faster.

When the processing has finished, predicted locations for cultural heritage objects are displayed as vector layers. There is one centre point layer and one outline polygon layer for each object type. The raster background layer is a local relief model (LRM).



## 2.2 Short demo of detection module

In order to quickly visualise the detection capabilities of the neural network, here is a short demo that compares detection results with the image annotations. The demo runs on the test images, i.e., images that have not been used in training of the neural network parameters.

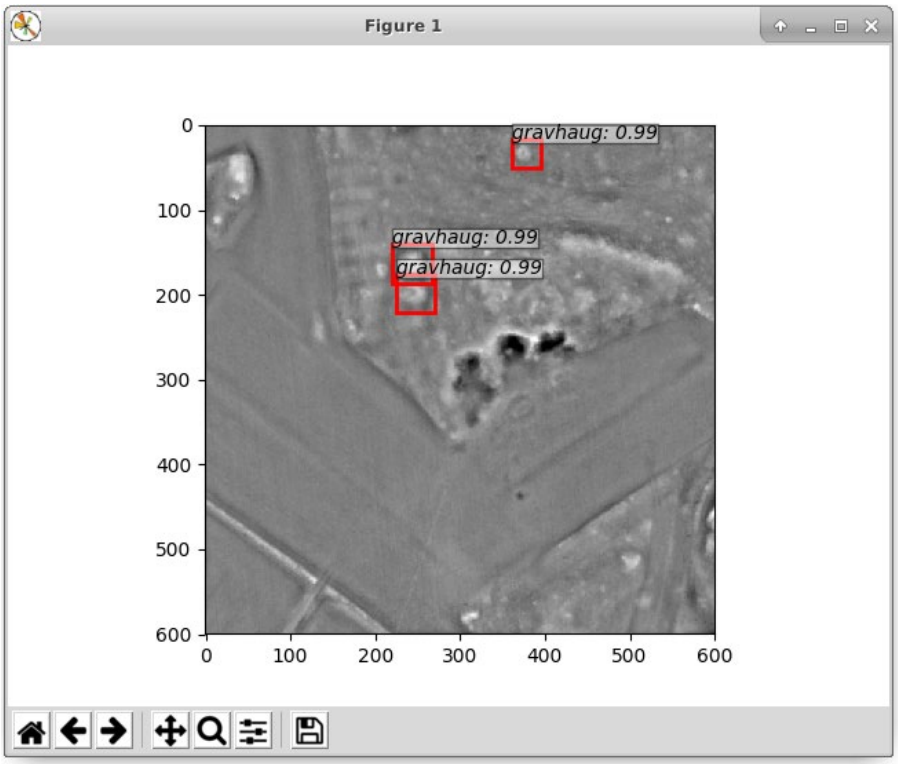
```
trier@cuda:/opt/nr/cultsearcher$ cd

trier@cuda:~$ cd /opt/nr/cultsearcher

trier@cuda:/opt/nr/cultsearcher$ source .env/bin/activate

(.env) trier@cuda:/opt/nr/cultsearcher$ cd simple-faster-rcnn-
pytorch-master
(.env) trier@cuda:/opt/nr/cultsearcher/simple-faster-rcnn-
pytorch-master$ python demo_2_test.py
```

Here are the same commands without the leading prompts:

```
cd

cd /opt/nr/cultsearcher

source .env/bin/activate

cd simple-faster-rcnn-pytorch-master

python demo_2_test.py
```

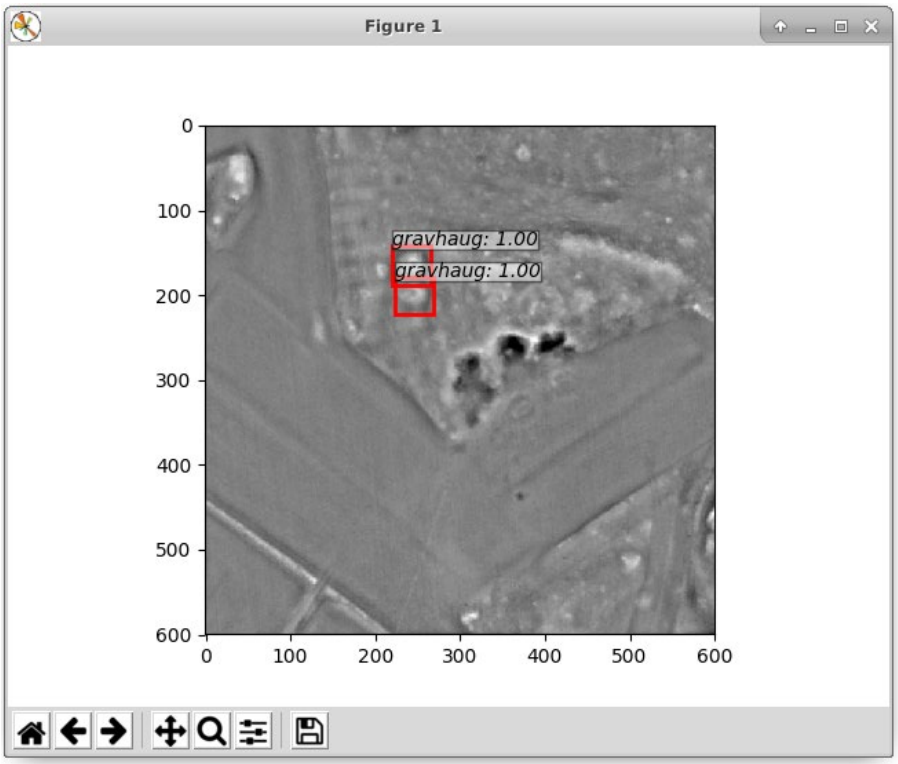
**Note**: there is a Boolean variable, `visualize_each_image`, in the source code. Make sure that `visualize_each_image` is set to `True` if you want to visualize each image, as shown in the figures below. However, if you want to quantify how well the detection module works on the test data, set it to `False`.

The demo shows one image with predicted object locations.

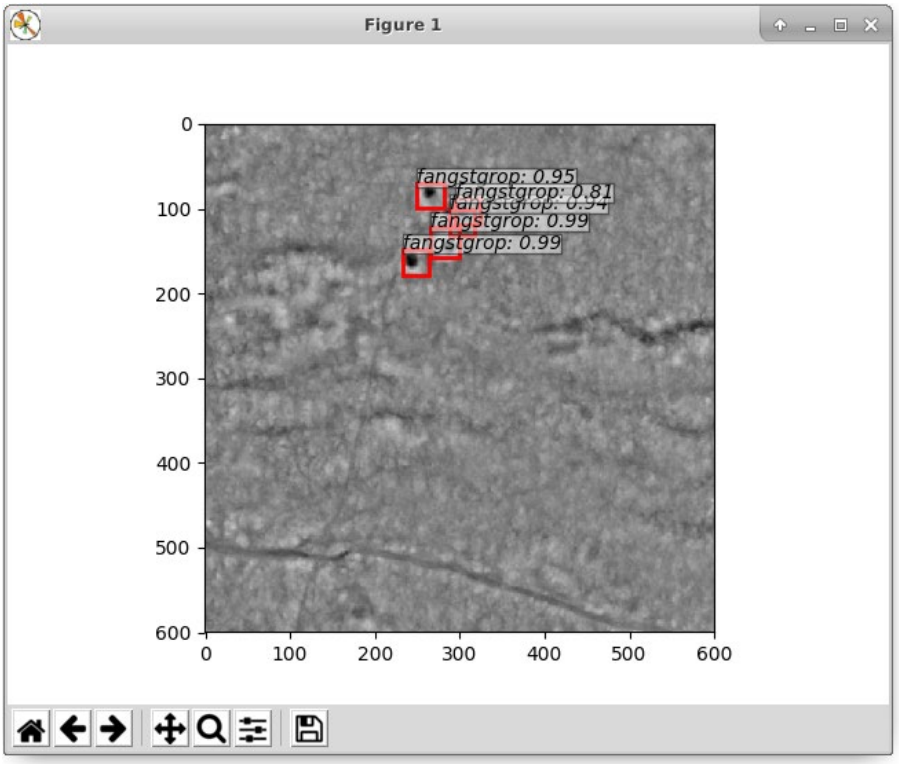


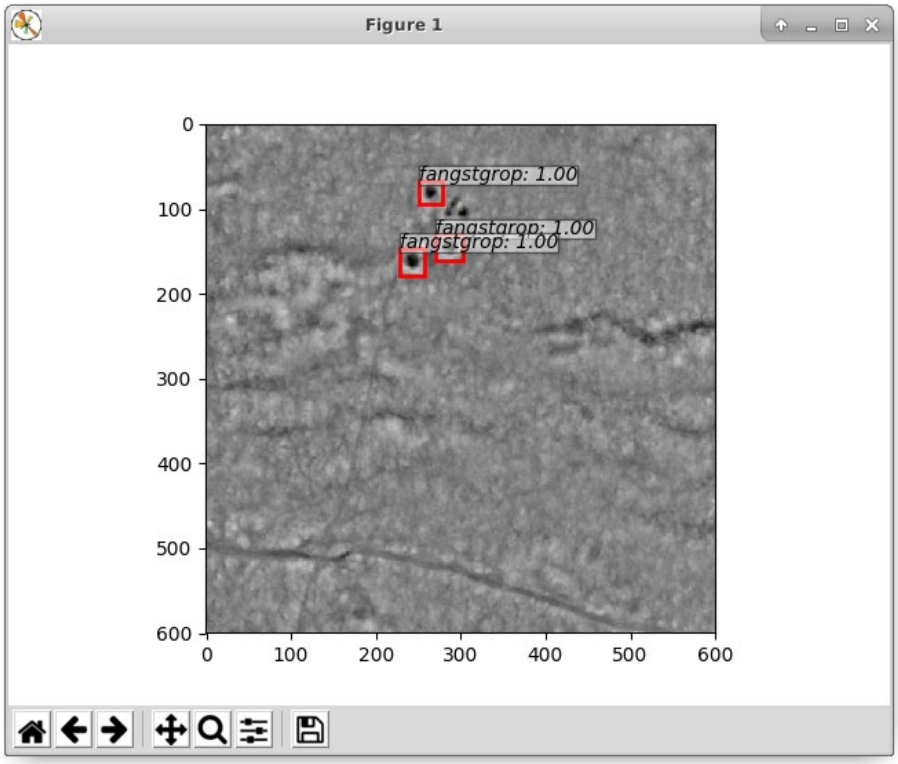

Close the window by clicking the ‘x’ in the upper right corner

The demo shows the same image with the presumed correct object locations.




Close the window, and the next image appears with predicted object locations.

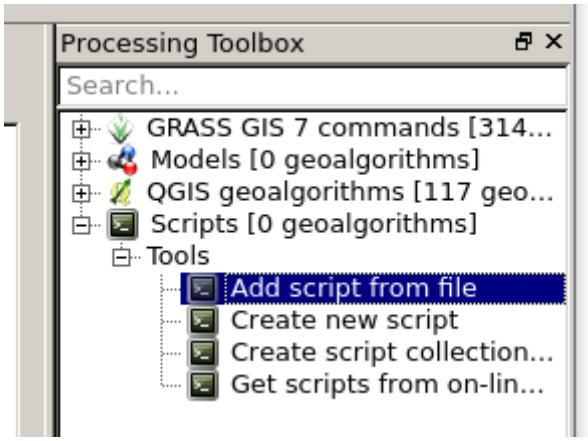




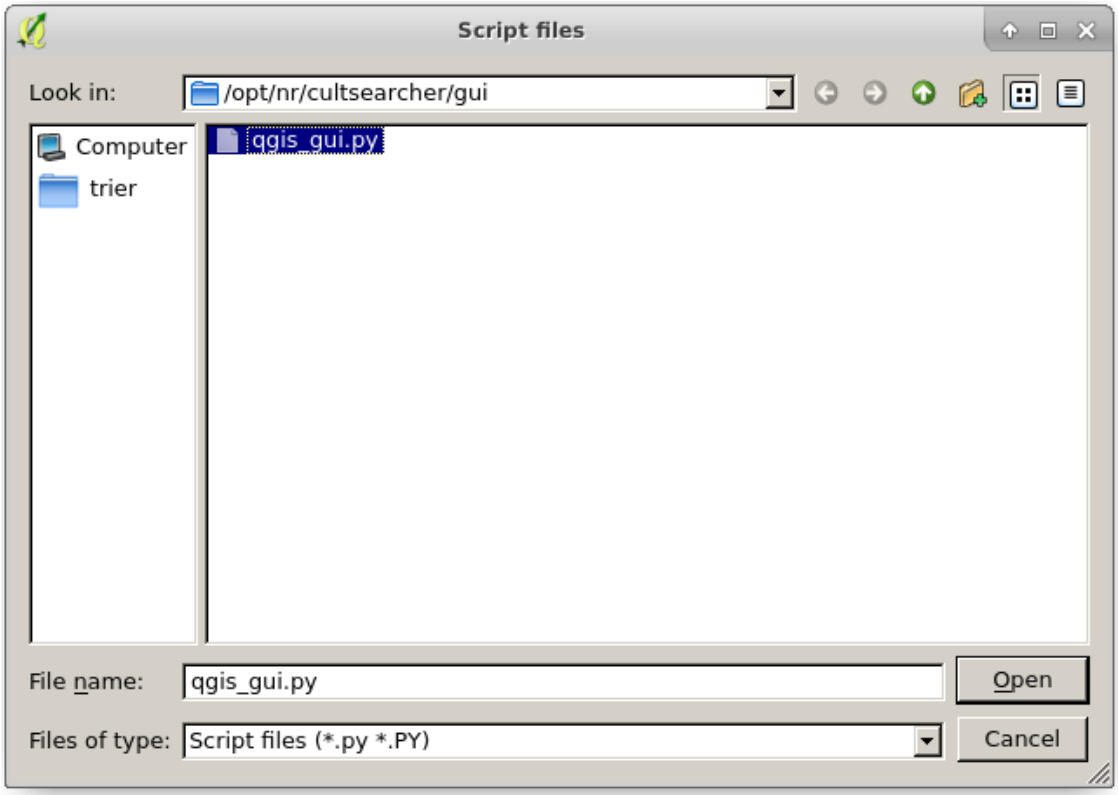

## 2.3 How to add cultsearcher as a user script

Follow the steps shown in the figures below to add `qgis_gui.py` as a user script.

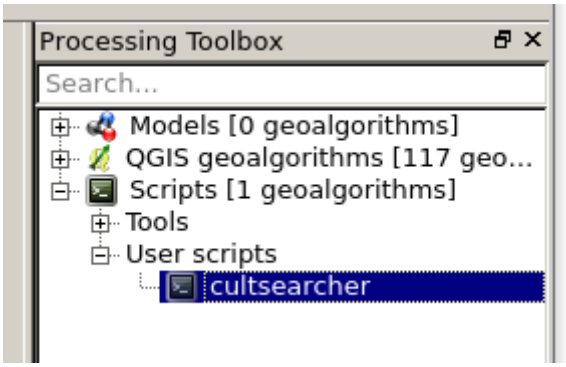
In the processing toolbox, double-click on ‘add script from file’.




In the dialog box that appears, select qgis_gui.py and click ‘open’.



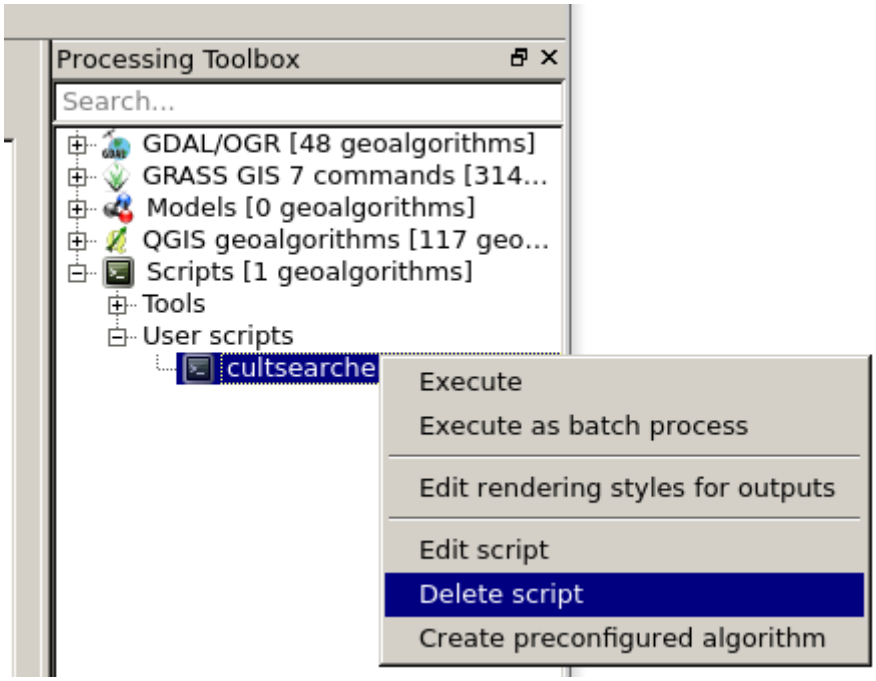

The script ‘cultsearcher’ now appears in the processing toolbox:




## 2.4 What to do if the source code file `qgis_gui.py` has been changed

If the source code in the file `qgis_gui.py` has been changed, then the script must be deleted from QGIS and then added again.

Right-click on ‘cultsearcher’ and select ‘delete script’.




Then add the script again as described above.

## 2.5 Directory structure

Source code, input data and processing results are located in various subfolders under `/opt/nr/cultsearcher/` as follows.

`/opt/nr/cultsearcher/lidar/` contains LAS files which are used as input data.

`/opt/nr/cultsearcher/gui/` contains the QGIS user script `qgis_gui.py`.

`/opt/nr/cultsearcher/detection/results/` contains processing results. There is one subfolder for each dataset. For each dataset, there is one subfolder `detections` with ESRI shape files, and one subfolder `lrm` with raster TIFF files.

`/opt/nr/cultsearcher/detection/work/` contains intermediate processing results. These may be deleted to save disk space.

`/opt/nr/cultsearcher/simple-faster-rcnn-pytorch-master/` contains source code.

`/opt/nr/cultsearcher/simple-faster-rcnn-pytorch-master/checkpoints/` contains neural network parameters learned during training.

`/opt/nr/cultsearcher/.env/` contains external python packages that have been installed.

`/opt/nr/cultsearcher/imgdataset/` contains training, validation and test images. Each image is 600 x 600 pixels in size and has annotations in the form of locations of known cultural heritage objects that are visible in the image.

# 3 Data

ALS point cloud datasets (Table 1, Table 2) were downloaded from http://hoydedata.no. This internet site provides free access to all ALS data in Norway.

Table 1. ALS datasets used for method development and evaluation.

| dataset | ALS project name in hoydedata.no | point density | object type |
|---|---|---|---|
| Larvik 2017 | NDH Larvik 5pkt 2017 | 5/m$^2$ | grave mound |
| Horten 2016 | NDH Vestfold 5pkt 2016 | 5/m$^2$ | grave mound |
| Hå Jæren 2017 | NDH Jæren-Randaberg-Sola 5pkt 2017 | 5/m$^2$ | grave mound |
| Oppdal Vang 2011 | Oppdal 12pkt 2011 | 12/m$^2$ | grave mound |
| Sarpsborg 2015 | NDH Østfold 5 pkt 2015 | 5/m$^2$ | grave mound |
| Steinkjer 2011 | Steinkjer 2011 | 1/m$^2$ | grave mound |
| Steinkjer 2017 | NDH Steinkjer 5pkt 2017 | 5/m$^2$ | grave mound |
| Brumunddal 2016 | NDH Brumunddal 5pkt 2016 | 5/m$^2$ | grave mound |
| Olstappen 2010 | Olstappen 2010 | 10/m$^2$ | pitfall trap |
| Dovre 2011 | Dovre 2011 | 5/m$^2$ | pitfall trap |
| Dovre Grimsdalen 2010 | Grimsdalen 2010 | 12/m$^2$ | pitfall trap |
| Nordfron 2012 | Midt-Gudbrandsdalen 2012 | 5/m$^2$ | pitfall trap |
| Vågå 2018 | NDH Vågå-Lom-Skjåk 5pkt 2018 | 5/m$^2$ | pitfall trap |
| Nordfron 2017 | NDH Ringebu-Fron-Gausdal 5pkt 2017 | 5/m$^2$ | pitfall trap |
| Nordfron 2018 | NDH Ringebu-Fron-Gausdal 5pkt 2018 | 5/m$^2$ | pitfall trap |
| Nordfron Venabu 2018 | NDH Venabu 5pkt 2018 | 5/m$^2$ | pitfall trap |
| Dovre 2013 | Nord-Gudbrandsdalen 2013 | 5/m$^2$ | pitfall trap |
| Dovre 2017 | NDH Lesja-Vågå 5pkt 2017 | 5/m$^2$ | pitfall trap |
| Dovre Folldal 2018 | NDH Folldal 5pkt 2018 | 5/m$^2$ | pitfall trap |
| Nordfron 2013 | Nord-Gudbrandsdalen 2013 | 5/m$^2$ | pitfall trap |
| Lesja 2013 | Nord-Gudbrandsdalen 2013 | 5/m$^2$ | charcoal kiln |

Table 2. ALS datasets used for new archaeological mapping.

| dataset | ALS project name in hoydedata.no | point density |
|---|---|---|
| Øvre Eiker 2015 | Drammen Eiker 2015 | 5/m$^2$ |
| Øvre Eiker Flesberg 2017 | NDH Flesberg-Rollag-Øvre Eiker 5pkt 2017 | 5/m$^2$ |
| Øvre Eiker Modum 2017 | NDH Modum-Sigdal 5pkt 2017 | 5/m$^2$ |

For all the ALS datasets in Table 1, vector maps of known locations of grave mounds, pitfall traps and charcoal kilns were provided as ESRI shape files. The vector maps of grave mounds and pitfall traps were provided by the Directorate for Cultural Heritage in Norway. The vector maps of charcoal kiln locations were provided by Oppland County

Administration. All the vector data were visually checked against the ALS data. Objects with no visual appearance in the ALS data were removed, while obvious omissions were added. For the Oppdal Vang 2011 dataset, small grave mounds were removed from the vector data.

The ALS datasets in Table 2 were selected to cover Øvre Eiker municipality, Buskerud County. Øvre Eiker had few known cultural heritage object locations, but had potential for a large number of previously unknown cultural heritage objects and also local interest in the municipality administration for detailed archaeological mapping. Combined, the three ALS datasets cover the entire area of the municipality.

## 3.1 Subdivision of labelled data into training, validation and test

The data in Table 1 were split into three parts, named 'training', 'validation', and 'test' (Table 3). The neural network parameters would be learned from the training data iteratively by minimising a loss function. The validation data would be used to select the best set of neural network parameters. The test data would then be used to estimate detection performance on data not seen during training.

On average, 69%, 20% and 11% of the known objects were included in the training, validation and test sets, respectively (Table 4). The splitting followed the below principles:

1. For each cultural heritage object type, one ALS dataset contributed to each of the three parts: training, validation and test. The three parts from the same ALS dataset were geographically disjoint, i.e., non-overlapping. This splitting strategy was used on Lesja 2013 (charcoal kilns), Larvik 2017 (grave mounds) and Nordfron Olstappen 2010 (pitfall traps).

2. Each of the remaining ALS datasets was assigned to either training, validation or test.

3. The known cultural heritage objects should be split into training, validation and test with approximately 70% in training, 20% in validation and 10% in test.

Thus, there was a spread in representativeness of the training and validation sets with respect to the test set.

One purpose of the splitting was to obtain realistic estimates on how the detection performance may be on unlabelled ALS datasets, which is the expected situation when doing detailed archaeological mapping. Another purpose was to obtain a sufficient amount of representative training data for tuning of the parameters of the deep neural network. A third purpose was to reduce the chances of overfitting of the neural network parameters. Overfitting means that the neural network performs well on data that are similar to the training data but performs badly on other data.

Table 3. Subdivision of ALS datasets into training, validation and test sets.

| object type | dataset | subset | object count | extent of dataset in UTM zone 33 N | | | |
|---|---|---|---|---|---|---|---|
| | | | | west | east | south | north |
| charcoal kiln | Lesja 2013 | training | 773 | 160 800 | 172 000 | 6 907 200 | 6 916 800 |
| | | validation | 190 | 154 400 | 172 000 | 6 902 400 | 6 919 600 |
| | | test | 95 | 144 800 | 154 400 | 6 916 000 | 6 922 800 |
| grave mound | Brumunddal 2016 | test | 73 | 260 000 | 283 200 | 6 736 200 | 6 774 600 |
| grave mound | Horten 2016 | training | 38 | 238 400 | 243 200 | 6 588 000 | 6 593 400 |
| grave mound | Hå Jæren 2017 | training | 84 | -44 000 | -36 000 | 6 531 800 | 6 545 400 |
| grave mound | Larvik 2017 | training | 288 | 206 400 | 220 800 | 6 547 200 | 6 563 400 |
| | | validation | 165 | 204 800 | 219 200 | 6 565 200 | 6 596 400 |
| | | test | 57 | 220 800 | 226 400 | 6 552 600 | 6 565 800 |
| grave mound | Oppdal Vang 2011 | training | 219 | 224 690 | 225 600 | 6 951 850 | 6 952 925 |
| grave mound | Sarpsborg 2015 | validation | 48 | 274 400 | 284 600 | 6 565 200 | 6 583 800 |
| grave mound | Steinkjer 2011 | validation | 30 | 321 600 | 348 800 | 7 087 800 | 7 113 000 |
| grave mound | Steinkjer 2017 | validation | 44 | 322 400 | 345 600 | 7 097 400 | 7 119 600 |
| pitfall trap | Dovre 2011 | training | 650 | 192 000 | 218 400 | 6 885 600 | 6 915 000 |
| pitfall trap | Dovre 2013 | test | 29 | 190 400 | 204 000 | 6 878 400 | 6 897 000 |
| pitfall trap | Dovre 2017 | test | 15 | 190 400 | 196 800 | 6 882 000 | 6 897 000 |
| pitfall trap | Dovre Folldal 2018 | test | 3 | 233 600 | 234 400 | 6 891 600 | 6 892 200 |
| pitfall trap | Dovre Grimsdalen 2010 | training | 155 | 219 200 | 231 200 | 6 893 400 | 6 899 400 |
| pitfall trap | Nordfron 2012 | training | 80 | 200 800 | 226 400 | 6 833 400 | 6 848 400 |
| pitfall trap | Nordfron 2013 | test | 31 | 191 200 | 195 200 | 6 831 000 | 6 832 200 |
| pitfall trap | Nordfron 2017 | validation | 16 | 211 200 | 224 800 | 6 831 000 | 6 842 400 |
| pitfall trap | Nordfron 2018 | validation | 215 | 196 800 | 212 000 | 6 821 400 | 6 841 800 |
| pitfall trap | Nordfron Olstappen 2010 | training | 68 | 195 470 | 202 400 | 6 827 400 | 6 830 400 |
| | | validation | 57 | 200 800 | 204 800 | 6 826 400 | 6 828 000 |
| | | test | 41 | 195 200 | 202 400 | 6 830 400 | 6 832 200 |
| pitfall trap | Nordfron Venabu 2018 | validation | 17 | 222 400 | 227 200 | 6 844 800 | 6 862 200 |
| pitfall trap | Vågå 2018 | training | 104 | 171 200 | 188 800 | 6 832 800 | 6 862 800 |

Table 4. Summary of ALS data used for neural network training and evaluation.

| | number of objects | | | | | | |
|---|---|---|---|---|---|---|---|
| object type | training | | validation | | test | | sum |
| charcoal kiln | 773 | 73 % | 190 | 18 % | 95 | 9 % | 1058 |
| grave mound | 629 | 60 % | 287 | 27 % | 130 | 12 % | 1046 |
| pitfall trap | 1057 | 71 % | 305 | 21 % | 119 | 8 % | 1481 |
| sum | 2459 | 69 % | 782 | 22 % | 344 | 10 % | 3585 |

## 3.2 Alternative subdivision

Table 5. Alternative subdivision of ALS datasets into training, validation and test sets.

| object type | dataset | subset | object count | extent of dataset in UTM zone 33 N | | | |
|---|---|---|---|---|---|---|---|
| | | | | west | east | south | north |
| charcoal kiln | Lesja 2013 | training | 773 | 160 750 | 172 000 | 6 907 150 | 6 919 600 |
| | | validation | 190 | 154 350 | 172 000 | 6 902 500 | 6 919 600 |
| | | test | 95 | 144 800 | 154 400 | 6 916 000 | 6 922 800 |
| grave mound | Brumunddal 2016 | validation | 23 | 260 000 | 280 000 | 6 753 600 | 6 774 600 |
| | | test | 50 | 269 600 | 283 200 | 6 736 200 | 6 753 000 |
| grave mound | Horten 2016 | training | 38 | 238 400 | 243 200 | 6 588 000 | 6 593 400 |
| grave mound | Hå Jæren 2017 | validation | 84 | -44 000 | -36 000 | 6 531 800 | 6 545 400 |
| grave mound | Larvik 2017 | training | 288 | 206 400 | 220 800 | 6 547 800 | 6 563 400 |
| | | validation | 165 | 205 600 | 218 400 | 6 565 800 | 6 596 400 |
| | | test | 57 | 220 800 | 226 400 | 6 553 200 | 6 565 200 |
| grave mound | Oppdal Vang | training | 219 | 224 690 | 225 600 | 6 951 850 | 6 952 925 |
| grave mound | Sarpsborg 2015 | validation | 30 | 274 400 | 280 000 | 6 576 000 | 6 583 800 |
| | | test | 18 | 276 000 | 284 800 | 6 565 200 | 6 575 400 |
| grave mound | Steinkjer 2011 | test | 30 | 321 600 | 348 800 | 7 087 800 | 7 113 000 |
| grave mound | Steinkjer 2017 | test | 44 | 322 400 | 345 600 | 7 097 400 | 7 119 600 |
| pitfall trap | Dovre 2011 | training | 368 | 199 200 | 218 400 | 6 902 400 | 6 915 000 |
| | | validation | 282 | 192 000 | 206 400 | 6 885 600 | 6 902 400 |
| pitfall trap | Dovre 2013 | test | 29 | 190 400 | 204 000 | 6 878 400 | 6 897 000 |
| pitfall trap | Dovre 2017 | test | 15 | 190 400 | 196 800 | 6 882 000 | 6 897 000 |
| pitfall trap | Dovre Folldal 2018 | test | 3 | 233 600 | 234 400 | 6 891 600 | 6 892 200 |
| pitfall trap | Dovre Grims-dalen 2010 | test | 155 | 219 200 | 231 200 | 6 893 400 | 6 899 400 |
| pitfall trap | Nordfron 2012 | training | 31 | 202 800 | 206 400 | 6 837 000 | 6 839 400 |
| | | validation | 18 | 221 600 | 226 400 | 6 835 800 | 6 848 400 |
| | | test | 31 | 200 800 | 219 200 | 6 833 400 | 6 840 600 |
| pitfall trap | Nordfron 2013 | training | 25 | 191 200 | 193 600 | 6 831 000 | 6 832 200 |
| | | test | 6 | 193 600 | 194 400 | 6 831 000 | 6 832 200 |
| pitfall trap | Nordfron 2017 | training | 3 | 220 000 | 221 600 | 6 841 800 | 6 842 400 |
| | | validation | 12 | 220 800 | 224 800 | 6 837 600 | 6 839 400 |
| | | test | 1 | 211 200 | 212 000 | 6 831 000 | 6 831 600 |
| pitfall trap | Nordfron 2018 | training | 48 | 210 400 | 212 000 | 6 837 000 | 6 839 400 |
| | | validation | 152 | 196 800 | 200 000 | 6 825 000 | 6 827 400 |
| | | test 1 | 6 | 208 800 | 210 400 | 6 841 200 | 6 841 800 |
| | | test 2 | 9 | 197 600 | 199 200 | 6 821 400 | 6 822 600 |
| pitfall trap | Nordfron Olstappen 2010 | training | 68 | 195 470 | 202 400 | 6 827 400 | 6 830 400 |
| | | validation | 57 | 200 800 | 204 800 | 6 826 400 | 6 828 000 |
| | | test | 41 | 195 200 | 202 400 | 6 830 400 | 6 832 200 |
| pitfall trap | Nordfron Venabu 2018 | validation | 10 | 222 400 | 224 000 | 6 861 000 | 6 862 200 |
| | | test | 7 | 224 000 | 227 200 | 6 844 800 | 6 858 600 |
| pitfall trap | Vågå 2018 | training | 70 | 184 000 | 188 800 | 6 849 650 | 6 862 800 |
| | | validation | 34 | 171 200 | 180 700 | 6 832 800 | 6 847 200 |

Table 6. Summary of alternative subdivision of ALS data used for neural network training and evaluation.

| | number of objects | | | | | | |
|---|---|---|---|---|---|---|---|
| object type | training | | validation | | test | | sum |
| grave mound | 508 | 51 % | 235 | 24 % | 253 | 25 % | 996 |
| pitfall trap | 784 | 42 % | 699 | 37 % | 387 | 21 % | 1870 |
| charcoal kiln | 960 | 73 % | 235 | 18 % | 115 | 9 % | 1310 |
| sum | 2252 | 54 % | 1169 | 28 % | 755 | 18 % | 4176 |

## 3.3 Unlabelled test data

The three unlabeled test data covering Øvre Eiker municipality (Table 2) consisted of 1493 LAS files in total (Table 7).

Table 7. Extent of the ALS datasets covering Øvre Eiker municipality.

| dataset | extent of dataset in UTM zone 32 N | | | | number of files |
|---|---|---|---|---|---|
| | west | east | south | north | |
| Øvre Eiker 2015 | 537 600 | 556 000 | 6 609 600 | 6 640 800 | 494 |
| Øvre Eiker Flesberg 2017 | 535 200 | 556 800 | 6 604 800 | 6 642 600 | 829 |
| Øvre Eiker Modum 2017 | 550 400 | 560 000 | 6 626 400 | 6 639 000 | 170 |
| Combined | 535 200 | 560 000 | 6 604 800 | 6 642 600 | 1 493 |

## 3.4 Overview maps of ALS datasets

### 3.4.1 Initial subdivision

Overview maps of the ALS datasets (Table 3) appear below (Figure 1-Figure 27).

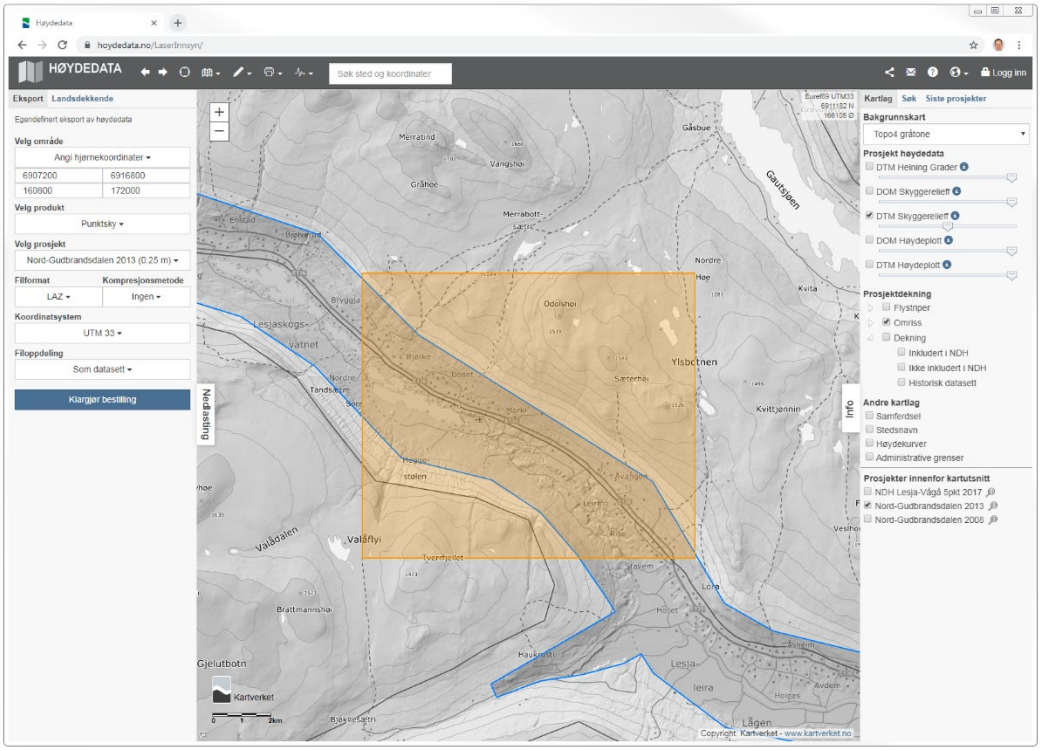



Figure 1. Lesja 2013 dataset, training subset.

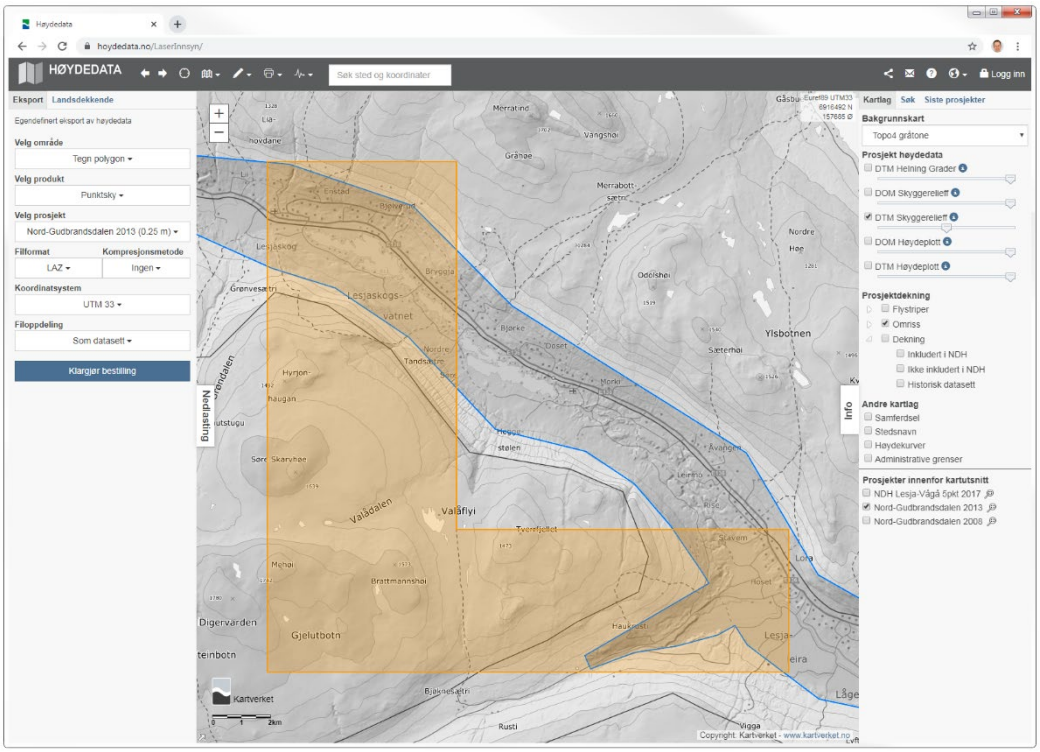



Figure 2. Lesja 2013 dataset, validation subset.

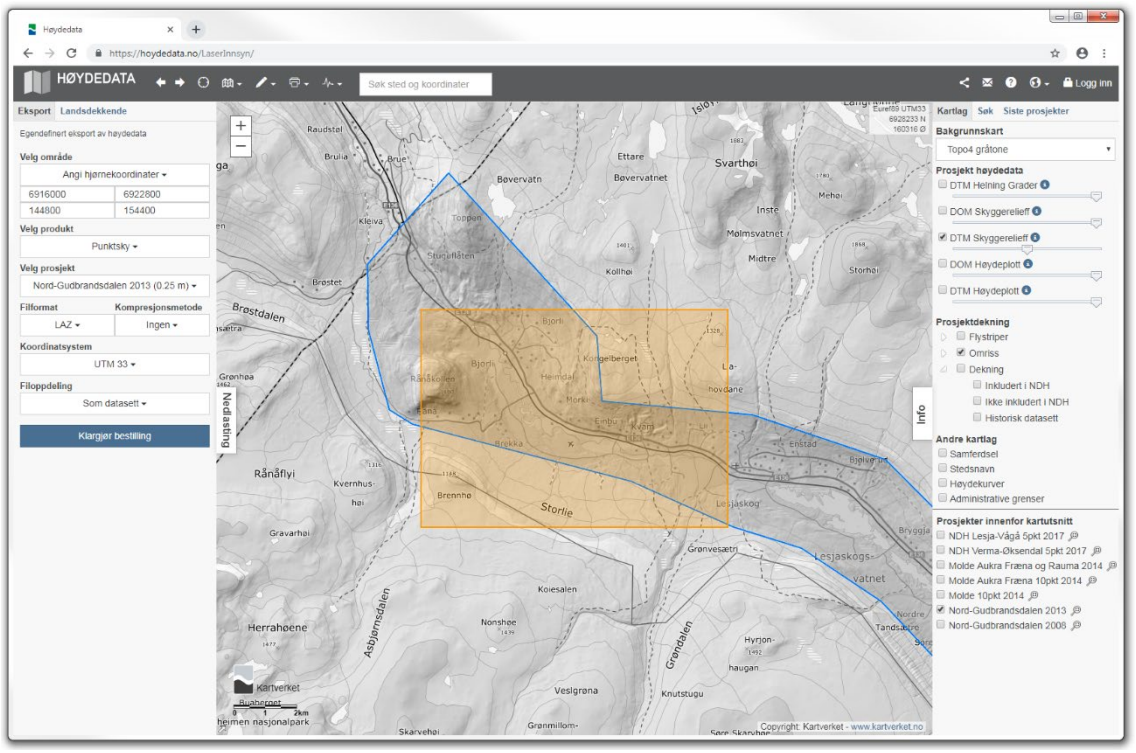



Figure 3. Lesja 2013 dataset, test subset.

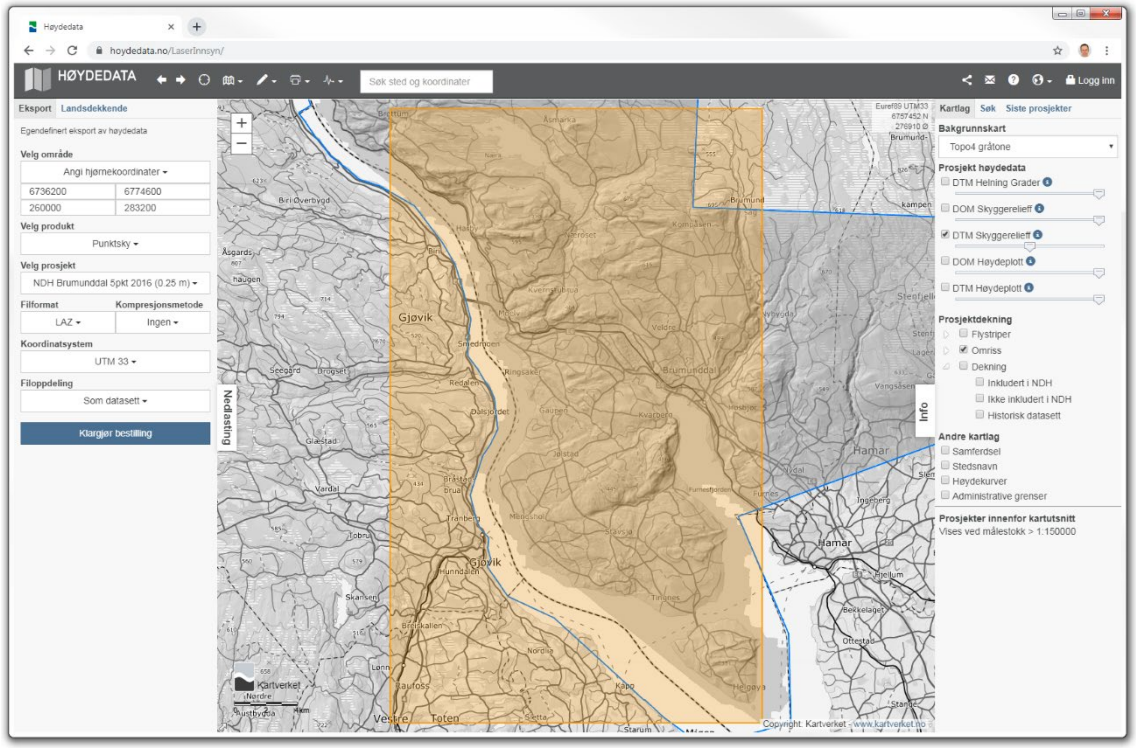



Figure 4. Brumunddal 2016 data set.

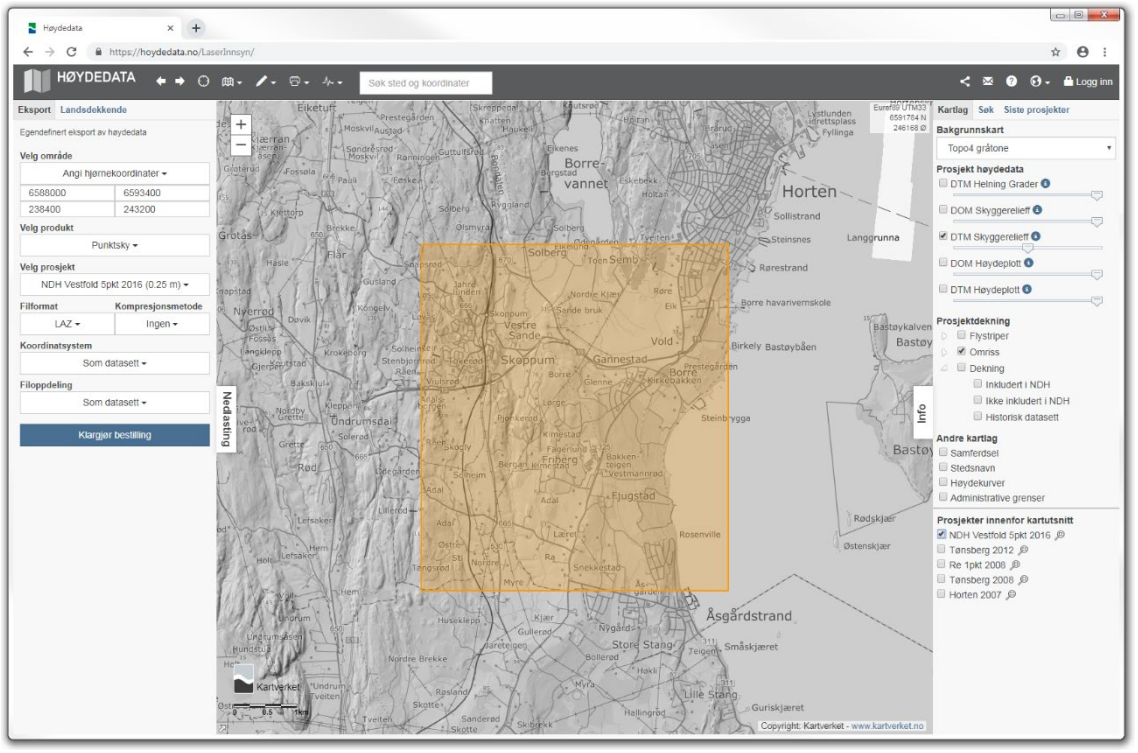



Figure 5. Horten 2016 dataset.

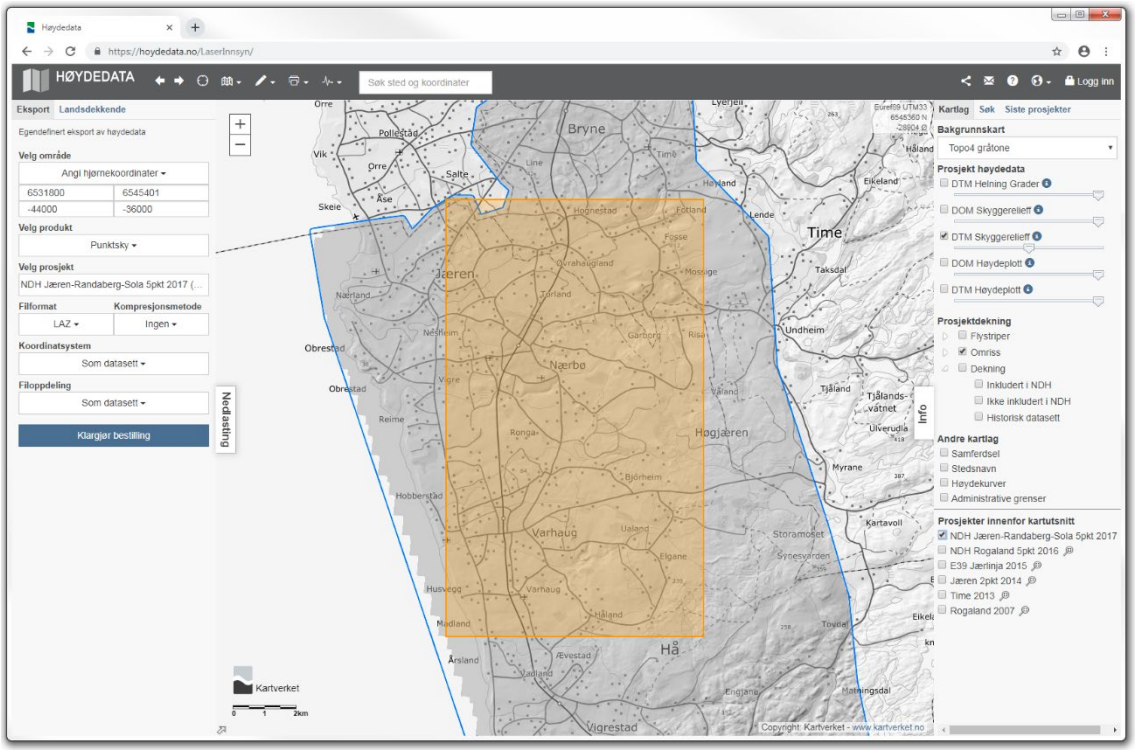



Figure 6. Hå Jæren 2017 dataset.

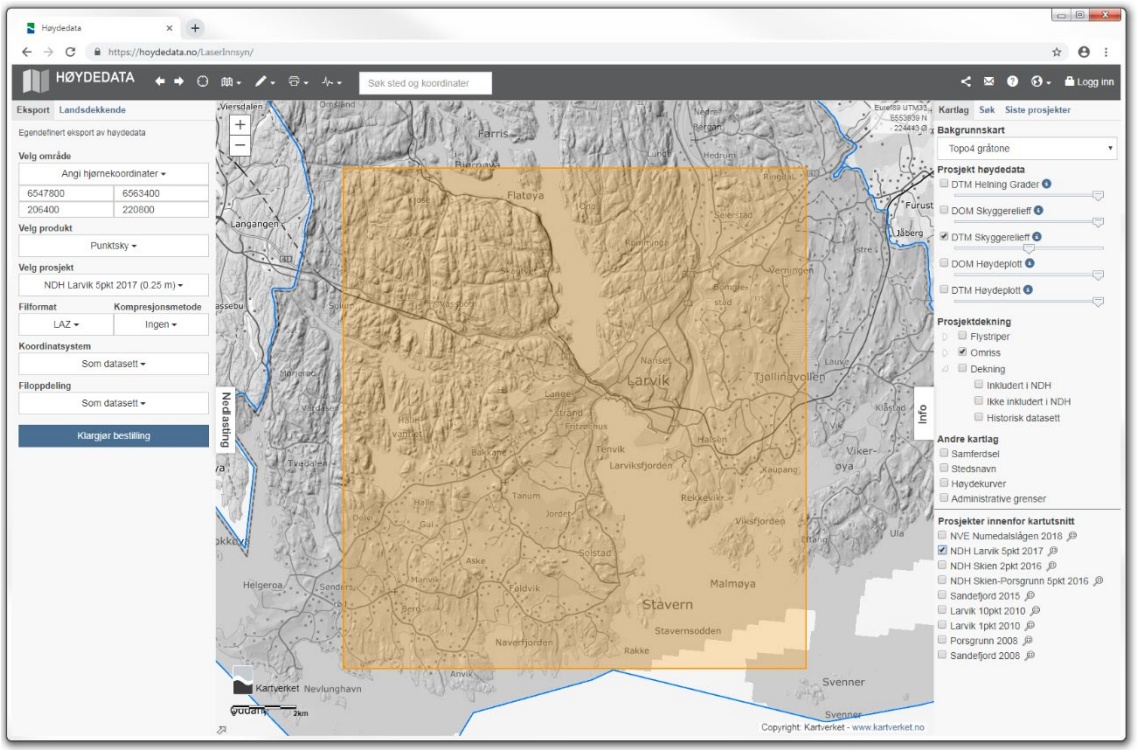



Figure 7. Larvik 2017 dataset, training subset.

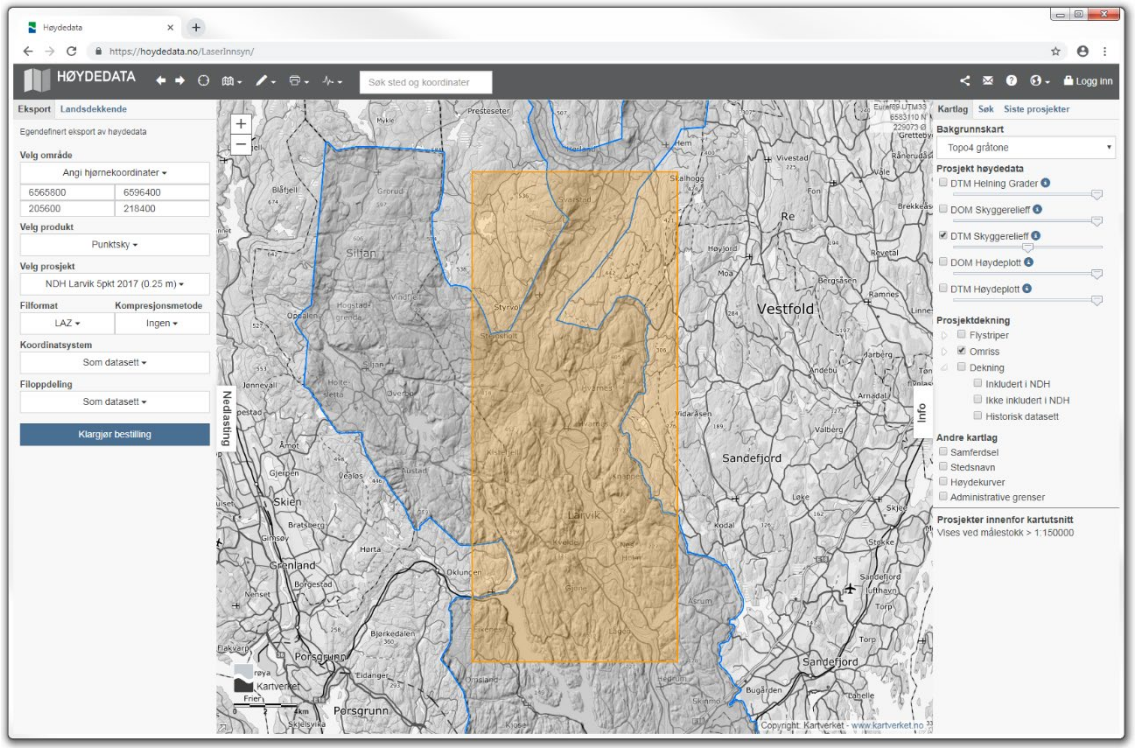



Figure 8. Larvik 2017 dataset, validation subset.

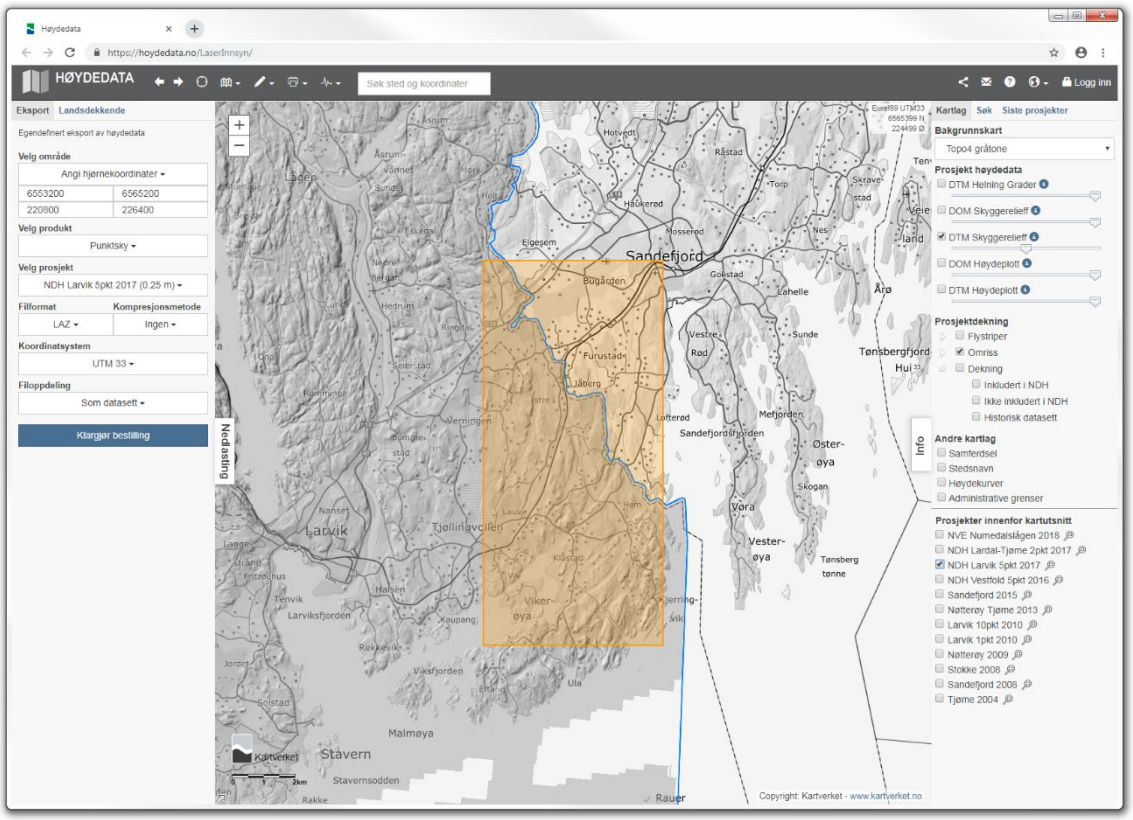



Figure 9. Larvik 2017 dataset, test subset.

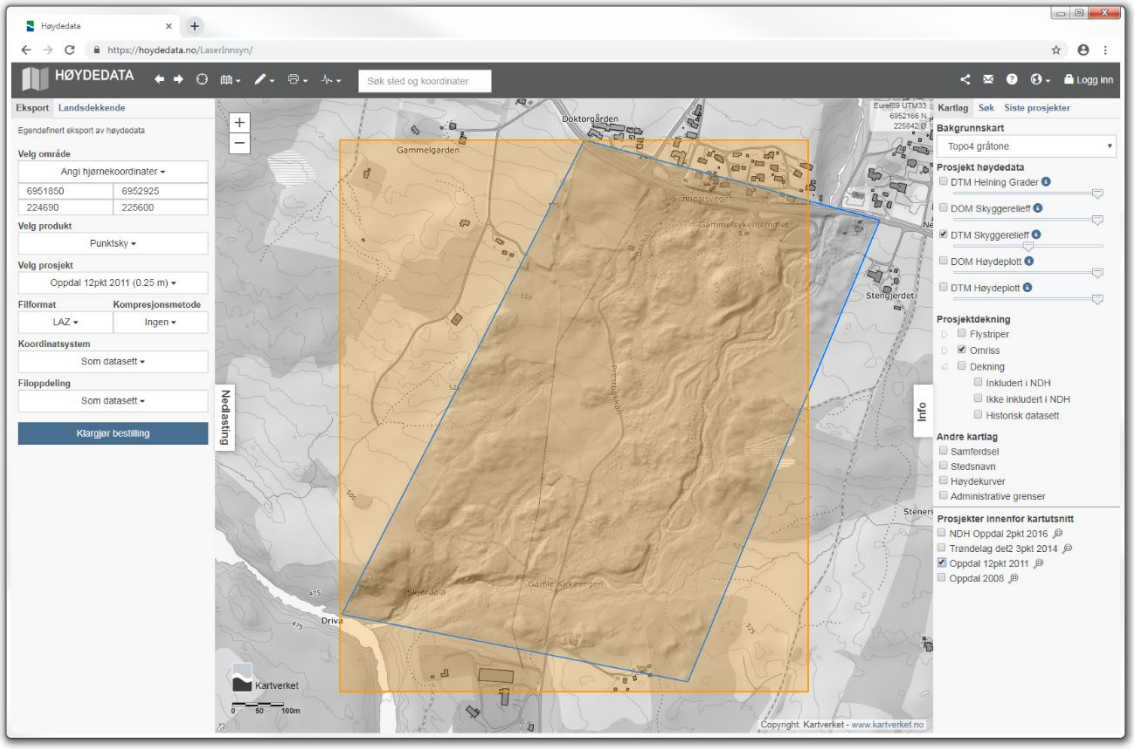



Figure 10. Oppdal Vang 2011 dataset.

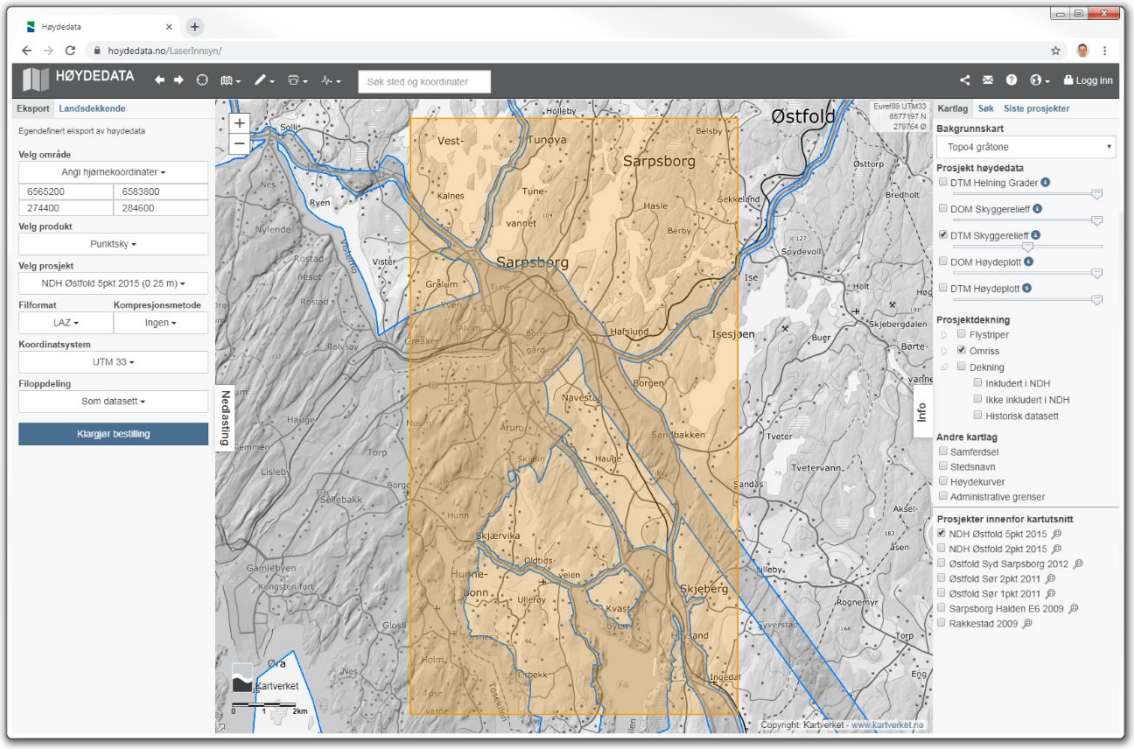



Figure 11. Sarpsborg 2015 dataset.

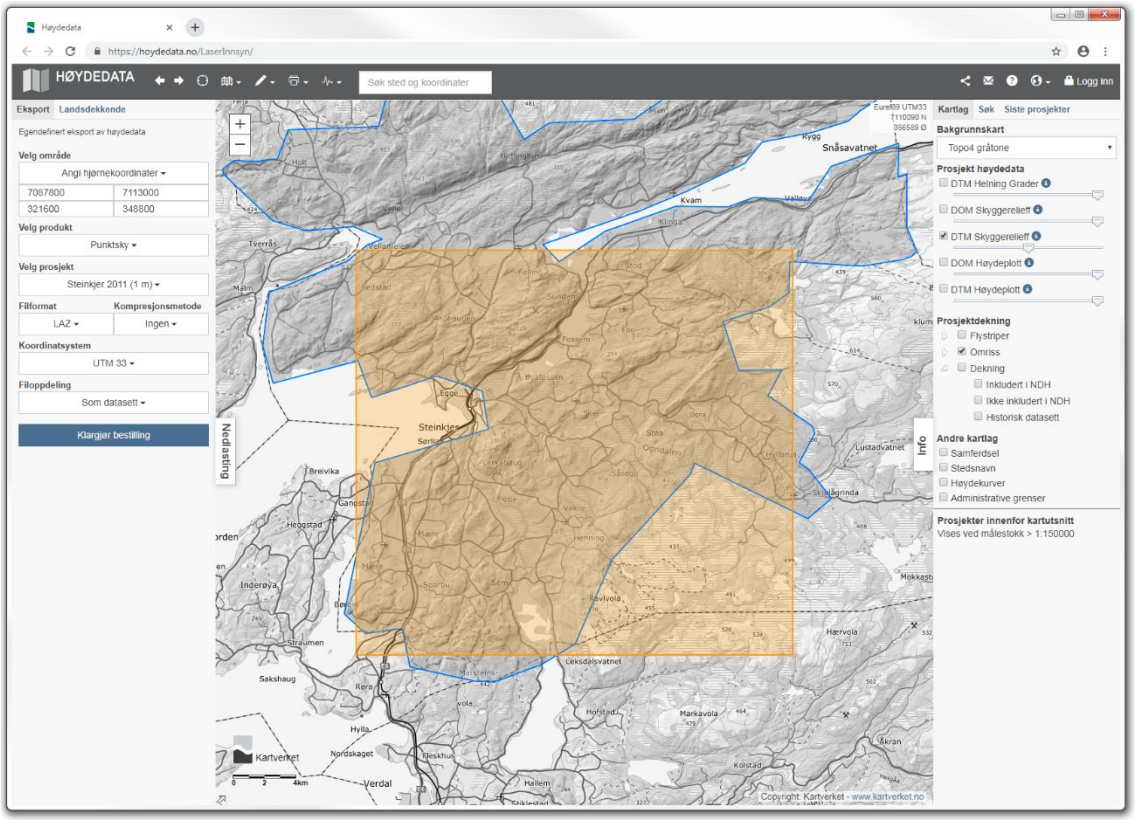



Figure 12. Steinkjer 2011 dataset.

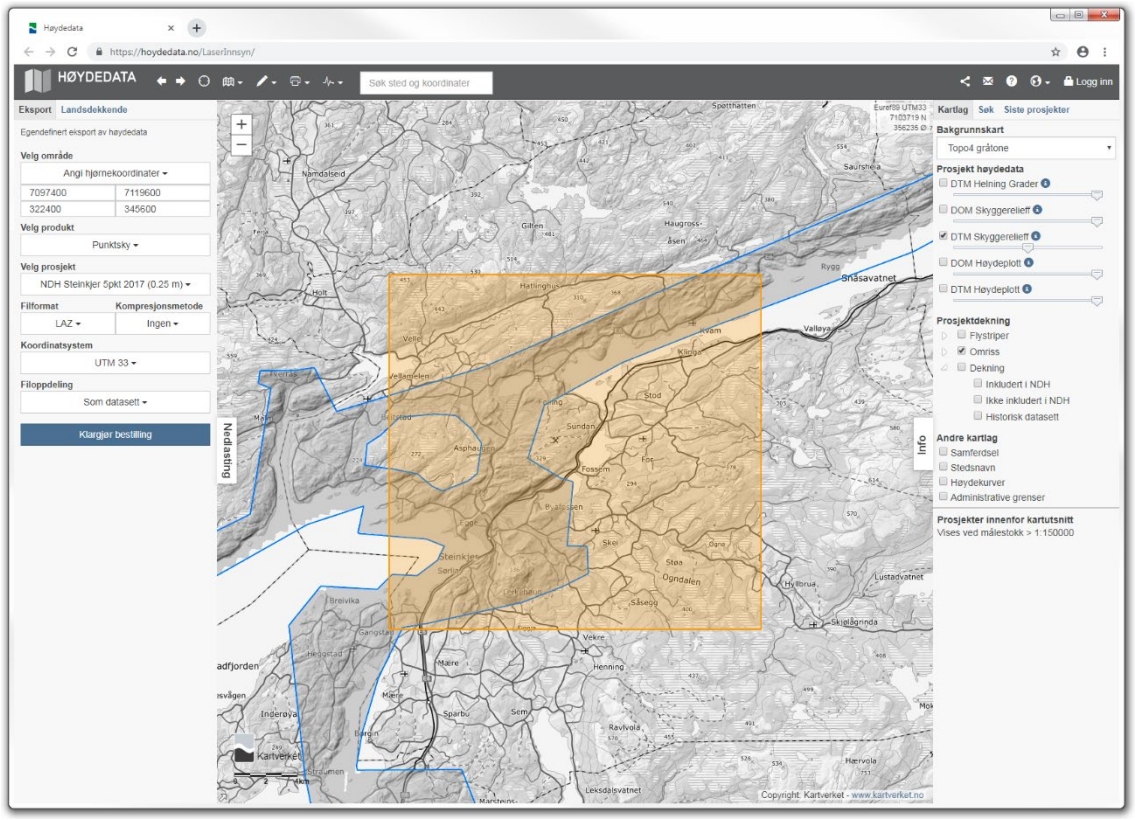



Figure 13. Steinkjer 2017 dataset.

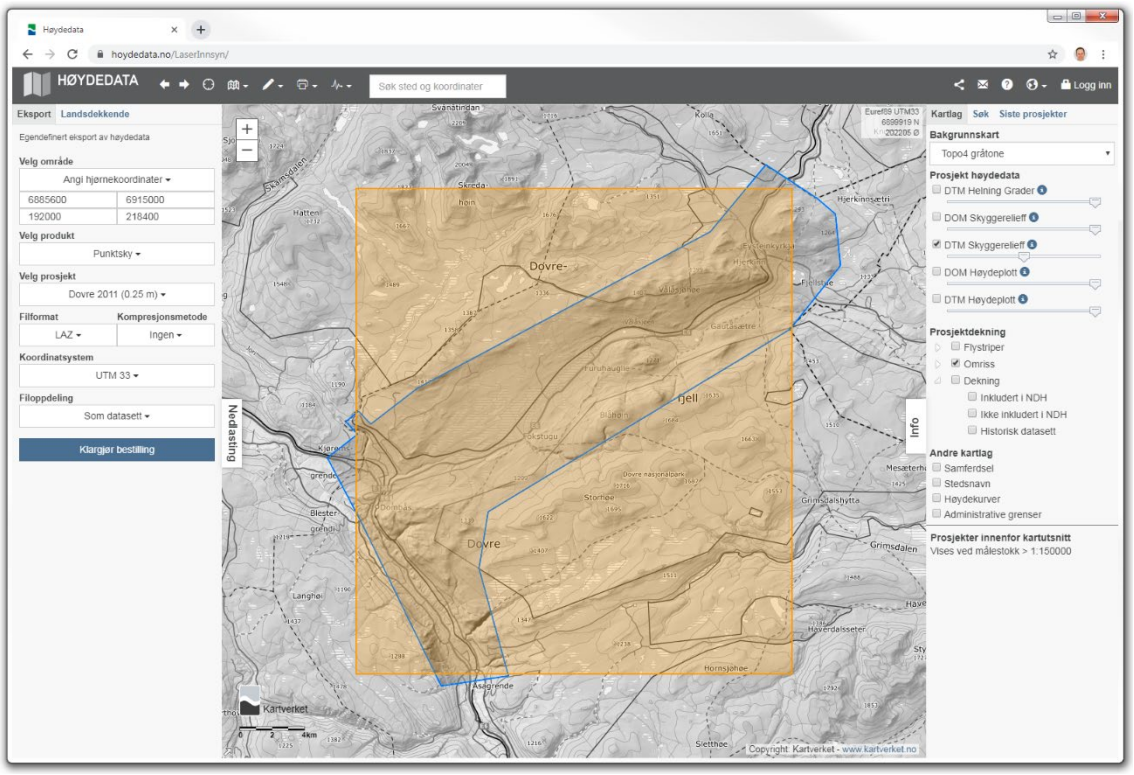



Figure 14. Dovre 2011 dataset.

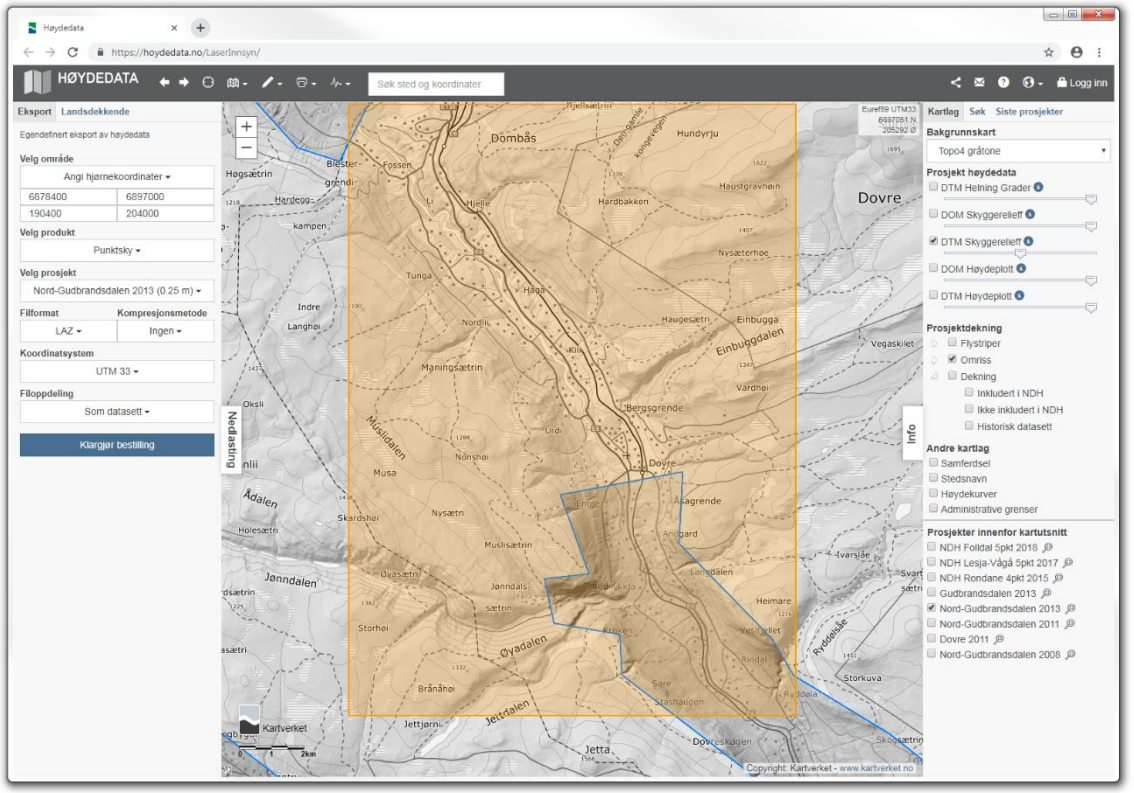



Figure 15. Dovre 2013 dataset.

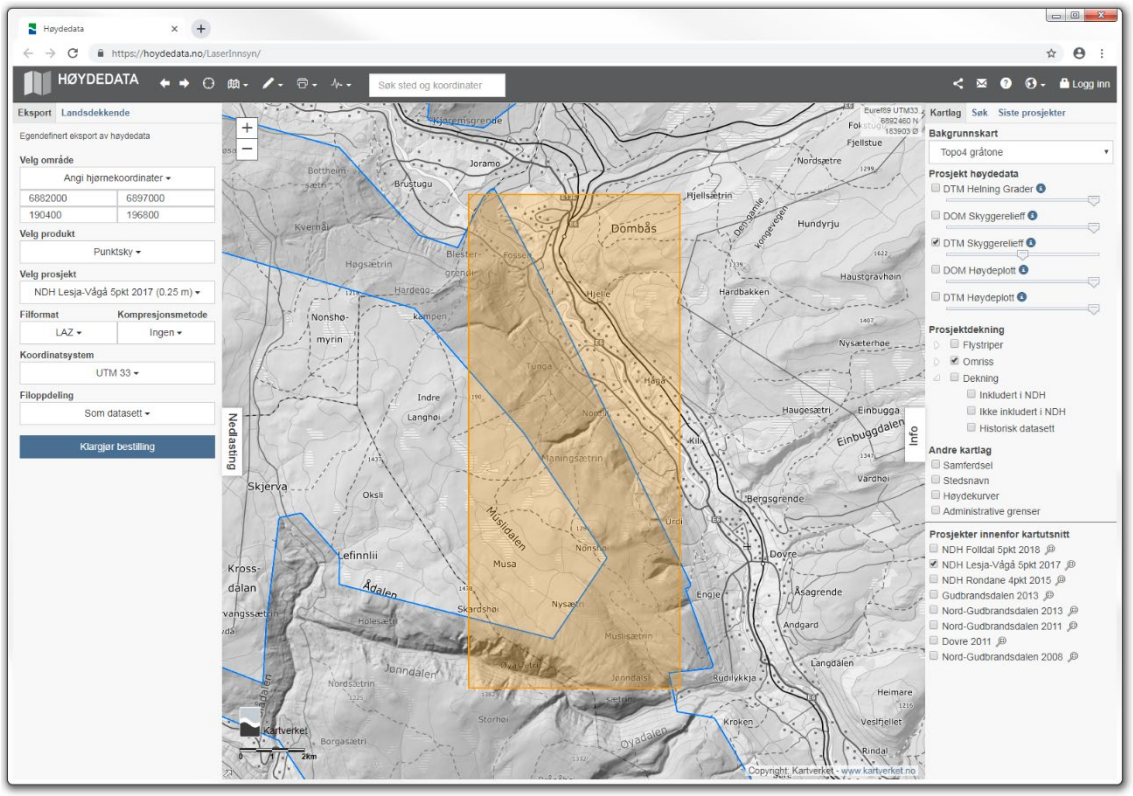



Figure 16. Dovre 2017 dataset.

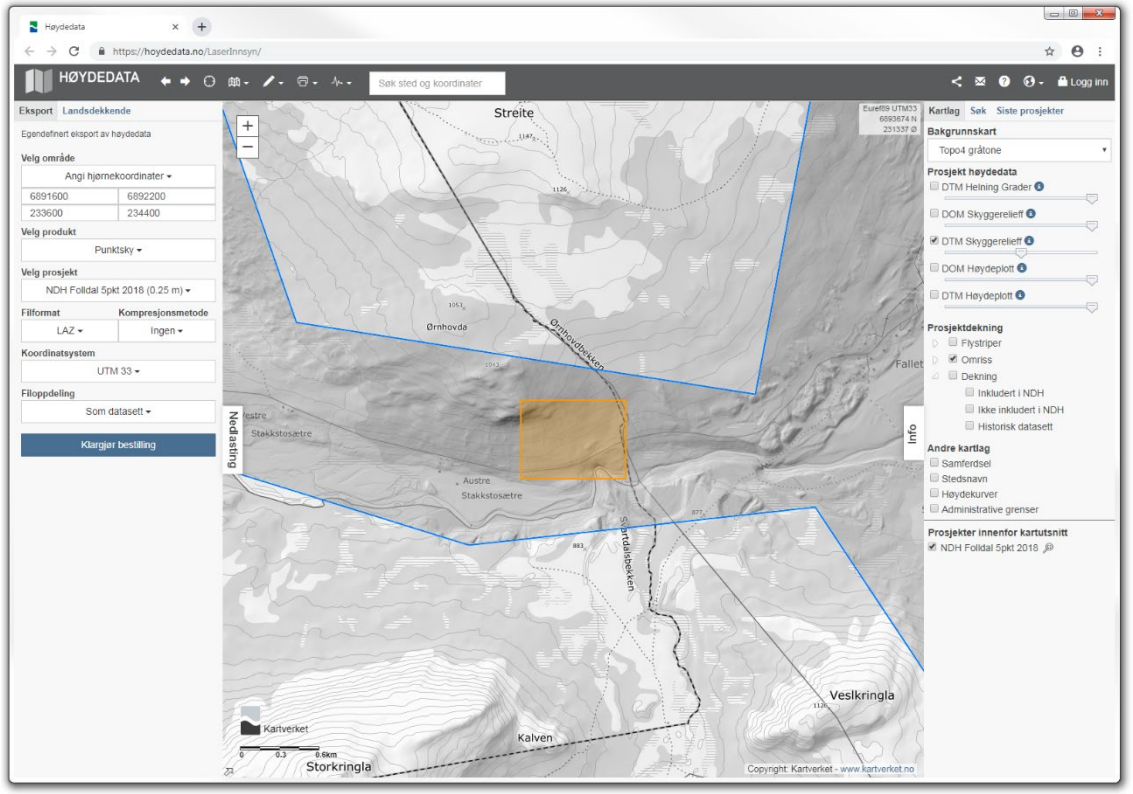



Figure 17. Dovre Folldal 2018 dataset.

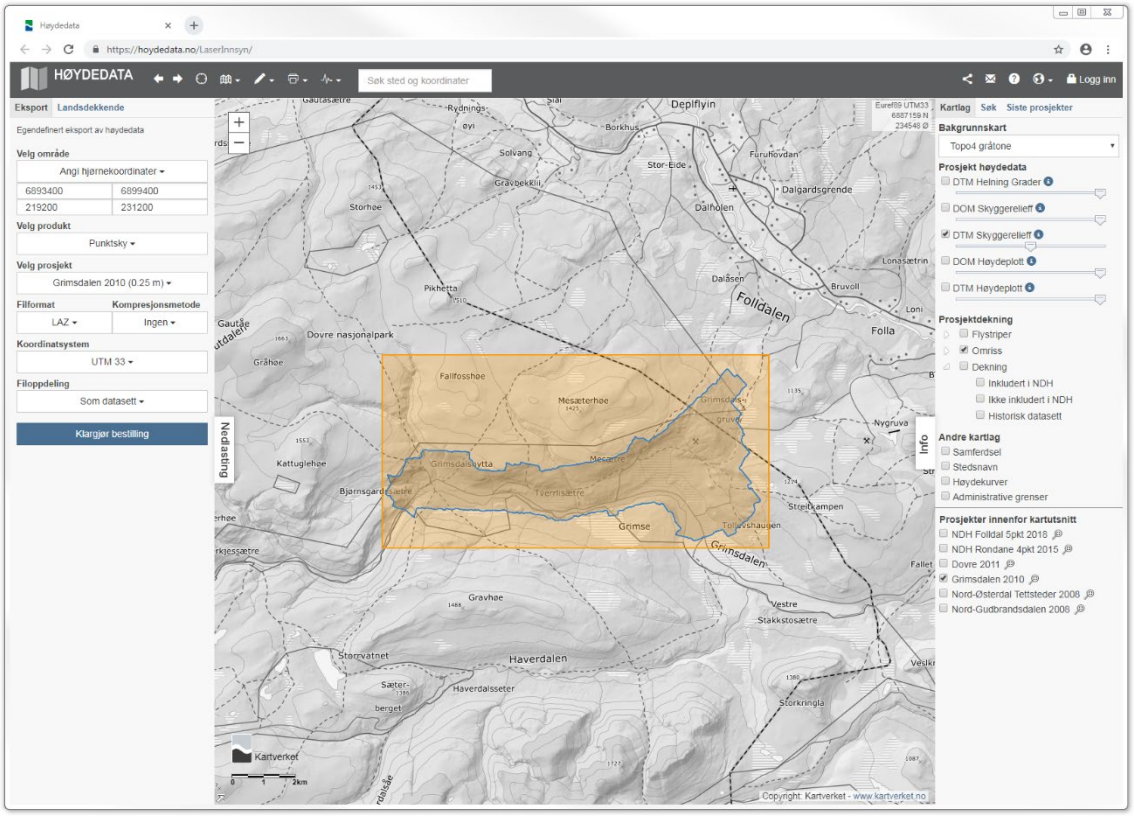



Figure 18. Dovre Grimsdalen 2010 dataset.

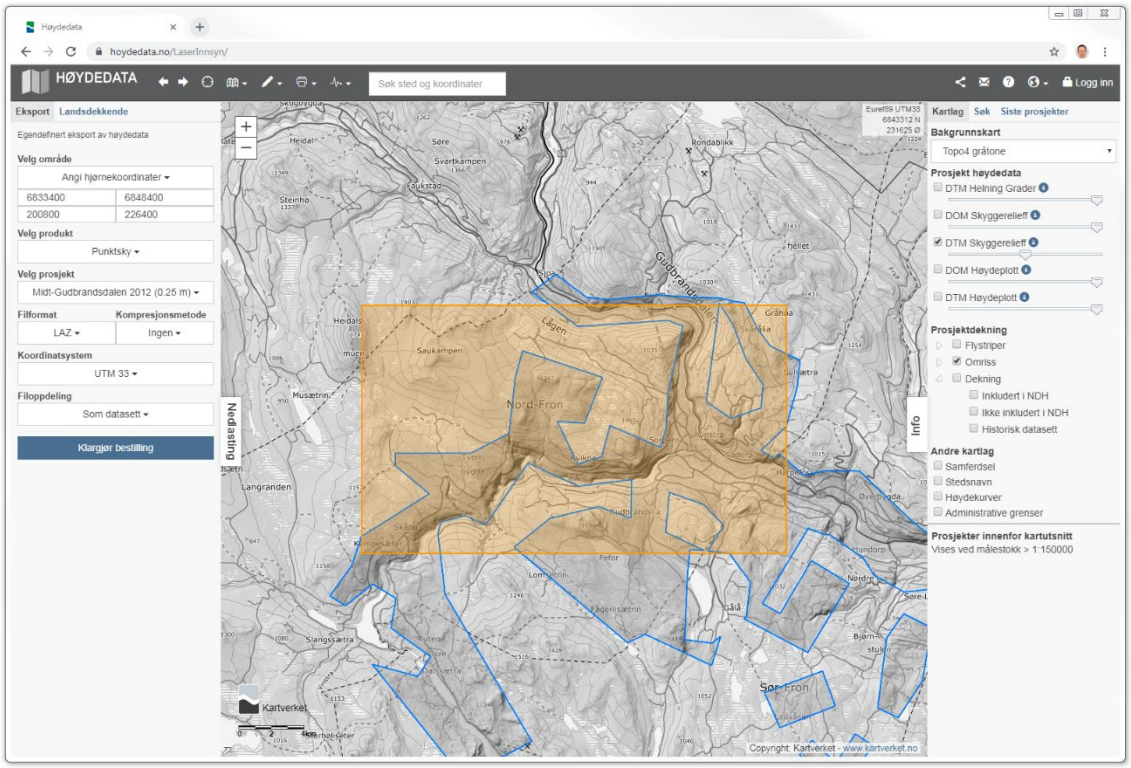



Figure 19. Nordfron 2012 dataset.

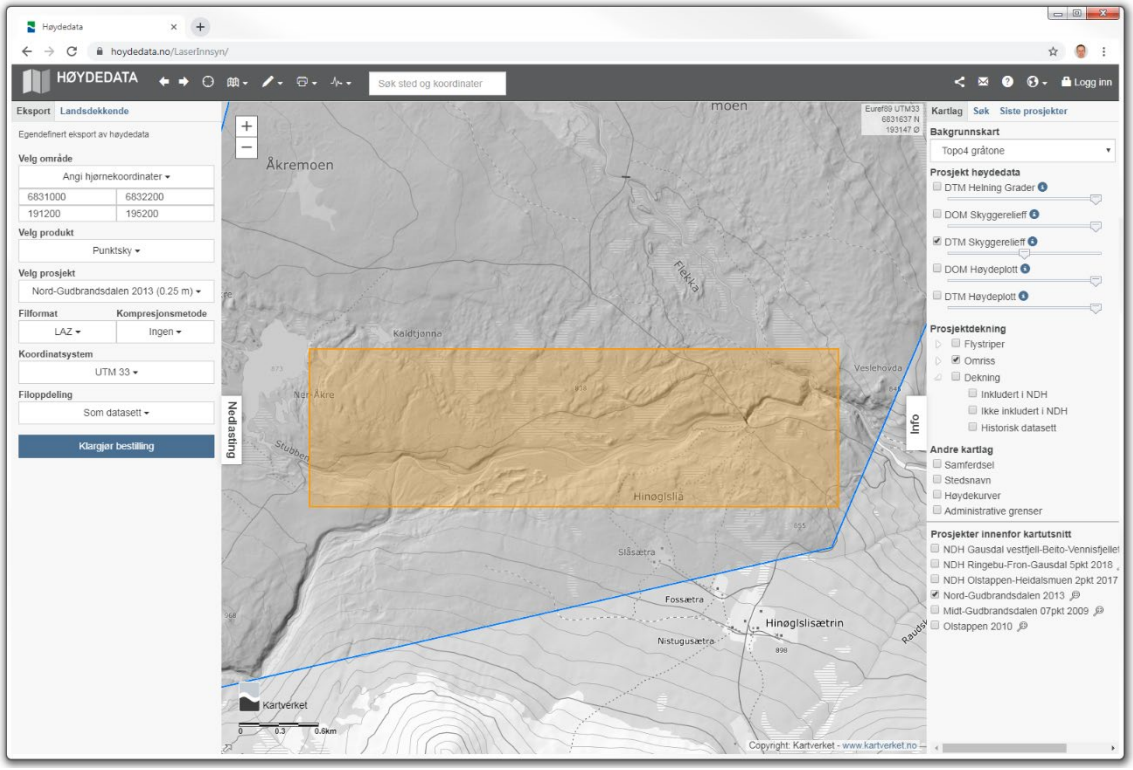



Figure 20. Nordfron 2013 dataset.

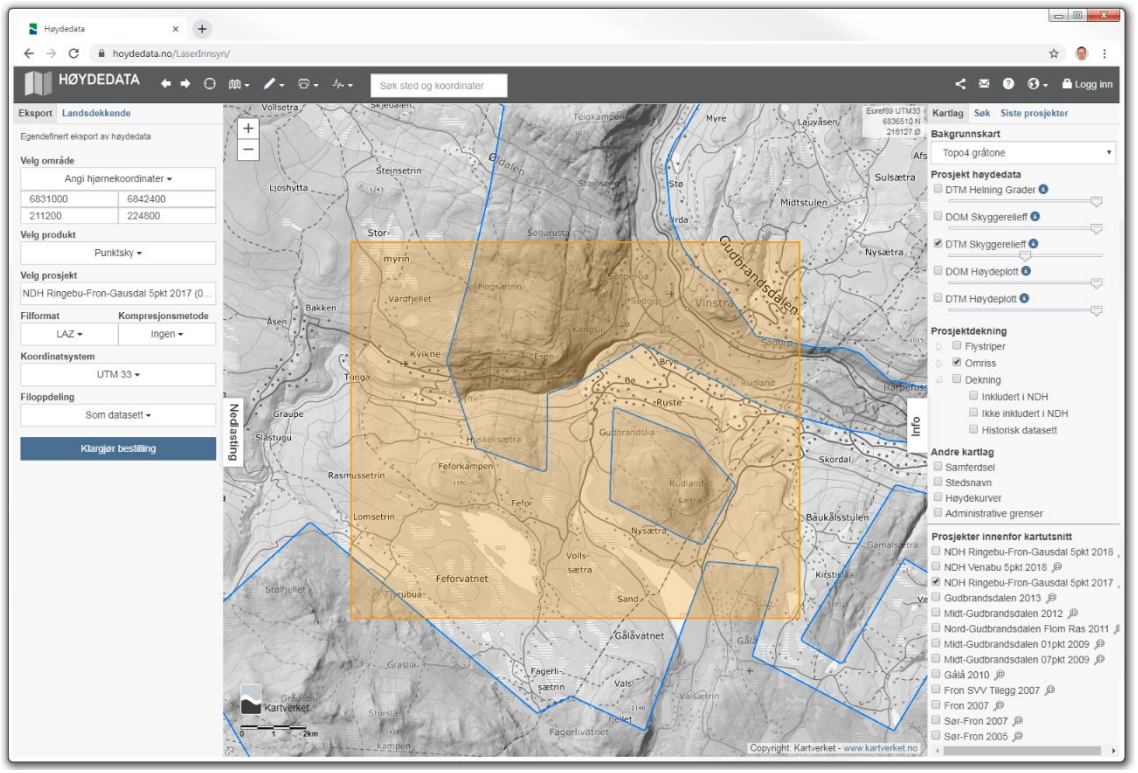



Figure 21. Nordfron 2017 dataset.

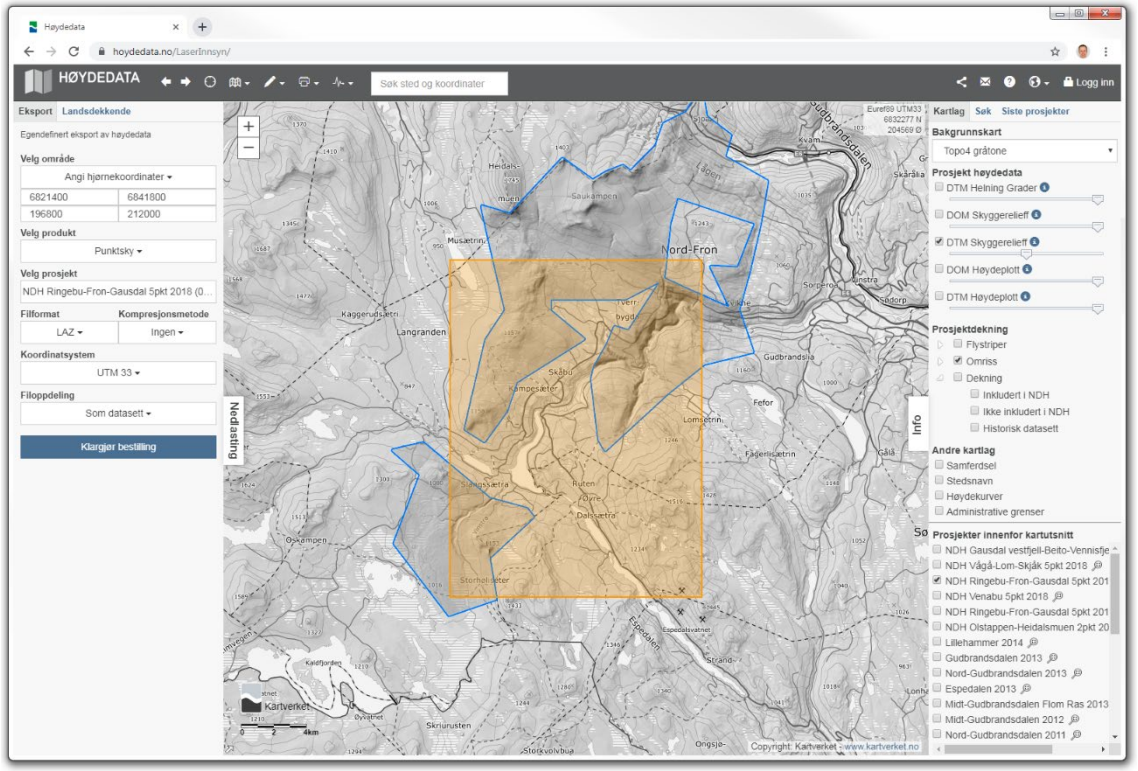



Figure 22. Nordfron 2018 dataset.

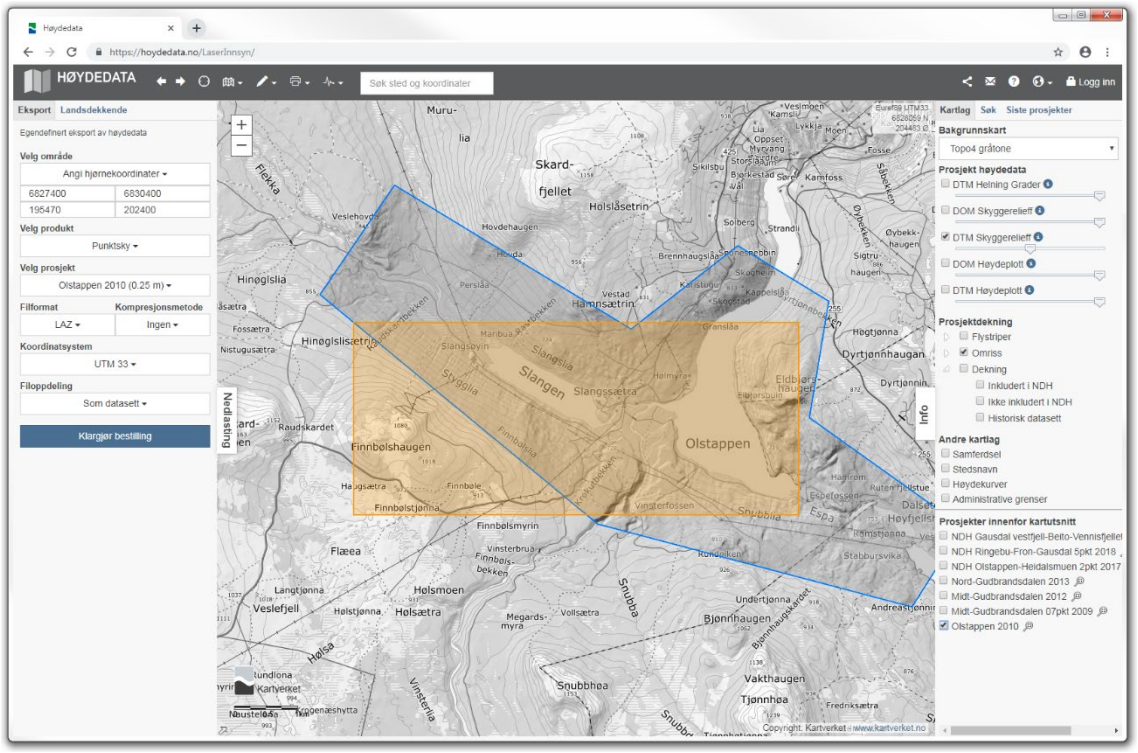



Figure 23. Nordfron Olstappen 2010 dataset, training subset.

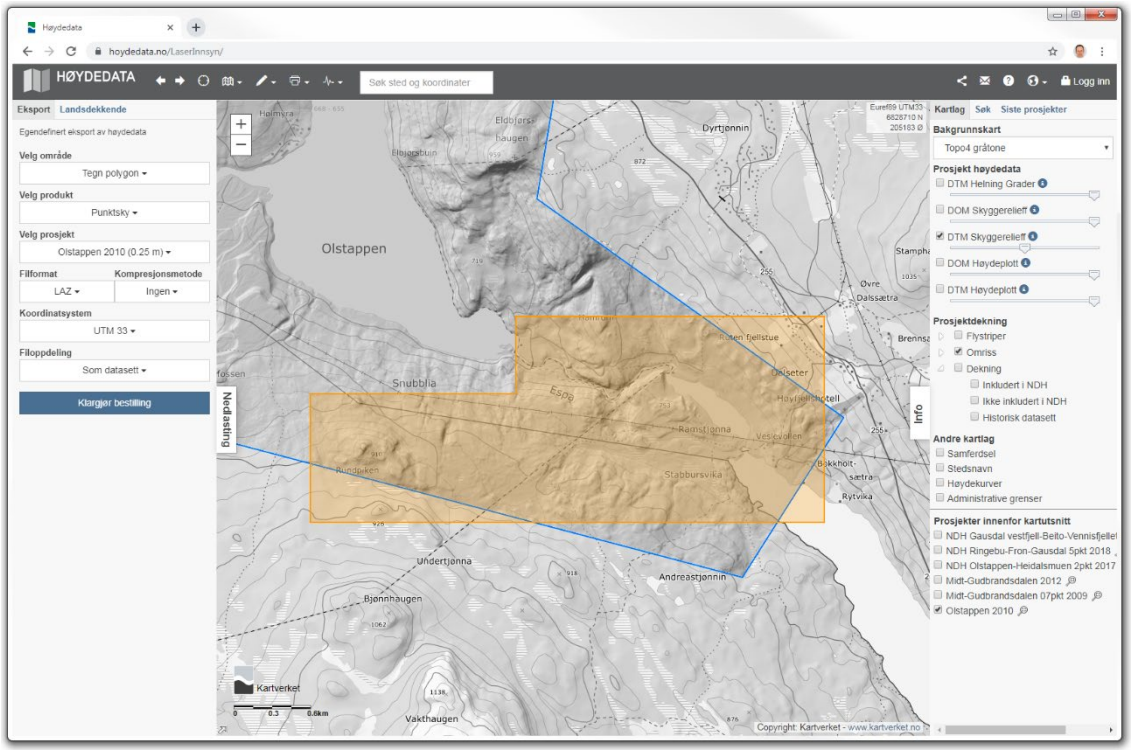



Figure 24. Nordfron Olstappen 2010 dataset, validation subset.

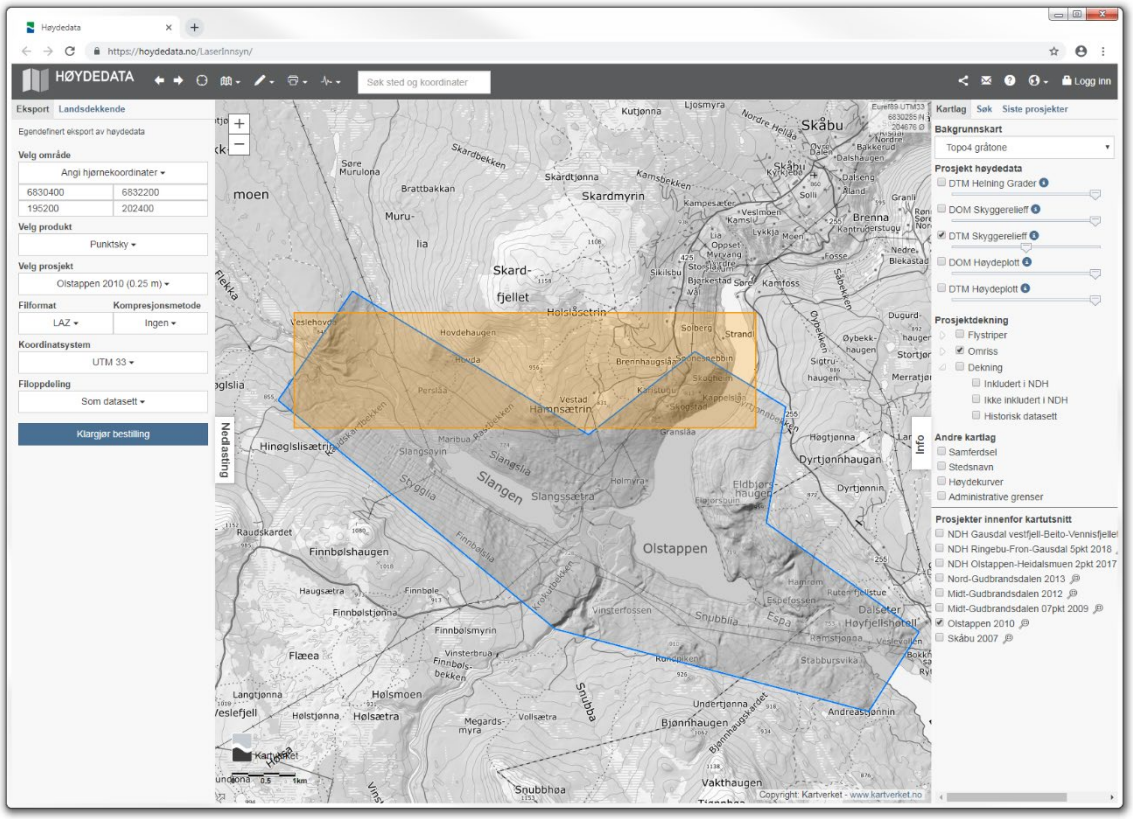



Figure 25. Nordfron Olstappen dataset, test subset.

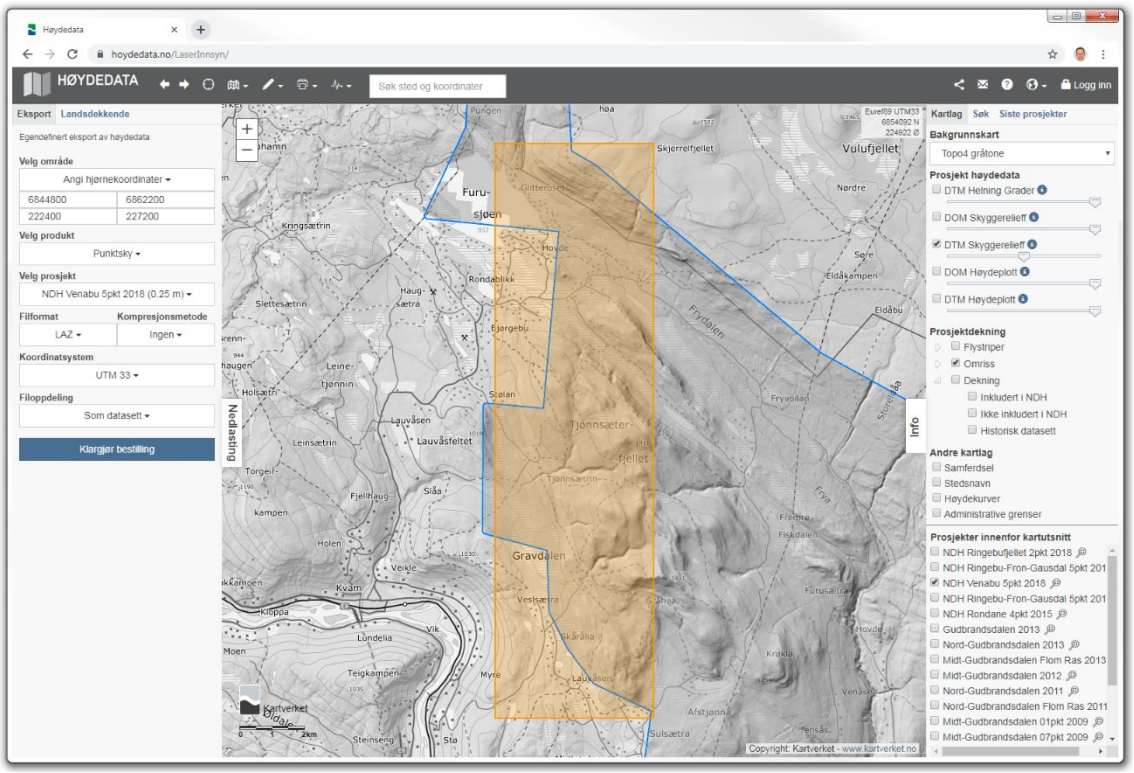



Figure 26. Nordfron Venabu 2018 dataset.

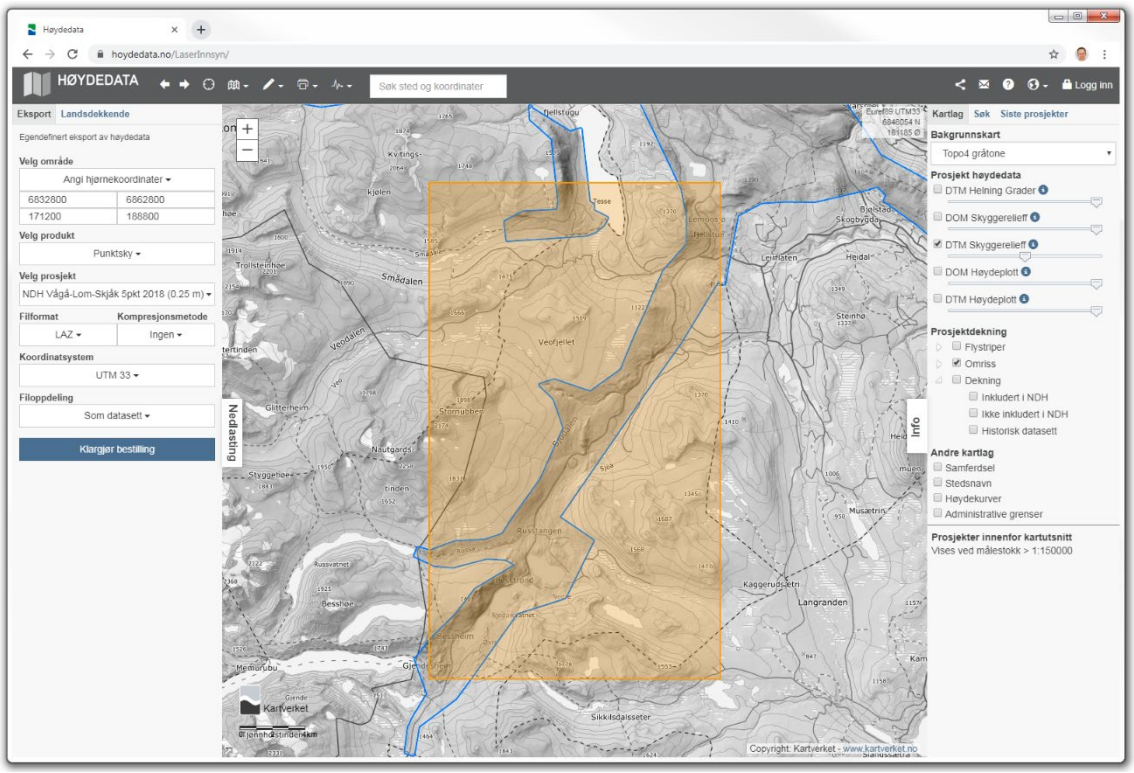



Figure 27. Vågå 2018 dataset.

### 3.4.2 Alternative subdivision

Overview maps of the ALS datasets (Table 5) appear below (Figure 28-Figure 66).

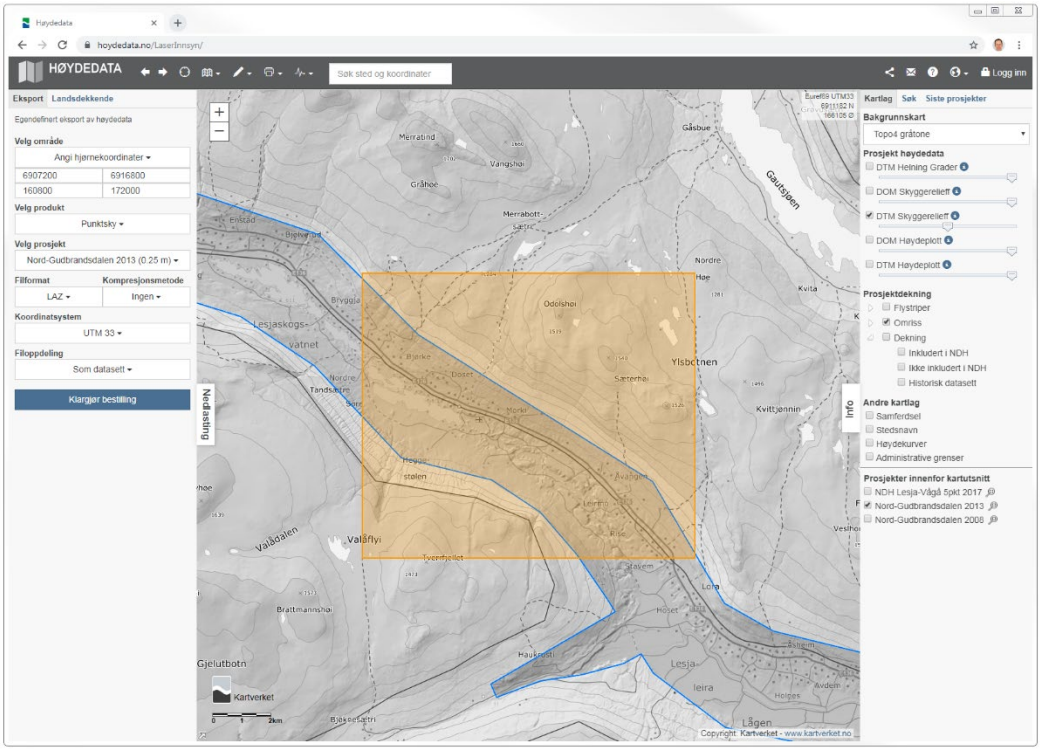



Figure 28. Lesja 2013 dataset, training subset.

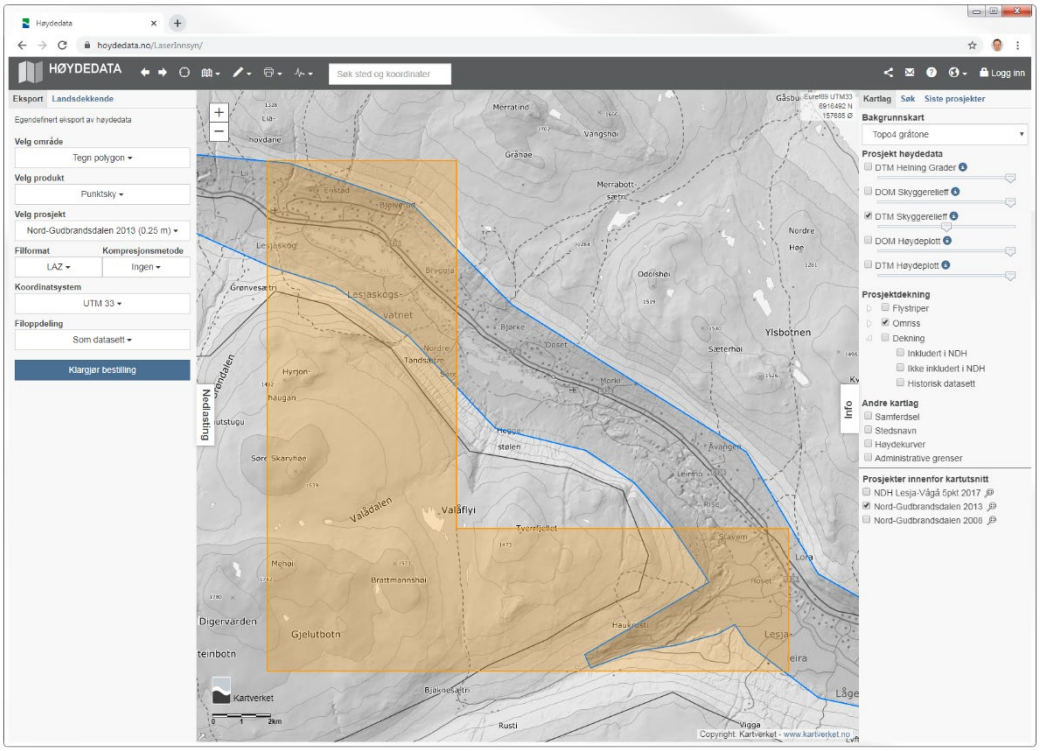



Figure 29. Lesja 2013 dataset, validation subset.

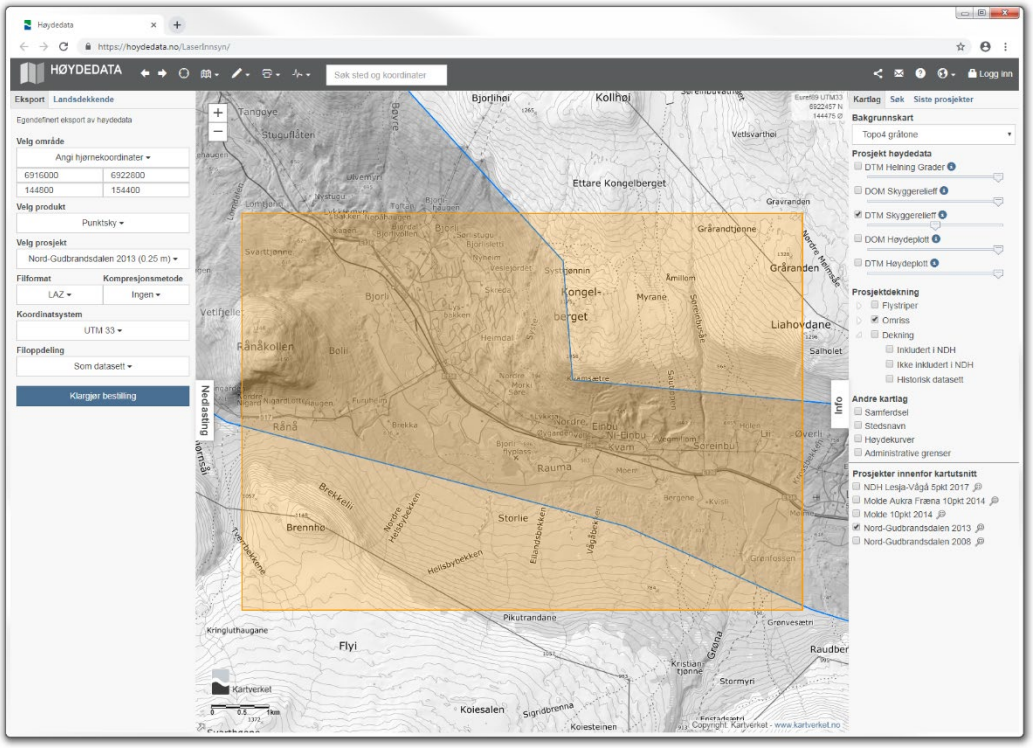



Figure 30. Lesja 2013 dataset, test subset.

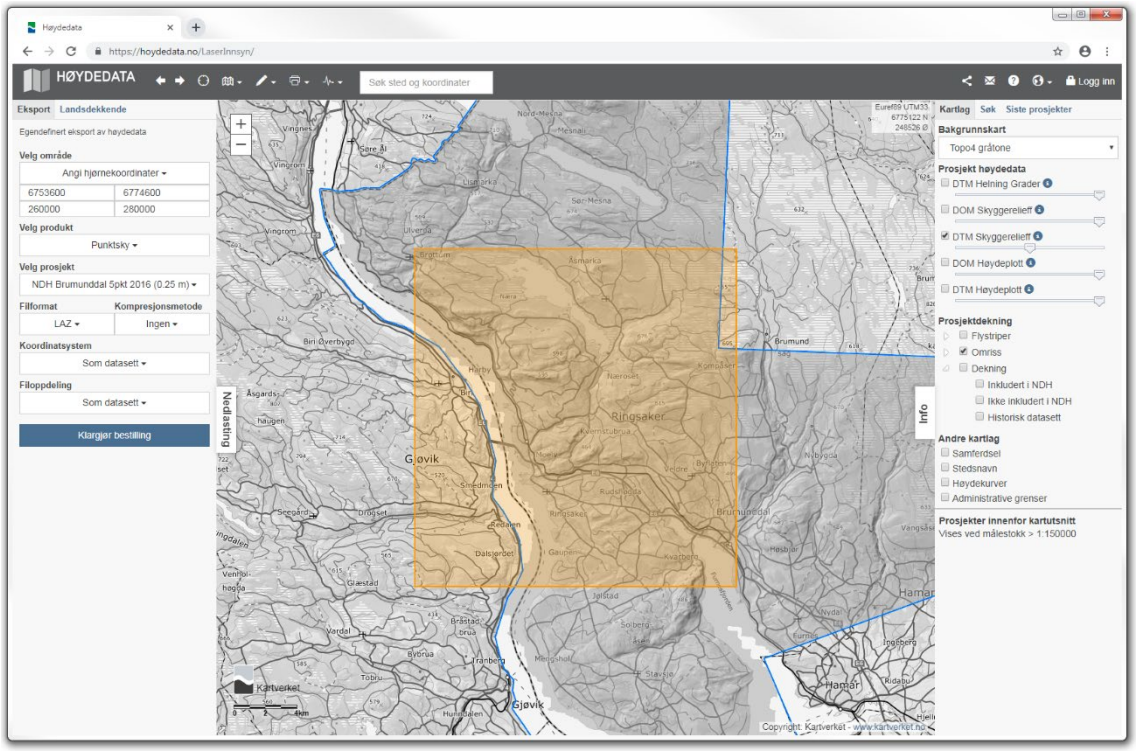



Figure 31. Brumunddal 2016 dataset, validation subset.

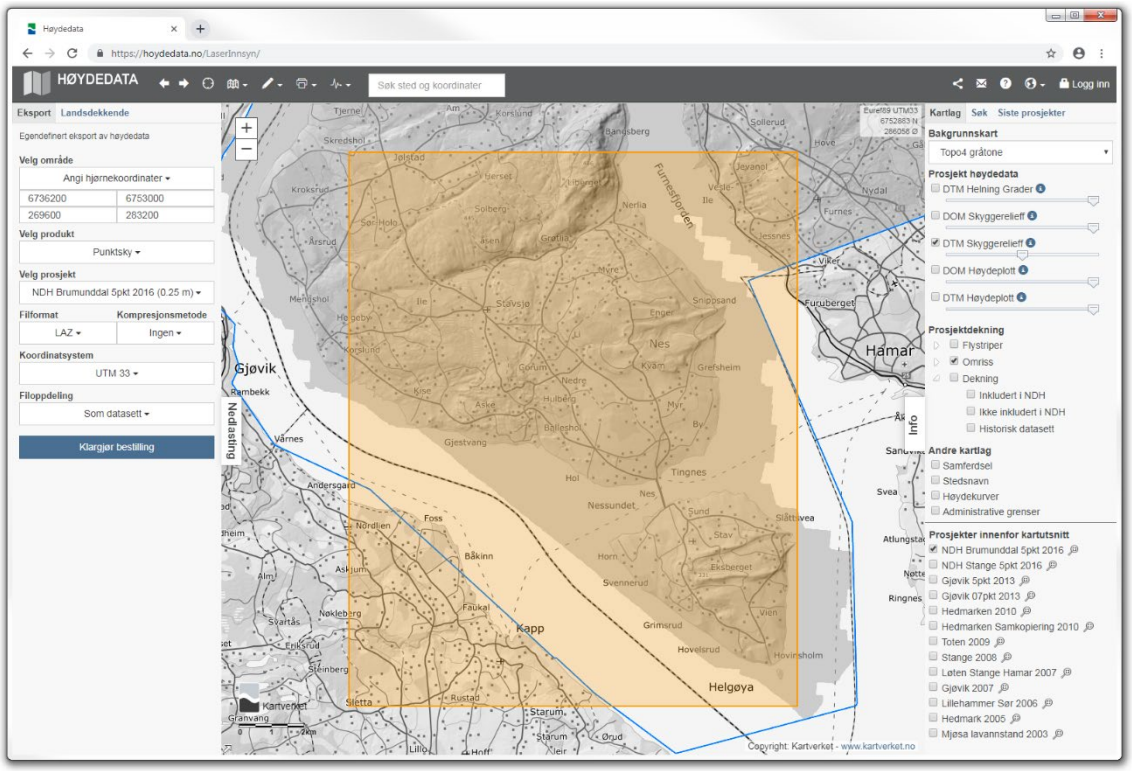



Figure 32. Brumunddal 2016 dataset, test subset.

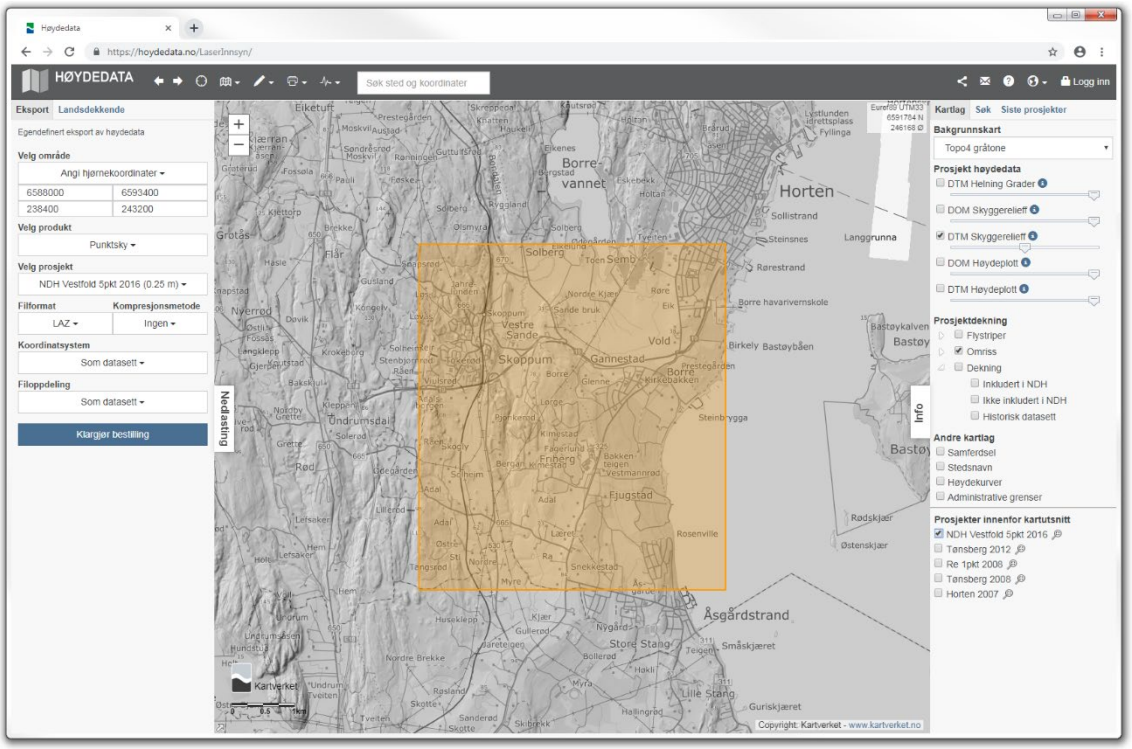



Figure 33. Horten 2016 dataset.

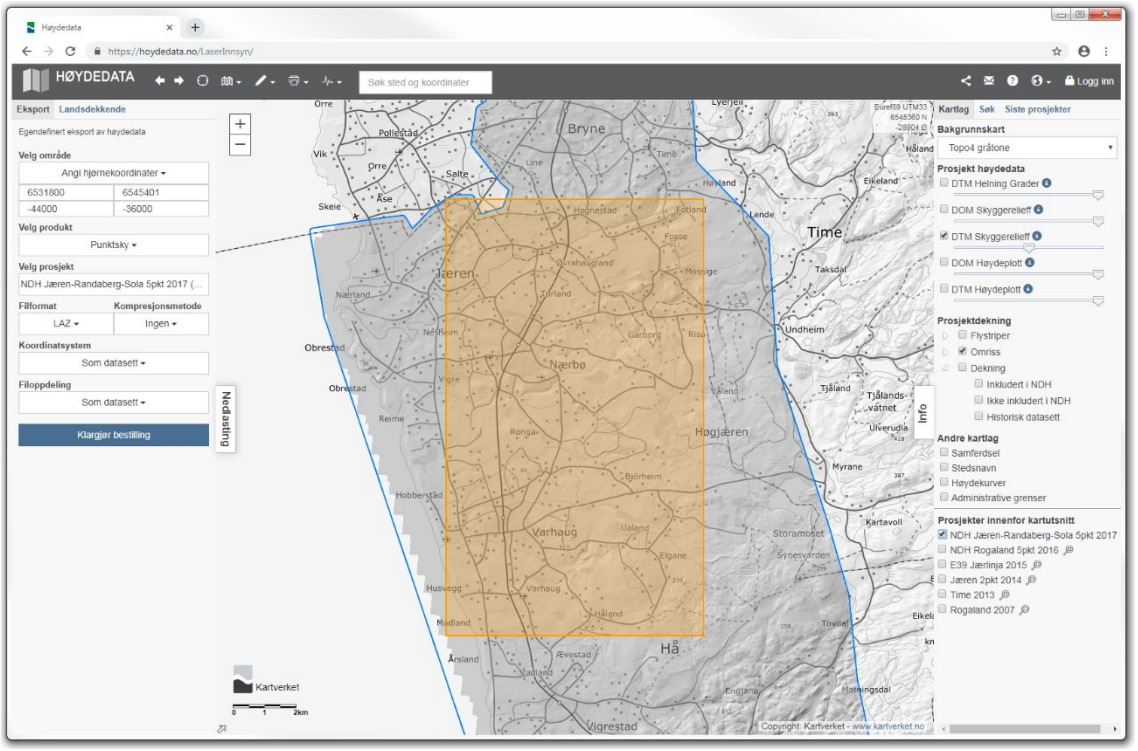



Figure 34. Hå Jæren 2017 dataset.

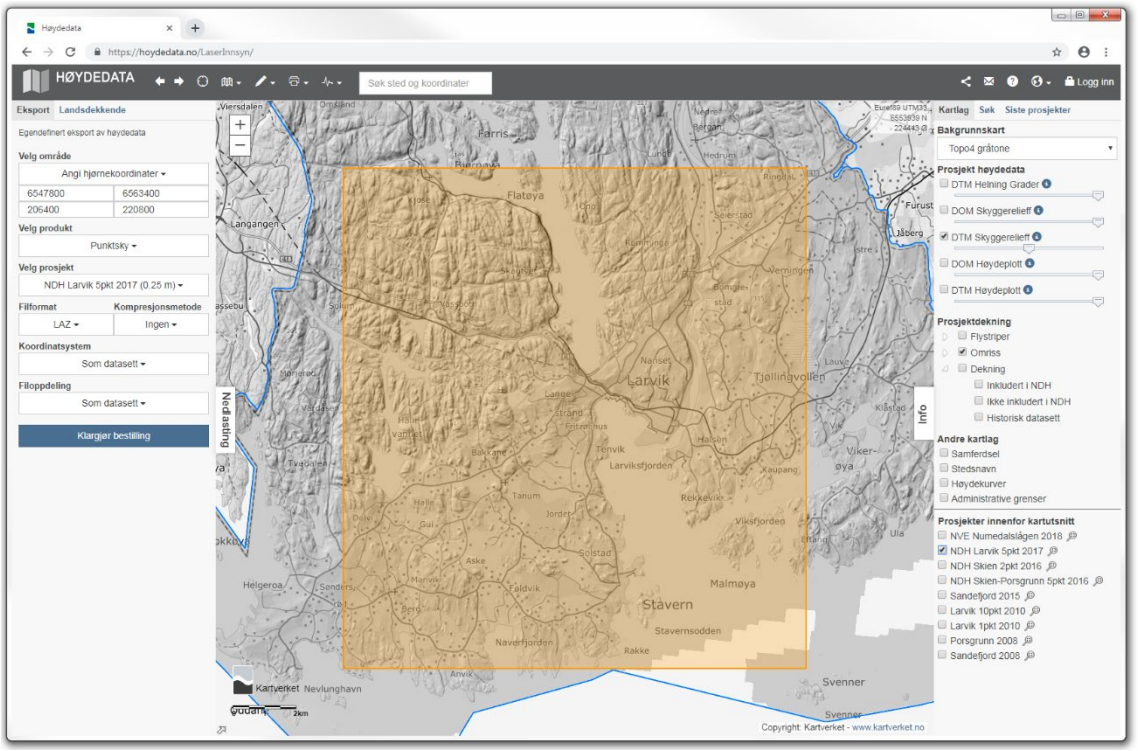



Figure 35. Larvik 2017 dataset, training subset.

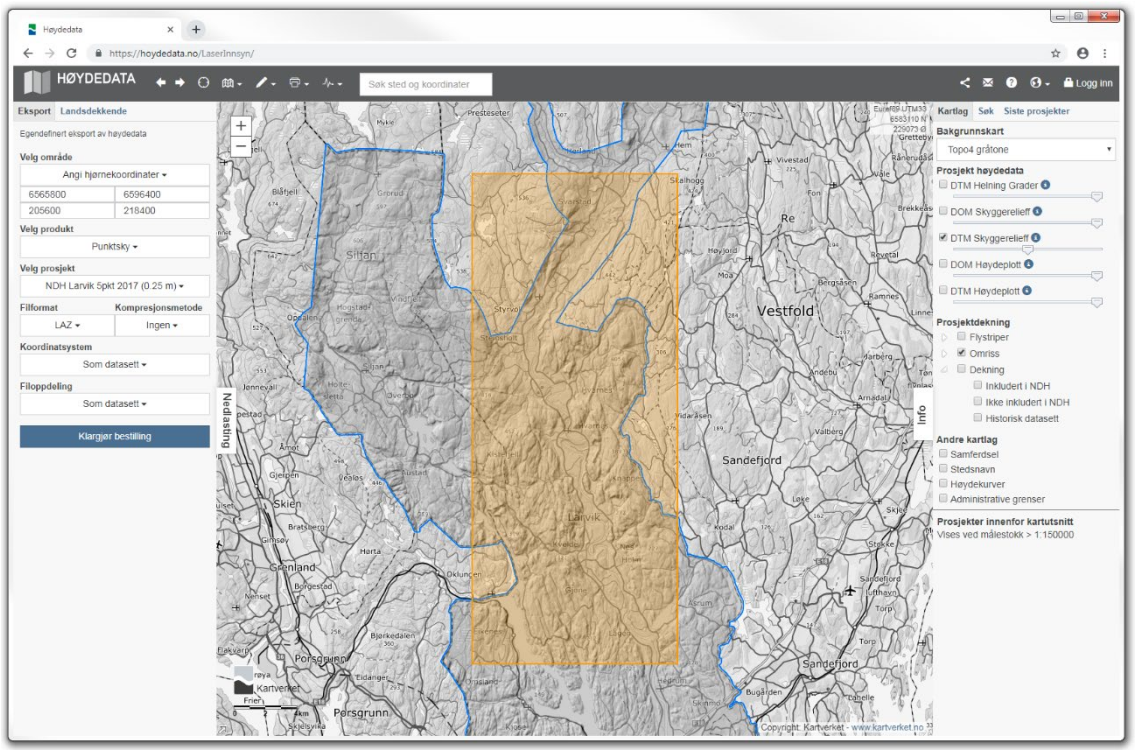



Figure 36. Larvik 2017 dataset, validation subset.

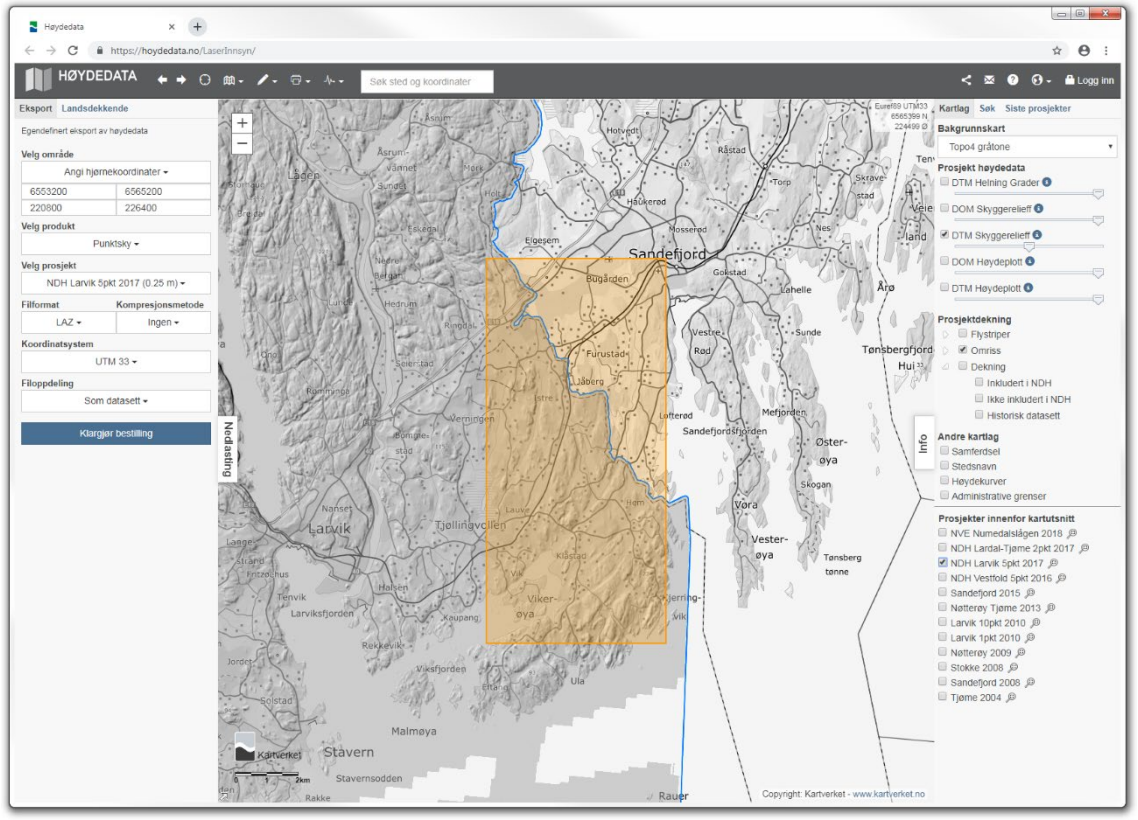



Figure 37. Larvik 2017 dataset, test subset.

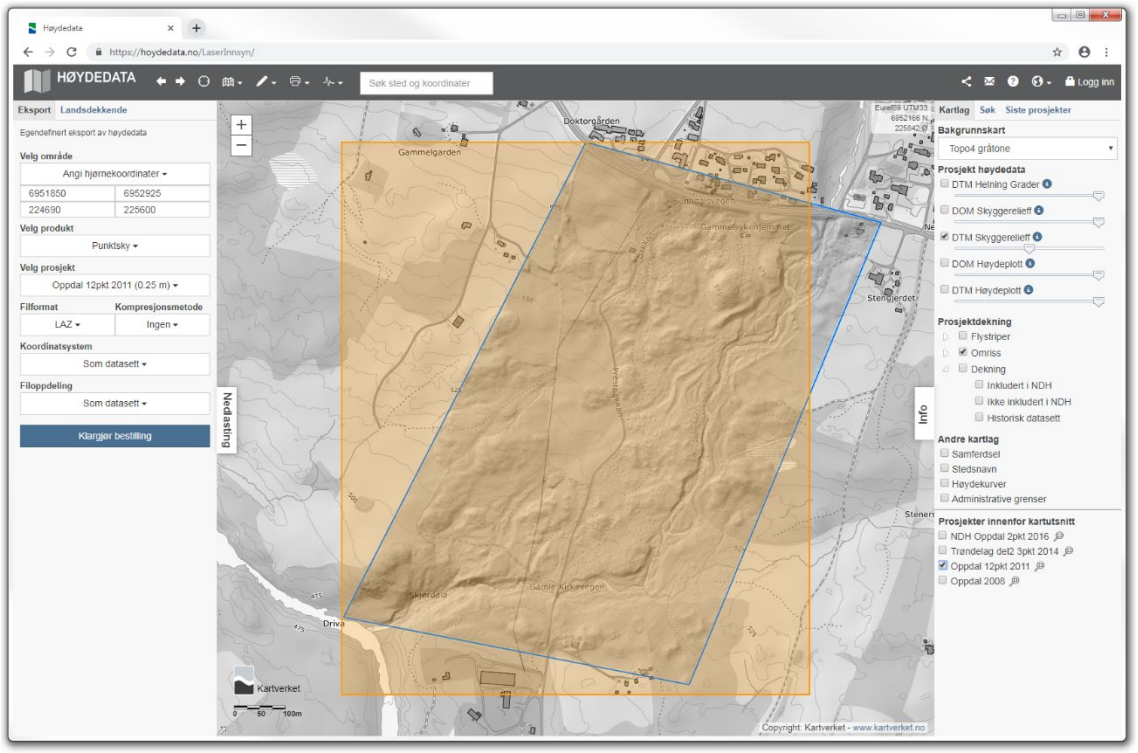



Figure 38. Oppdal Vang 2011 dataset.

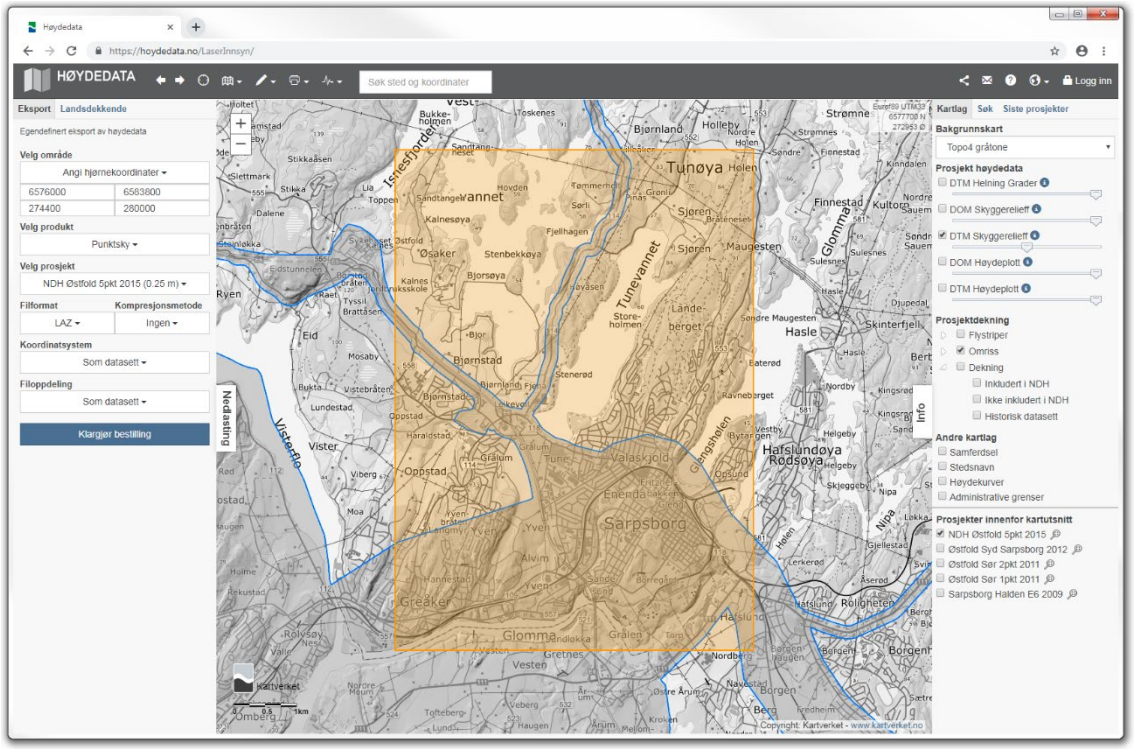



Figure 39. Sarpsborg 2015 dataset, validation subset.

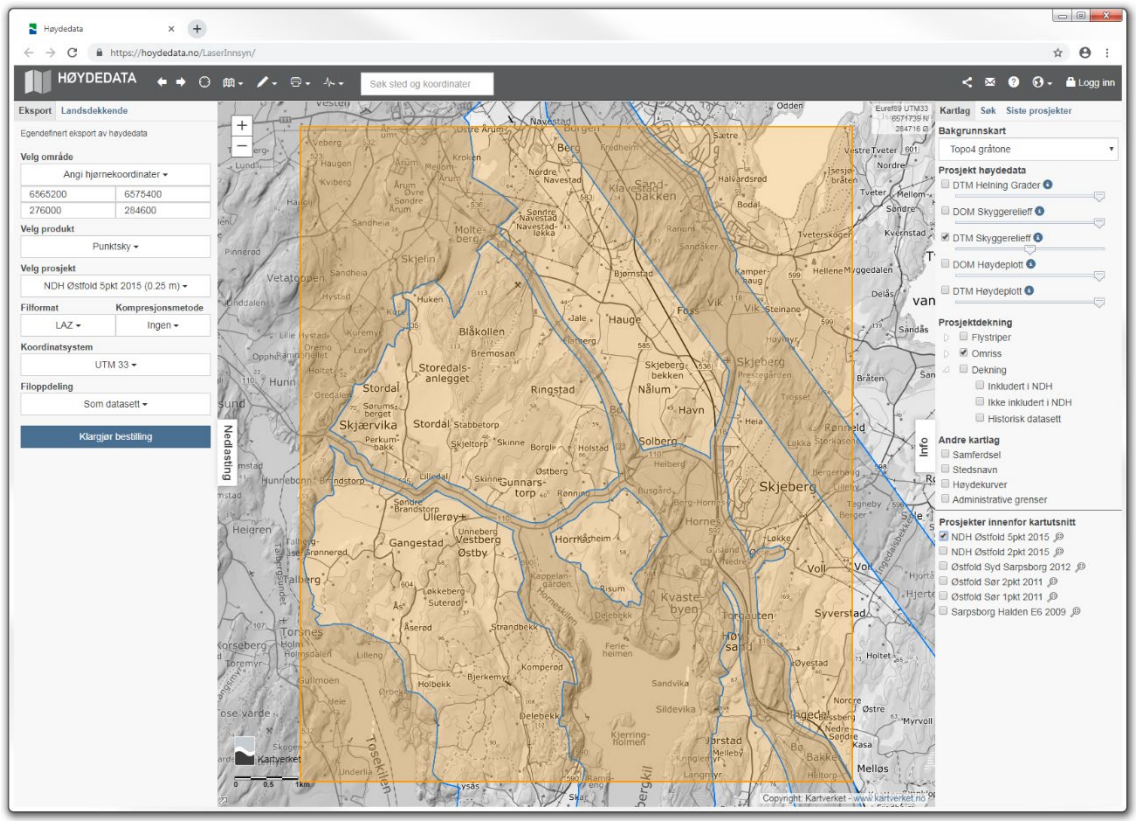



Figure 40. Sarpsborg 2015 dataset, test subset.

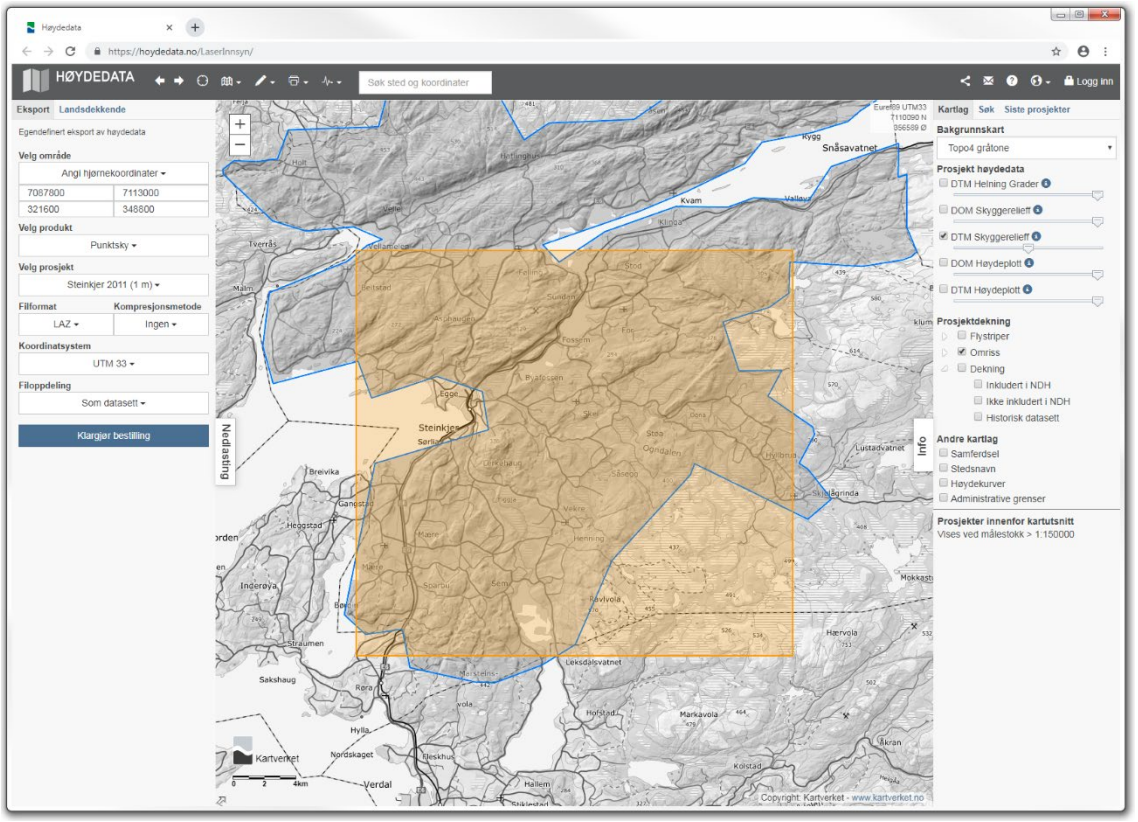



Figure 41. Steinkjer 2011 dataset.

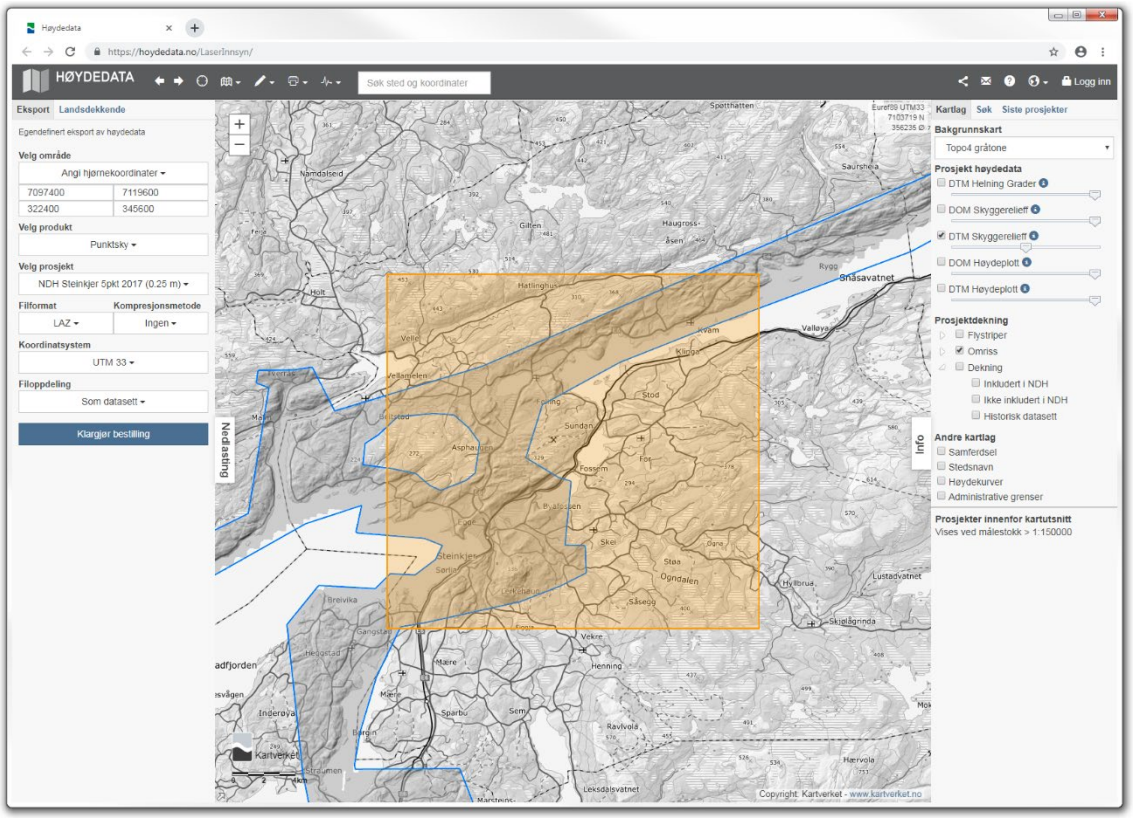



Figure 42. Steinkjer 2017 dataset.

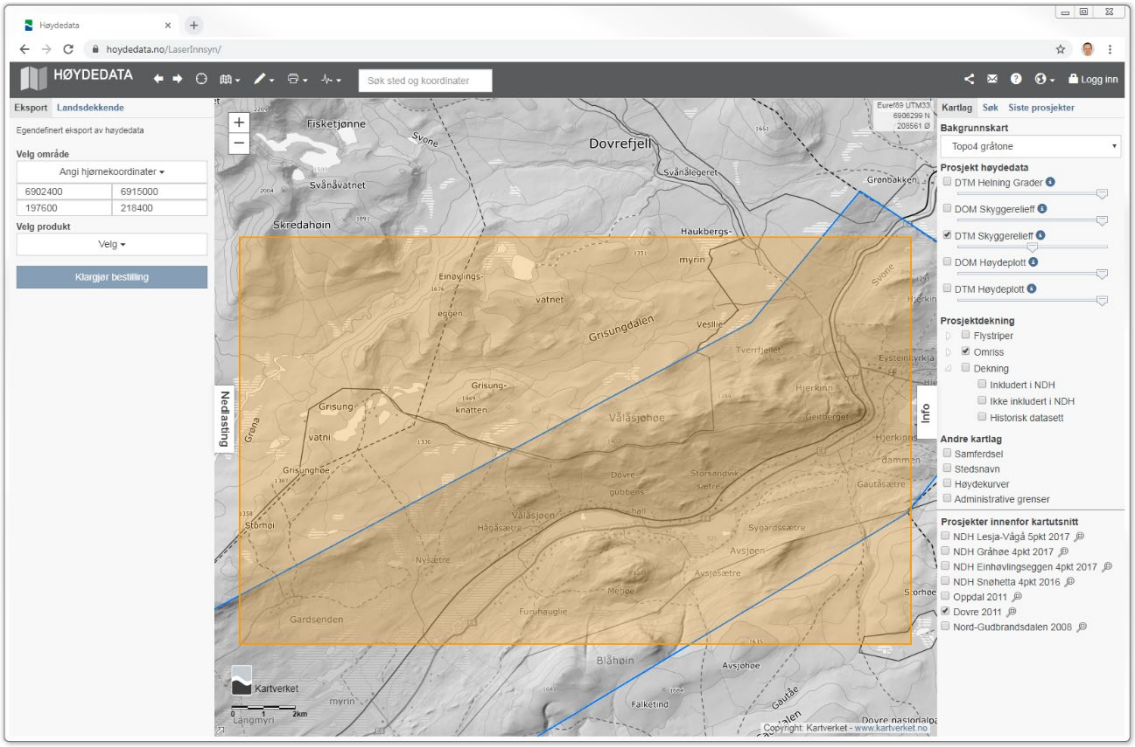



Figure 43. Dovre 2011 dataset, training subset.

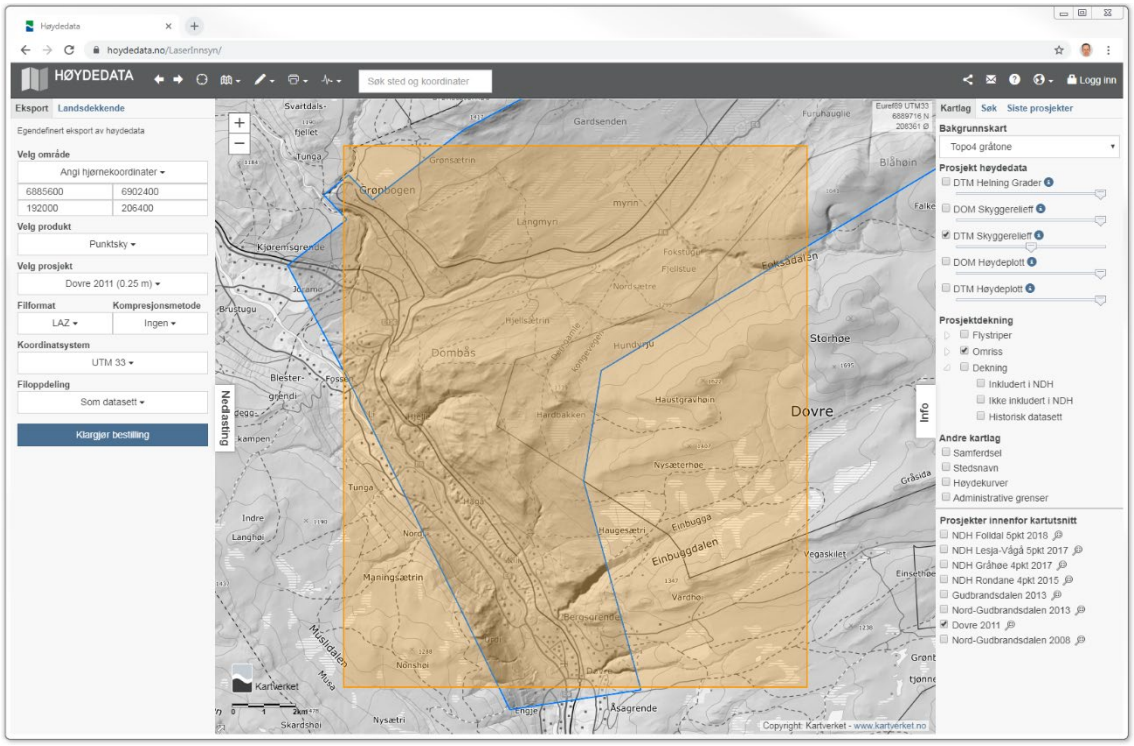



Figure 44. Dovre 2011 dataset, validation subset.

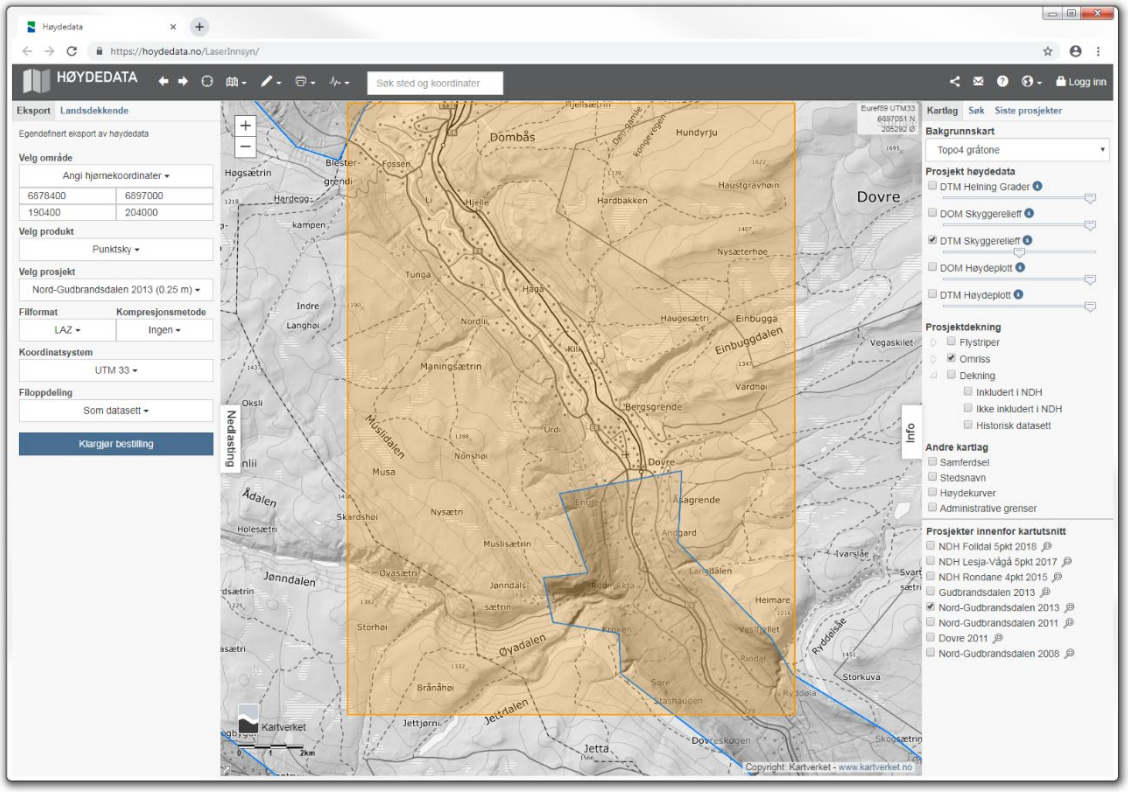



Figure 45. Dovre 2013 dataset.

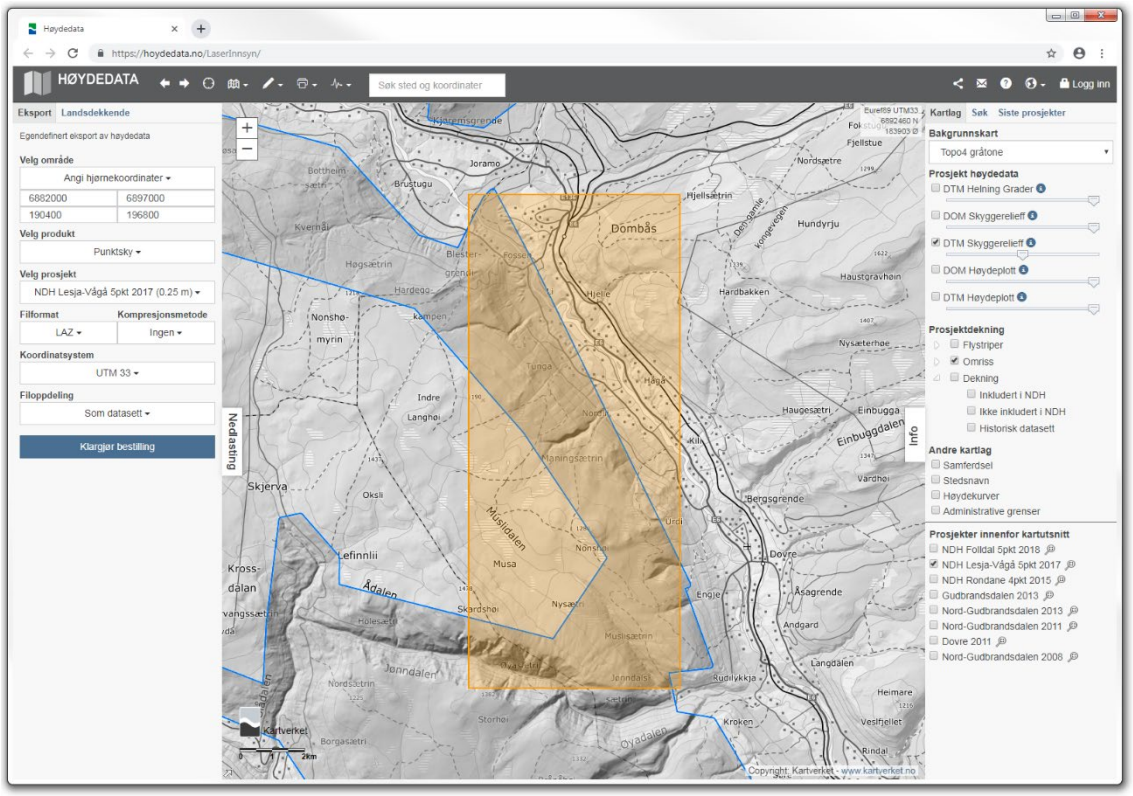



Figure 46. Dovre 2017 dataset.

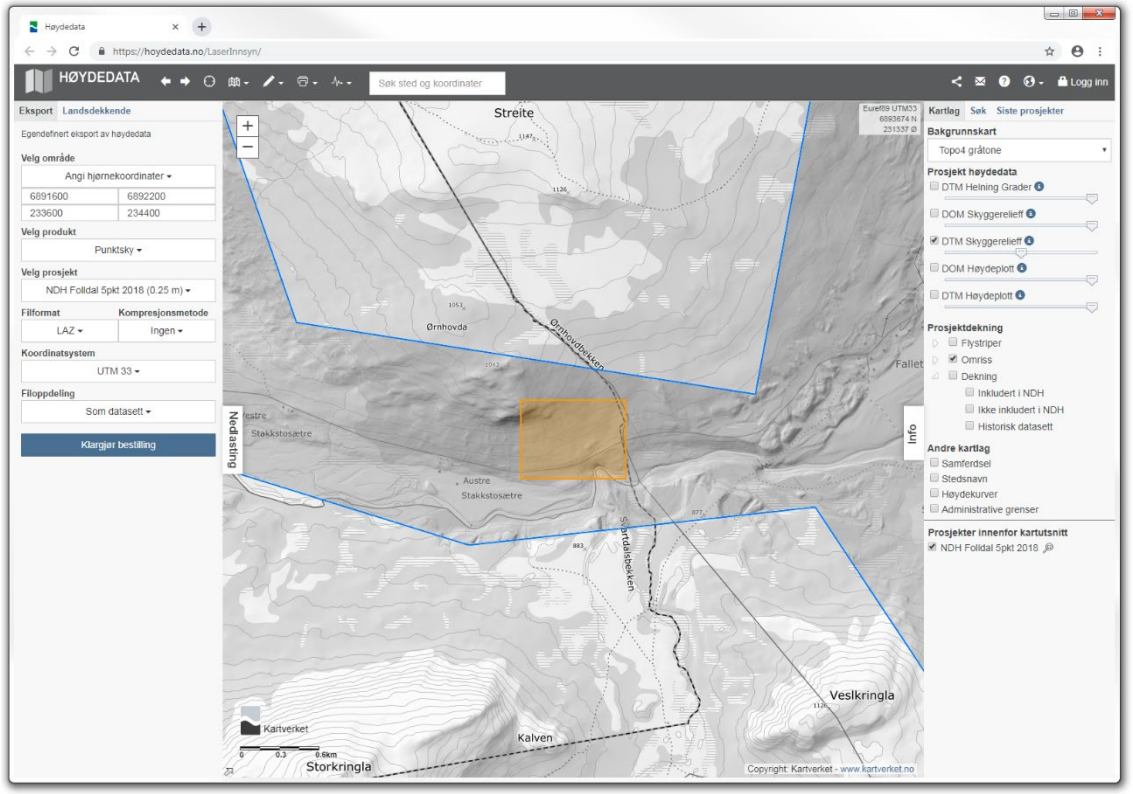



Figure 47. Dovre Folldal 2018 dataset.

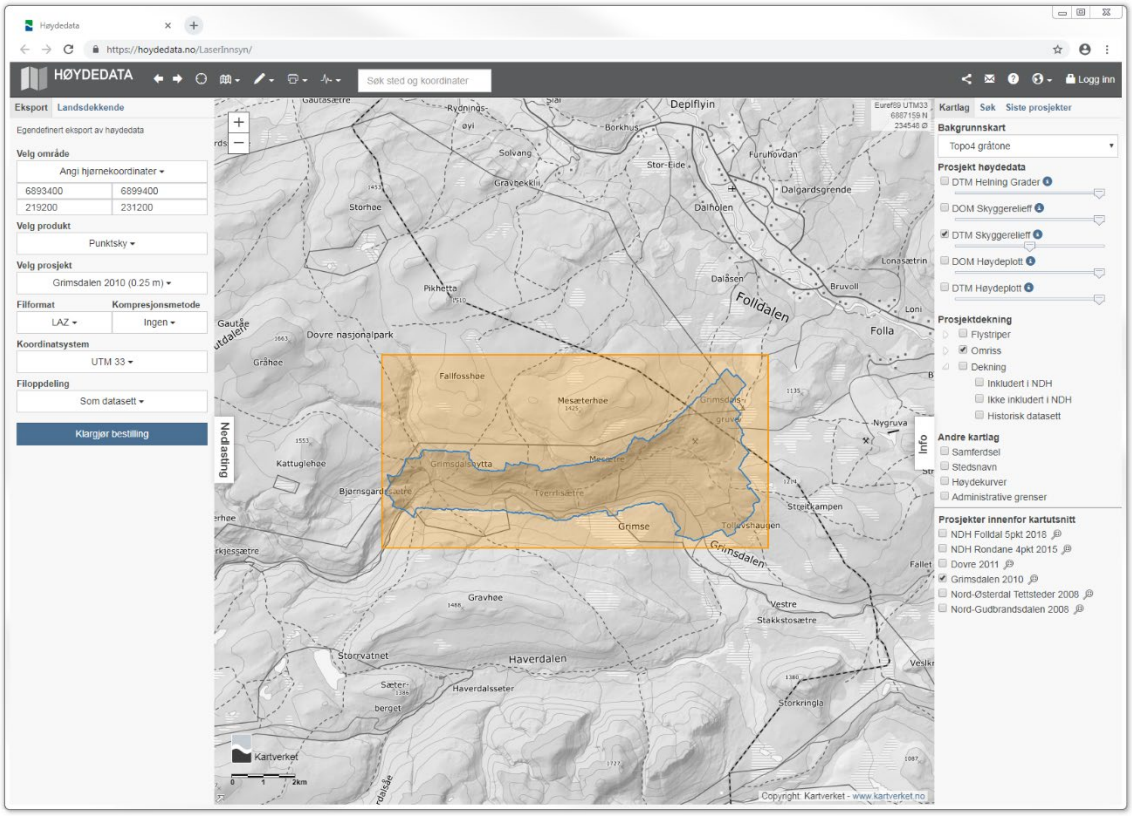



Figure 48. Dovre Grimsdalen 2010 dataset.

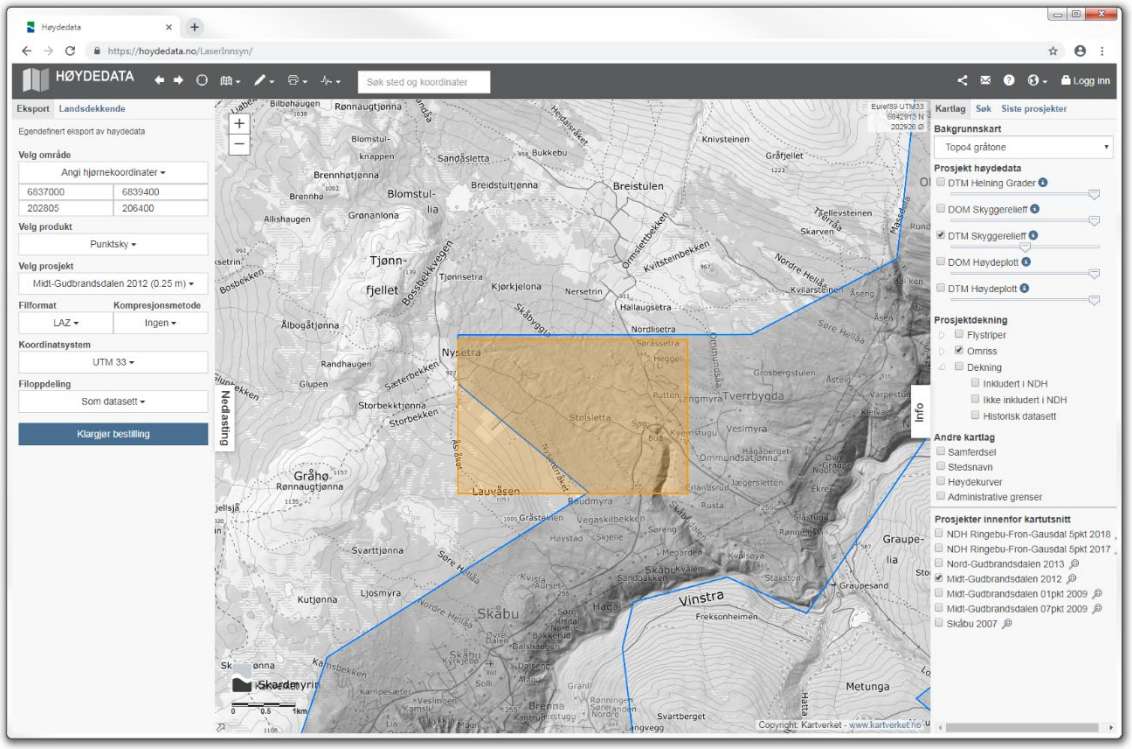



Figure 49. Nordfron 2012 dataset, training subset.

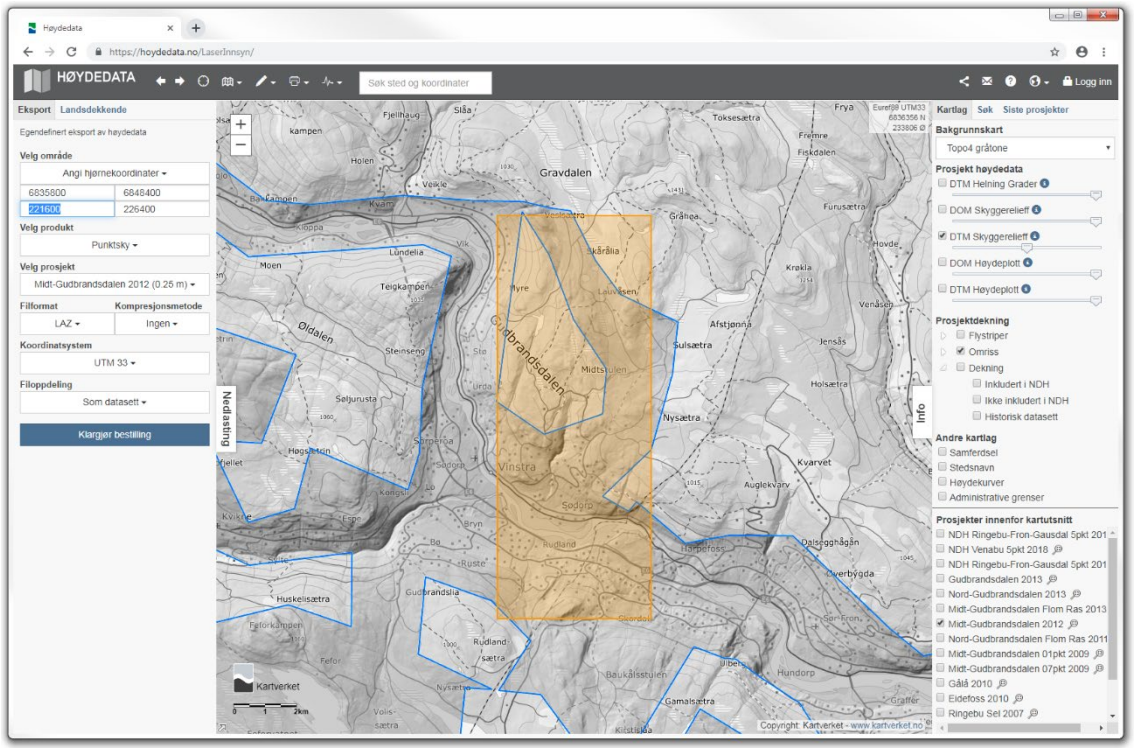



Figure 50. Nordfron 2012 dataset, validation subset.

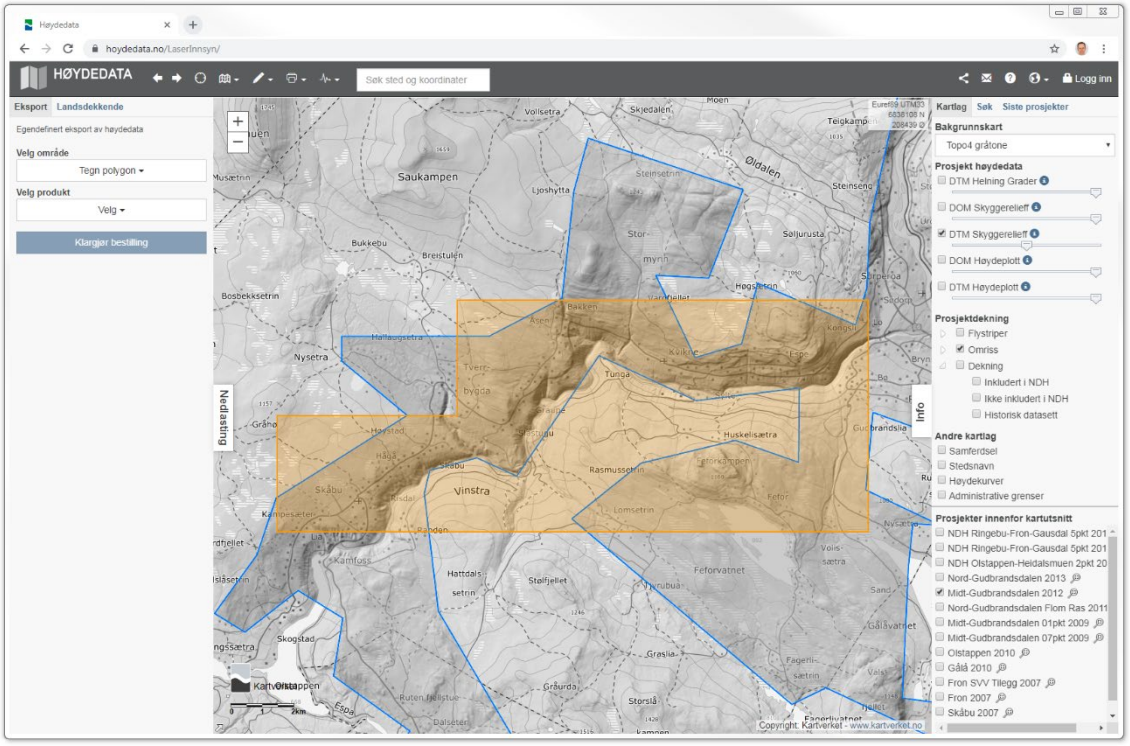



Figure 51. Nordfron 2012 dataset, test subset.

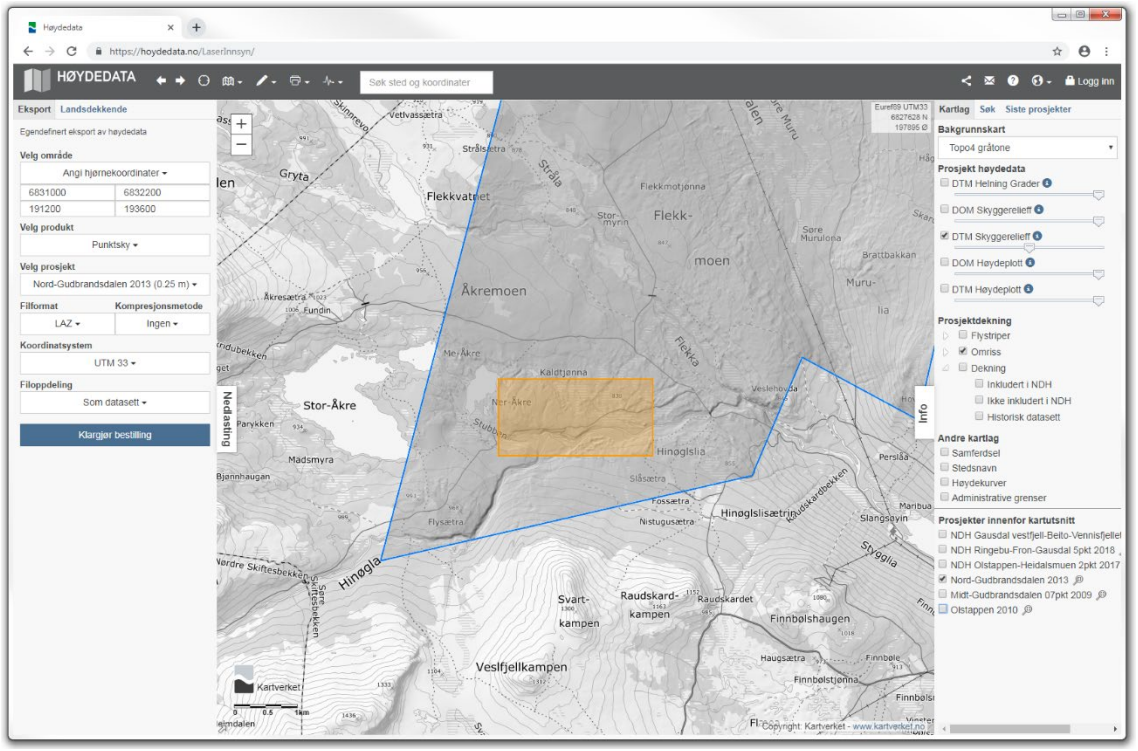



Figure 52. Nordfron 2013 dataset, training subset.

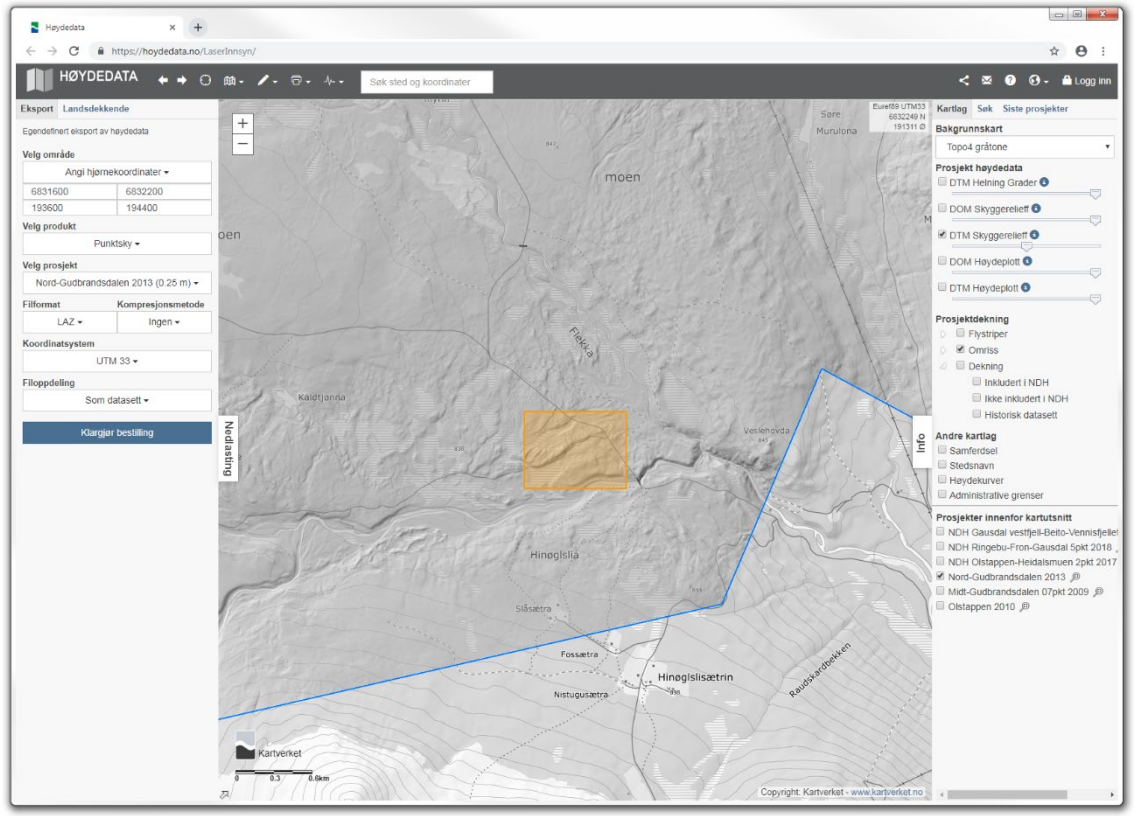



Figure 53. Nordfron 2013 dataset, test subset.

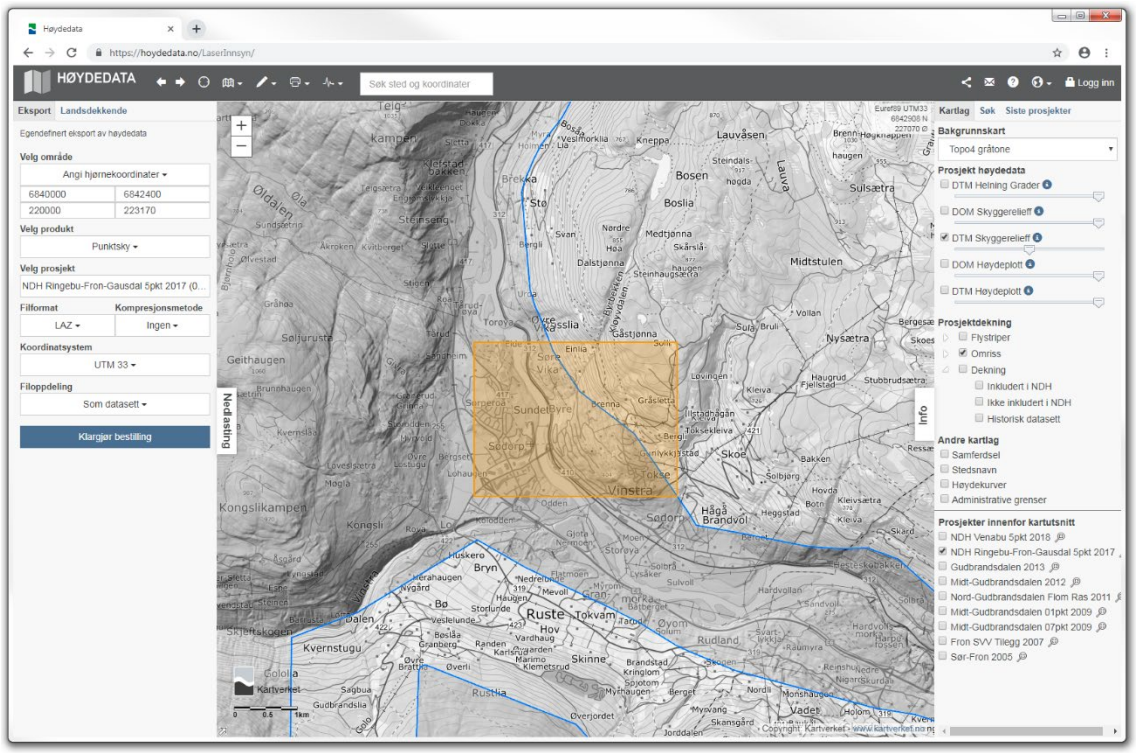



Figure 54. Nordfron 2017 dataset, training subset.

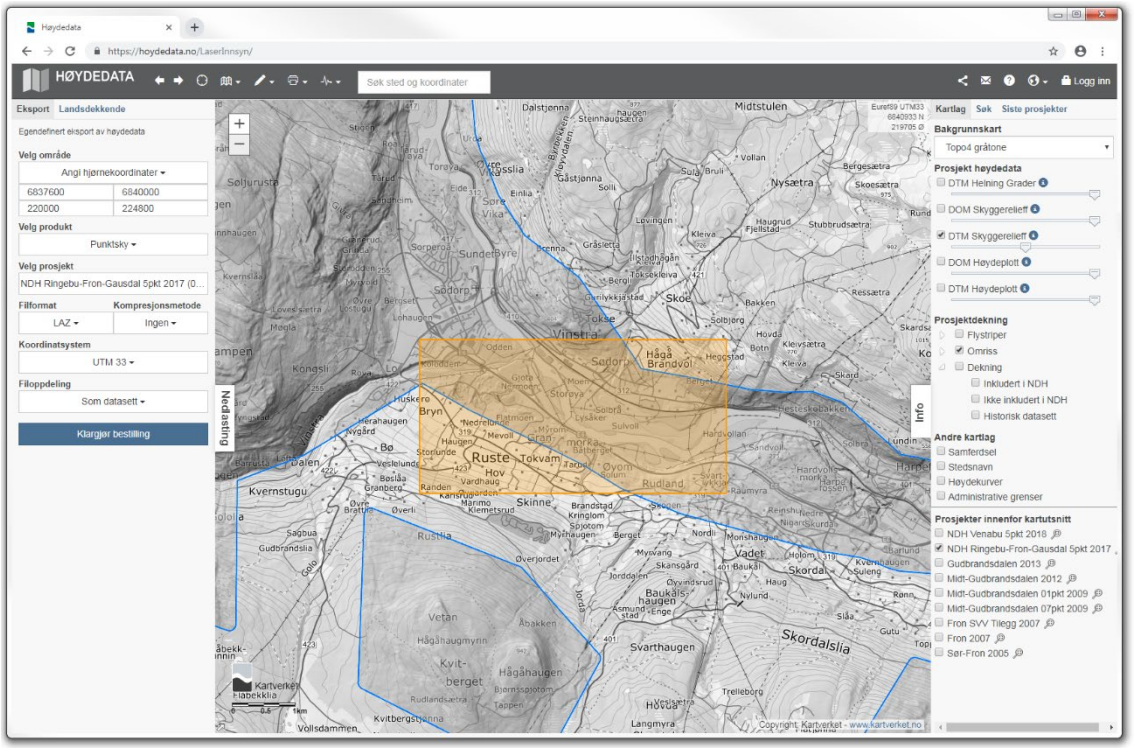



Figure 55. Nordfron 2017 dataset, validation subset.

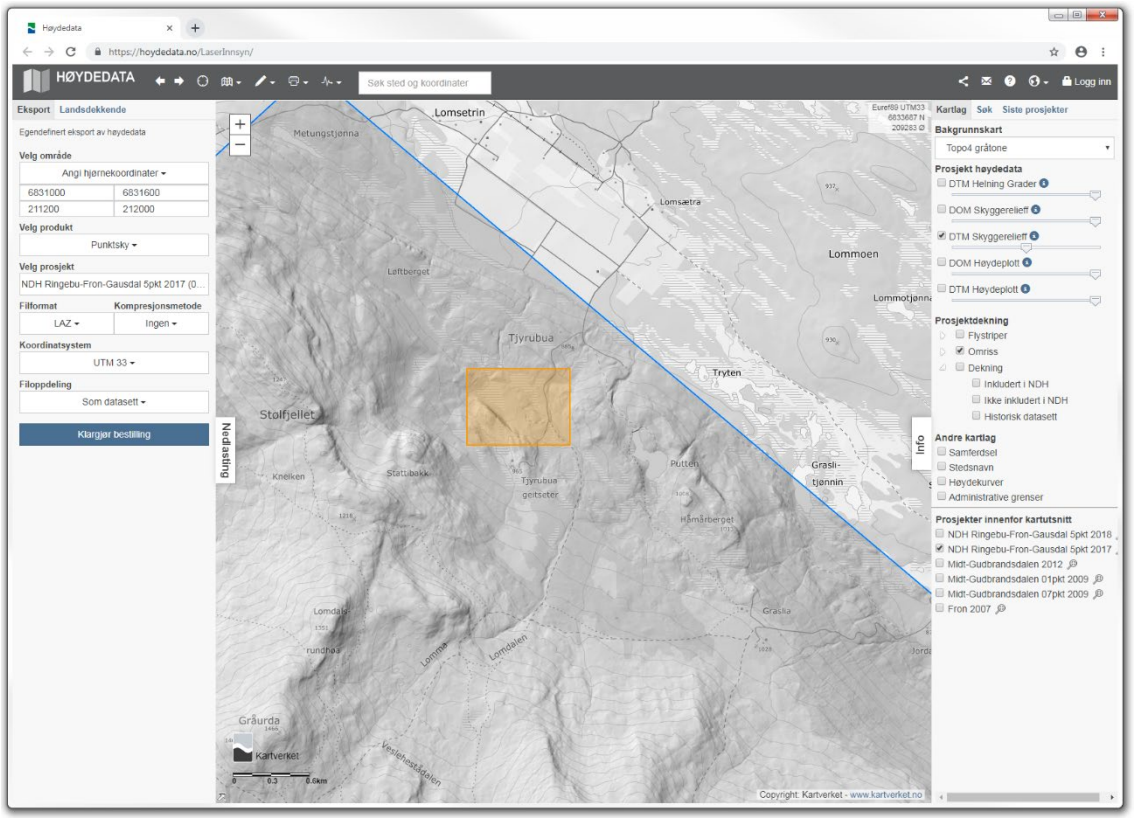



Figure 56. Nordfron 2017 dataset, test subset.

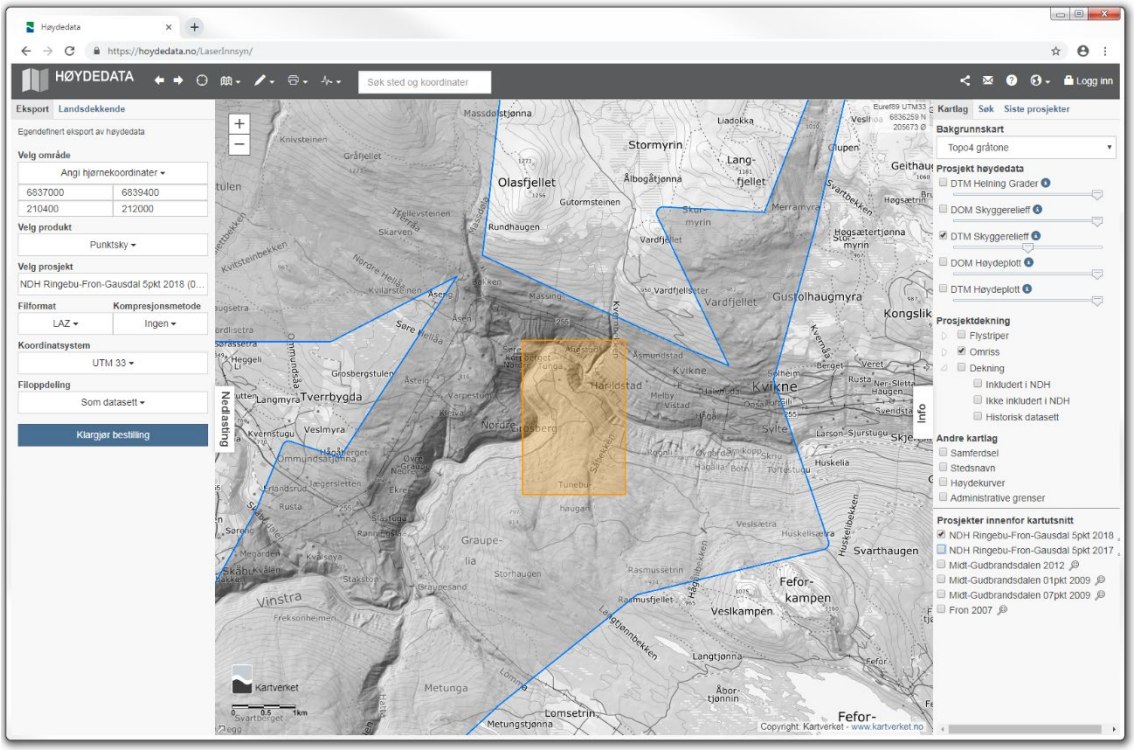



Figure 57. Nordfron 2018 dataset, training subset.

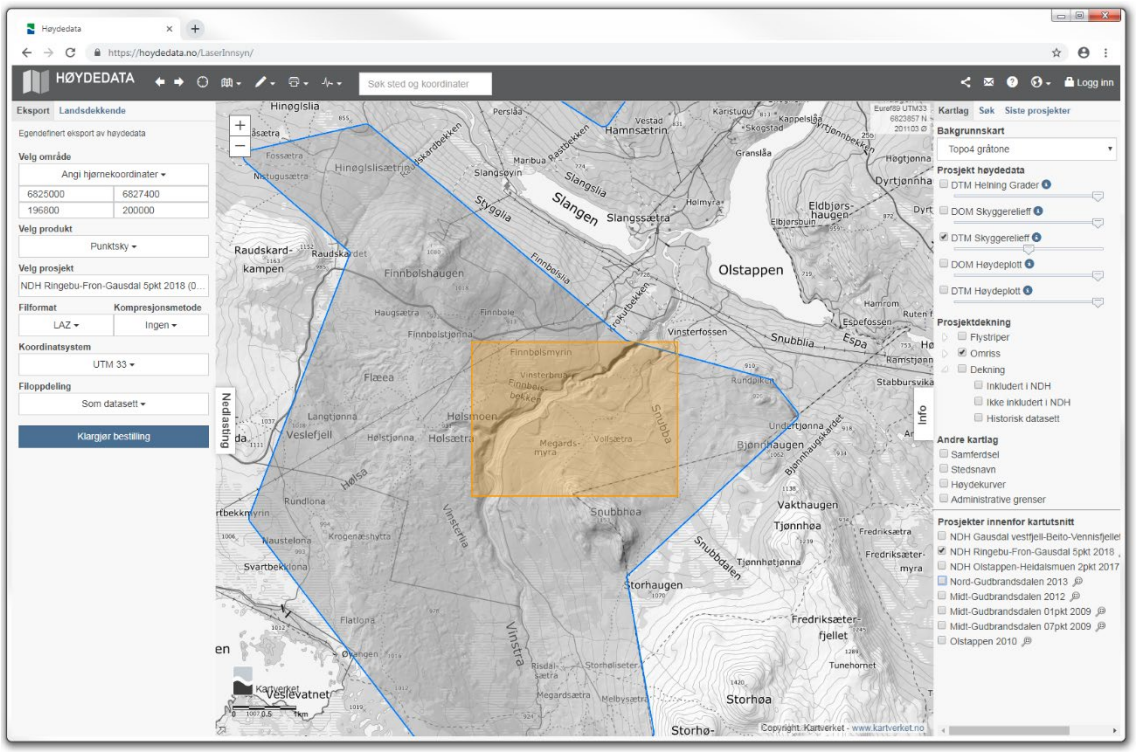



Figure 58. Nordfron 2018 dataset, validation subset.

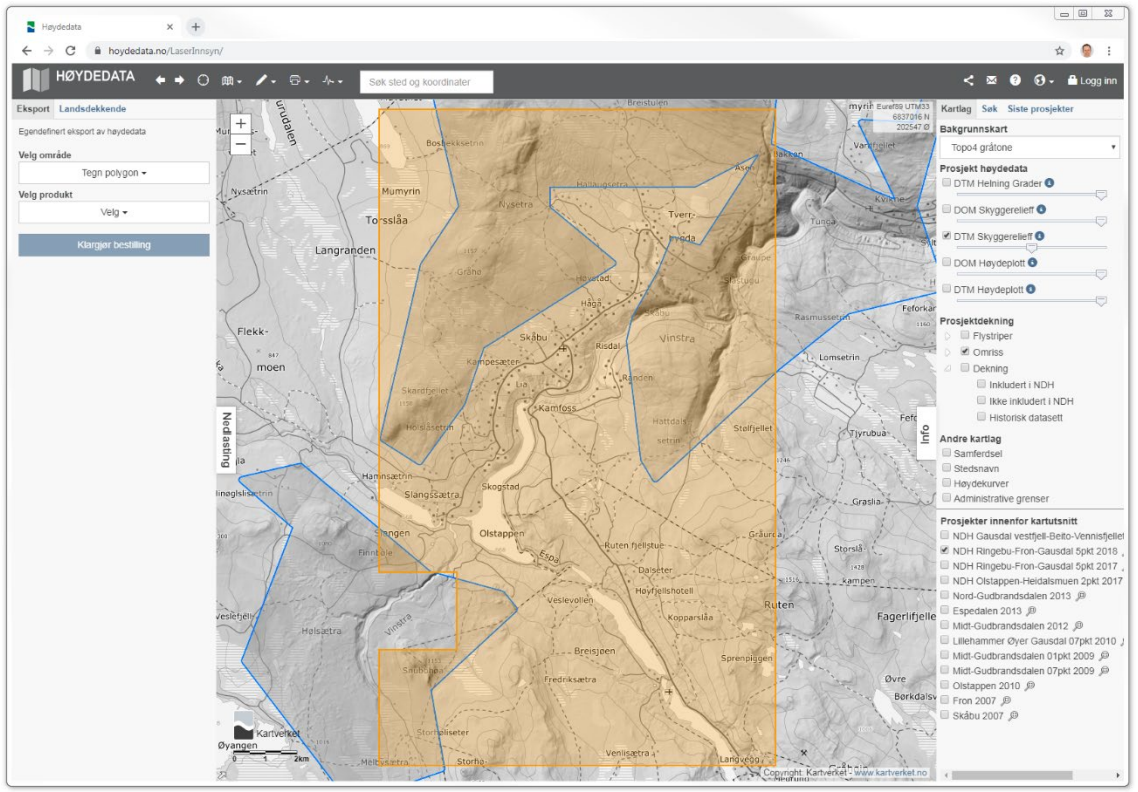



Figure 59. Nordfron 2018 dataset, test subset.

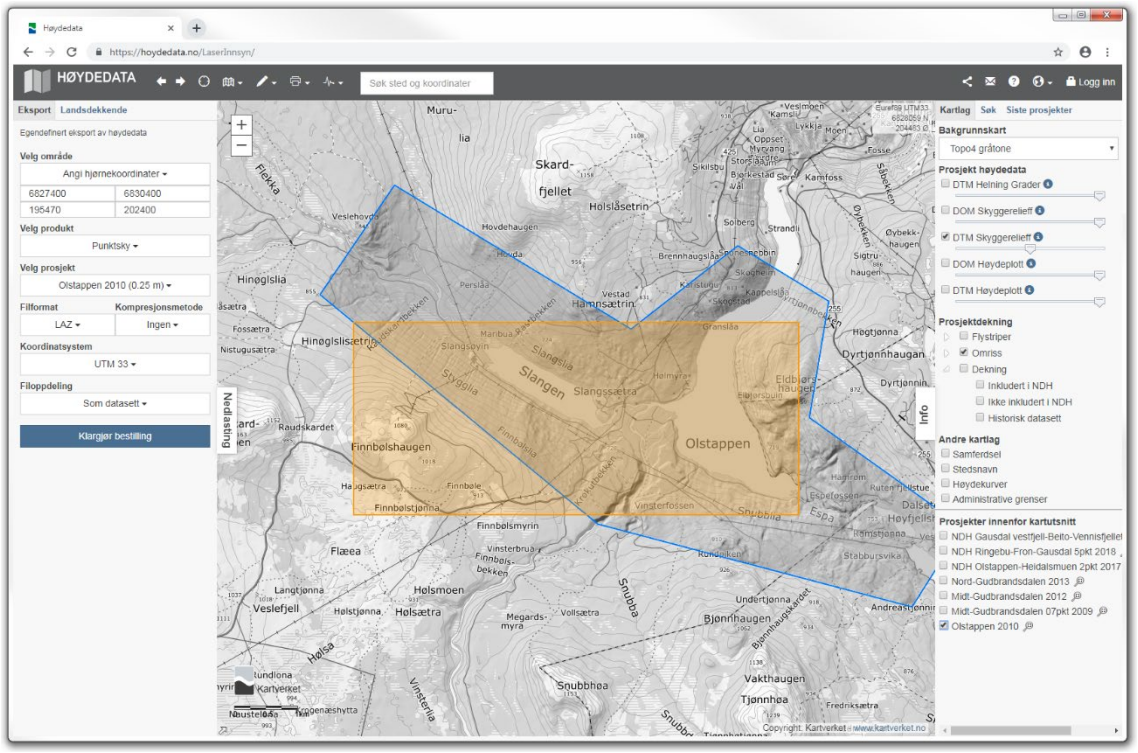



Figure 60. Nordfron Olstappen 2010 dataset, training subset.

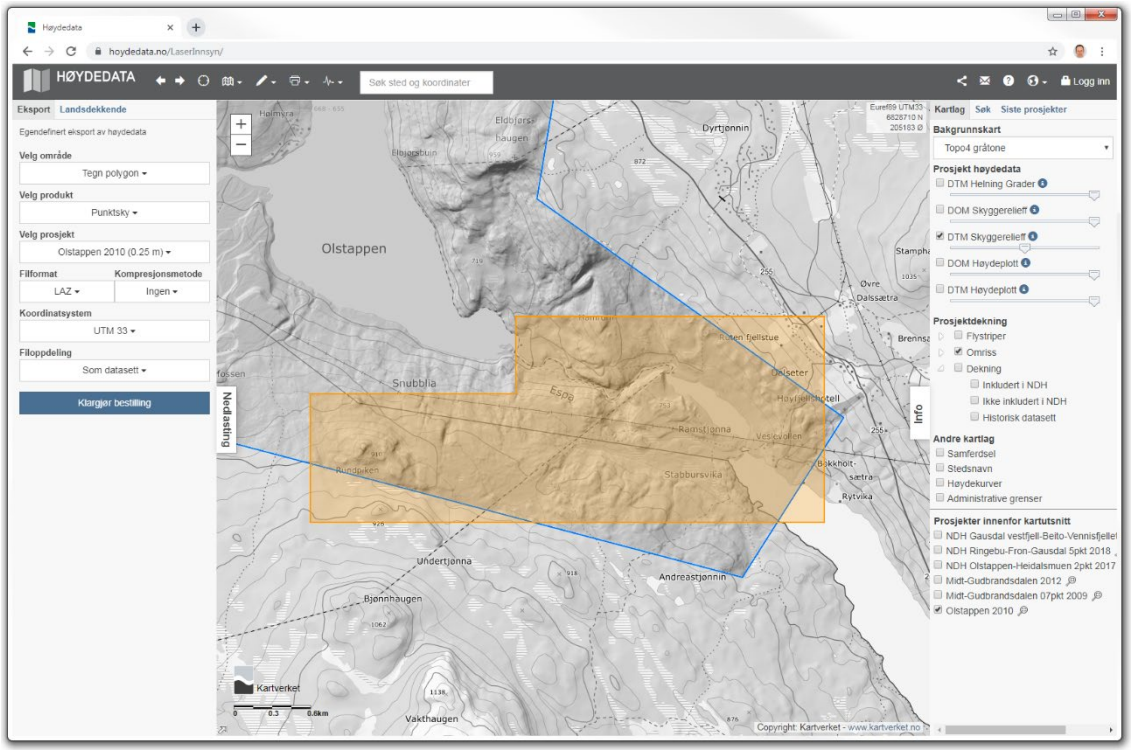



Figure 61. Nordfron Olstappen 2010 dataset, validation subset.

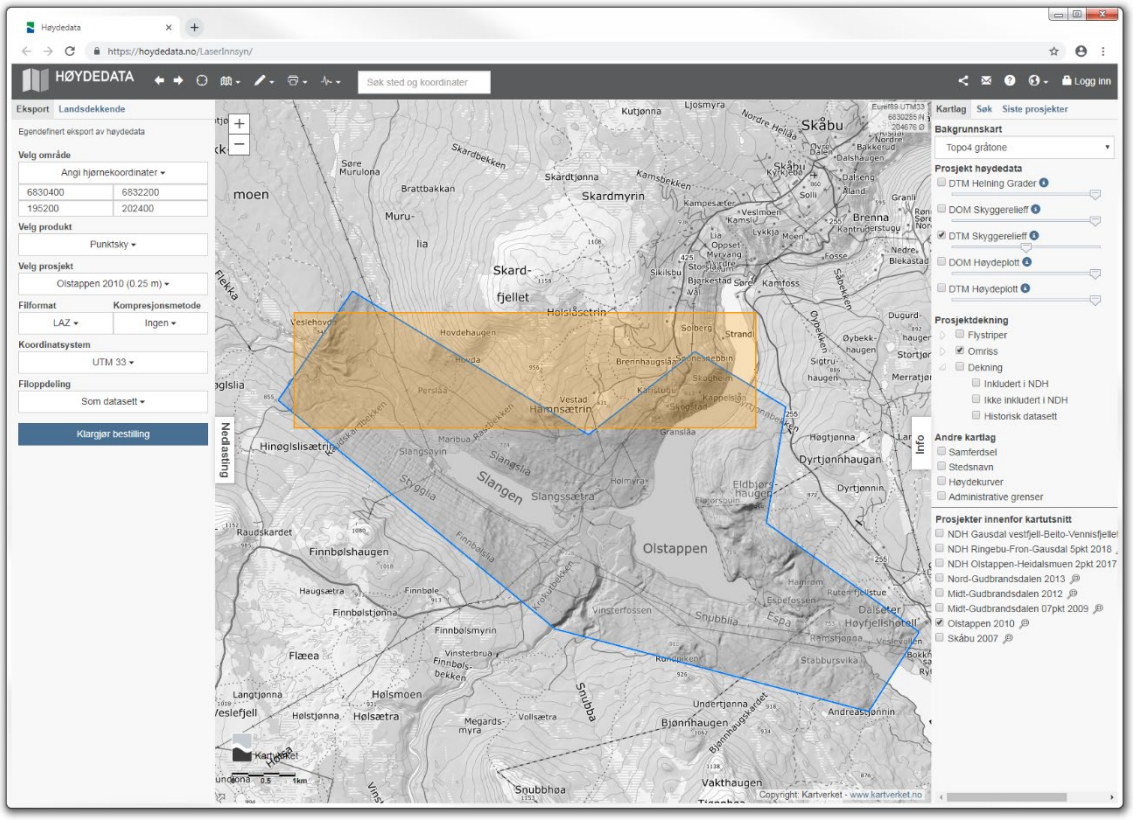



Figure 62. Nordfron Olstappen dataset, test subset.

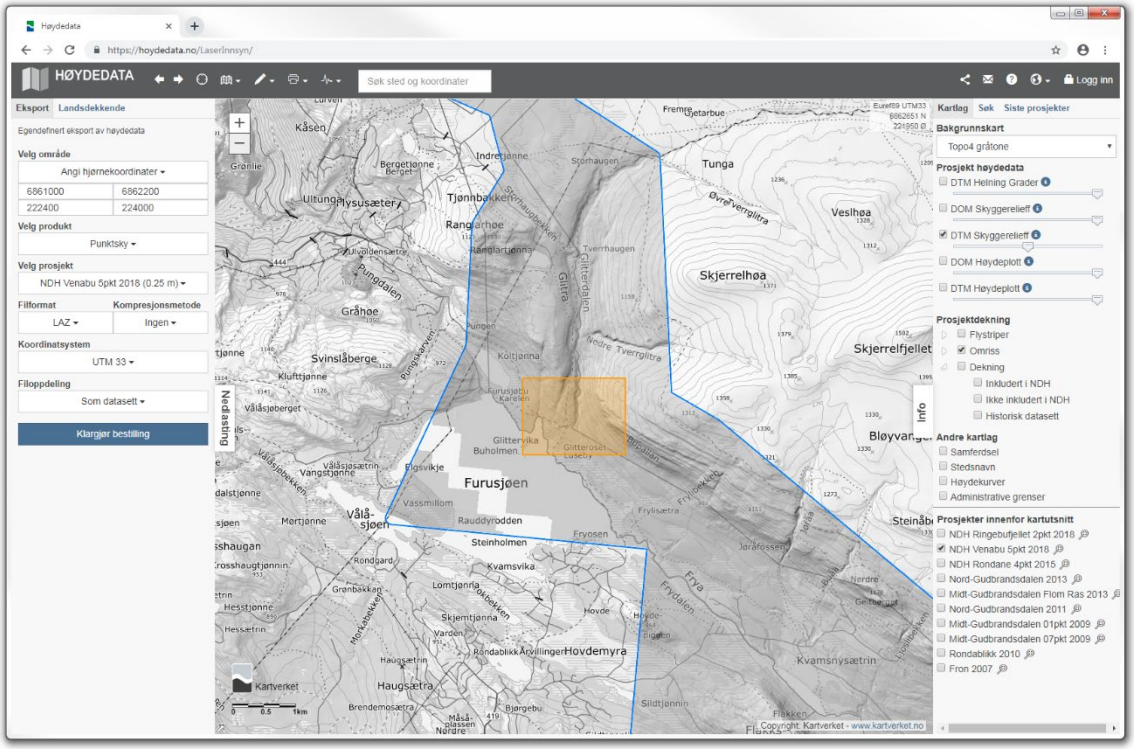



Figure 63. Nordfron Venabu 2018 dataset, validation subset.

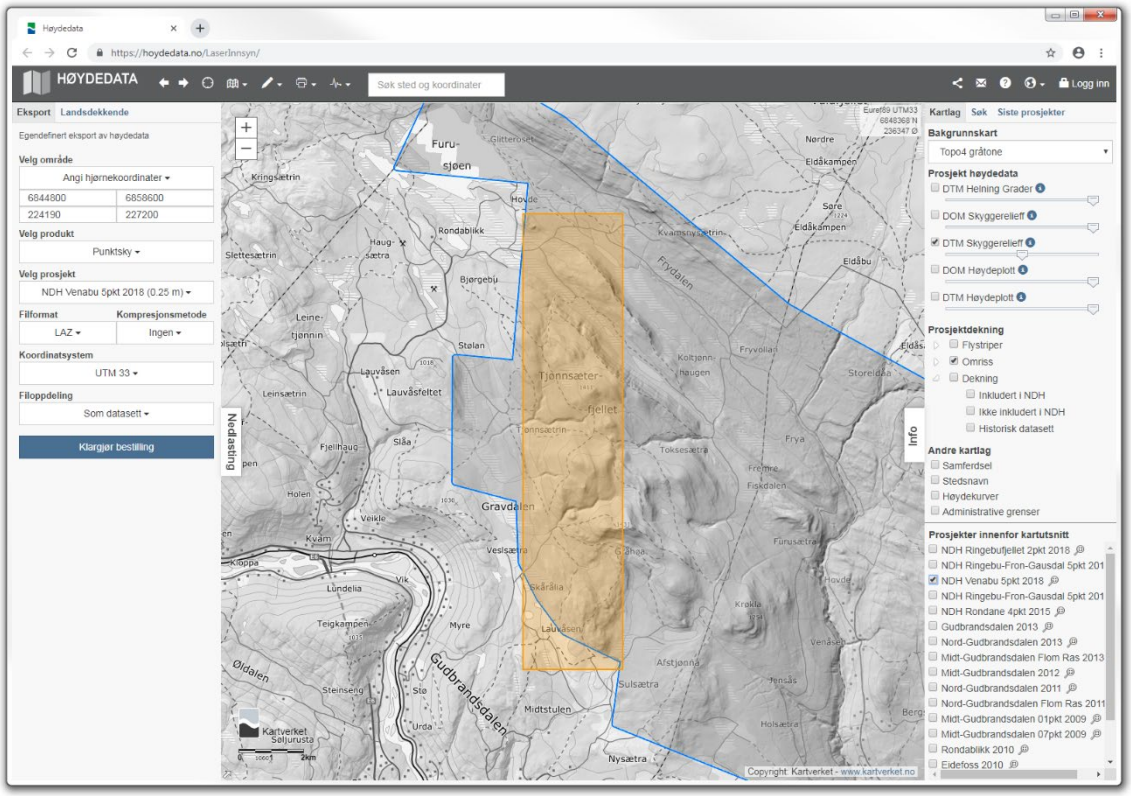



Figure 64. Nordfron Venabu 2018 dataset, test subset.

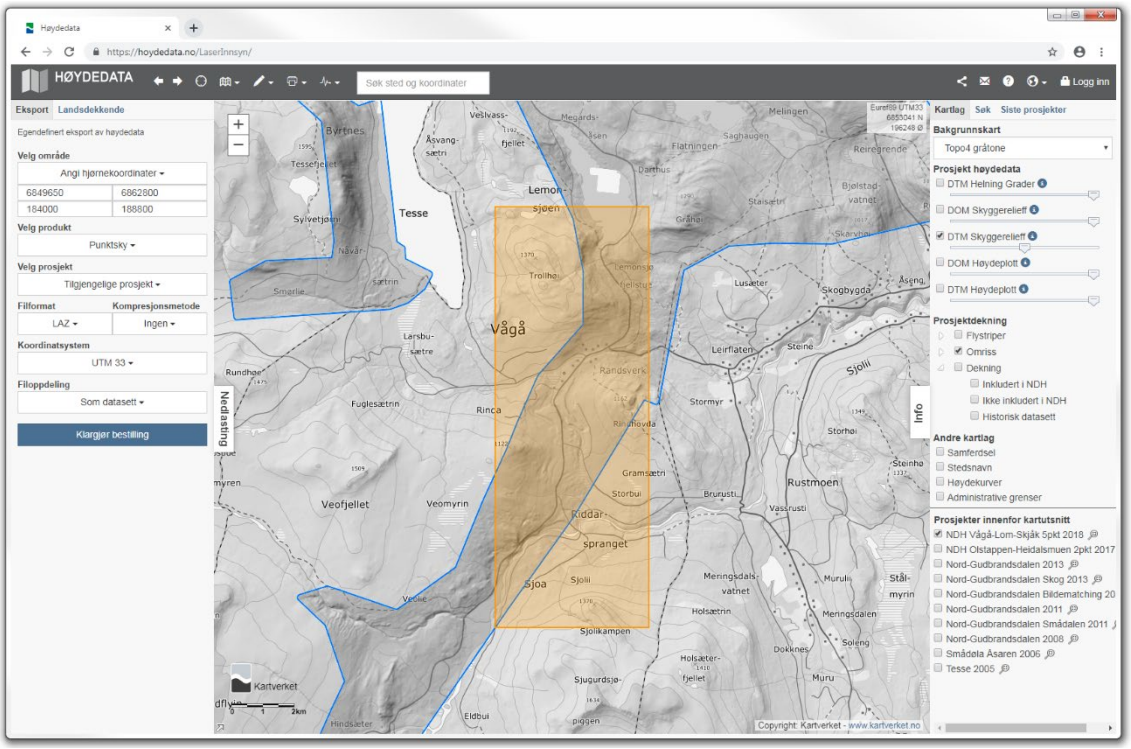



Figure 65. Vågå 2018 dataset, training subset.

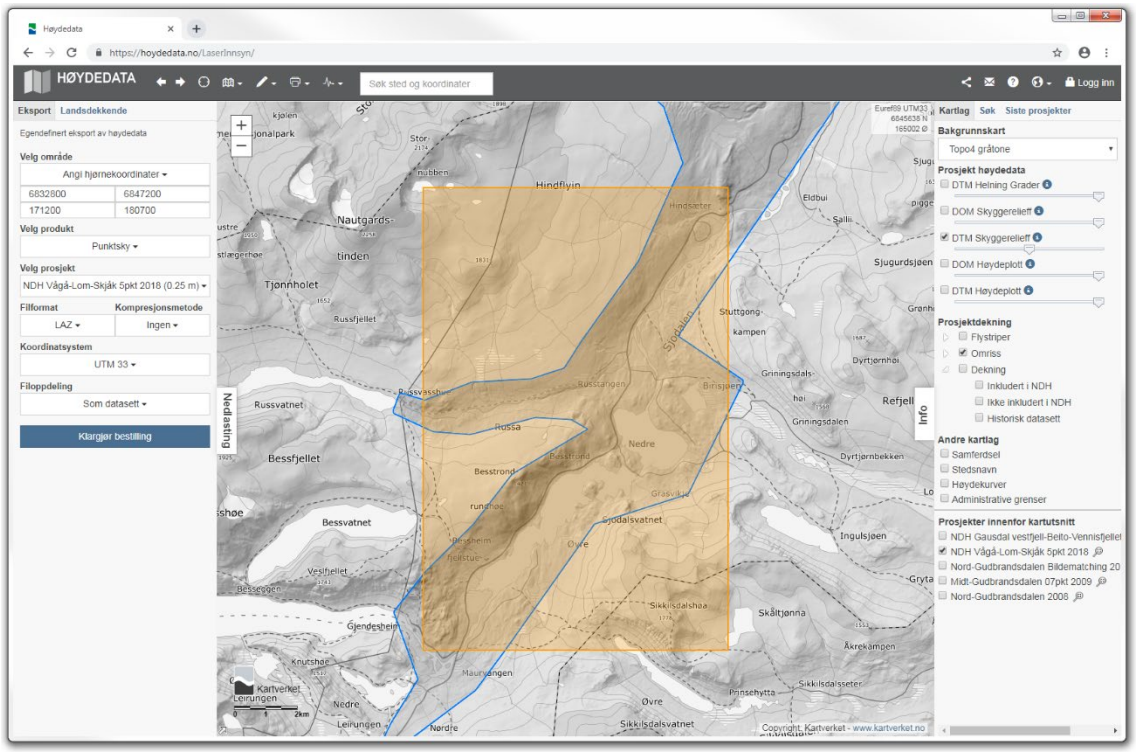



Figure 66. Vågå 2018 dataset, validation subset.

### 3.4.3 Unlabelled test data

Overview maps of the ALS datasets (Table 7) appear below (Figure 67-Figure 69).

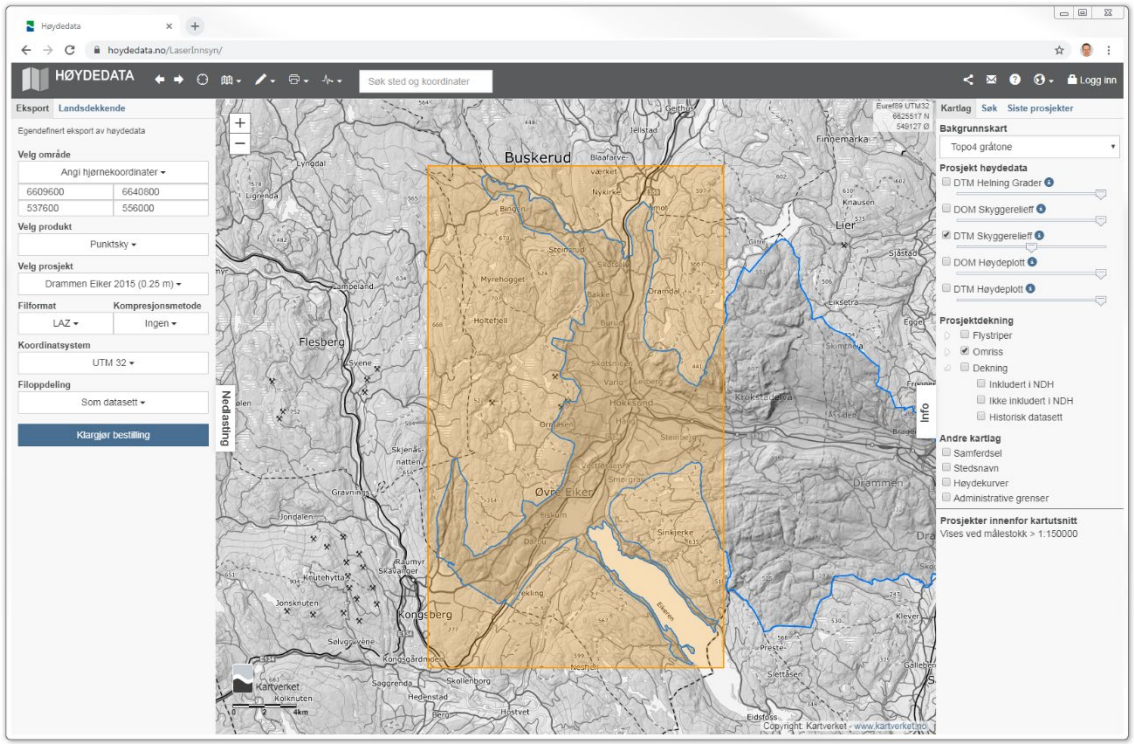



Figure 67. Øvre Eiker 2015 dataset.

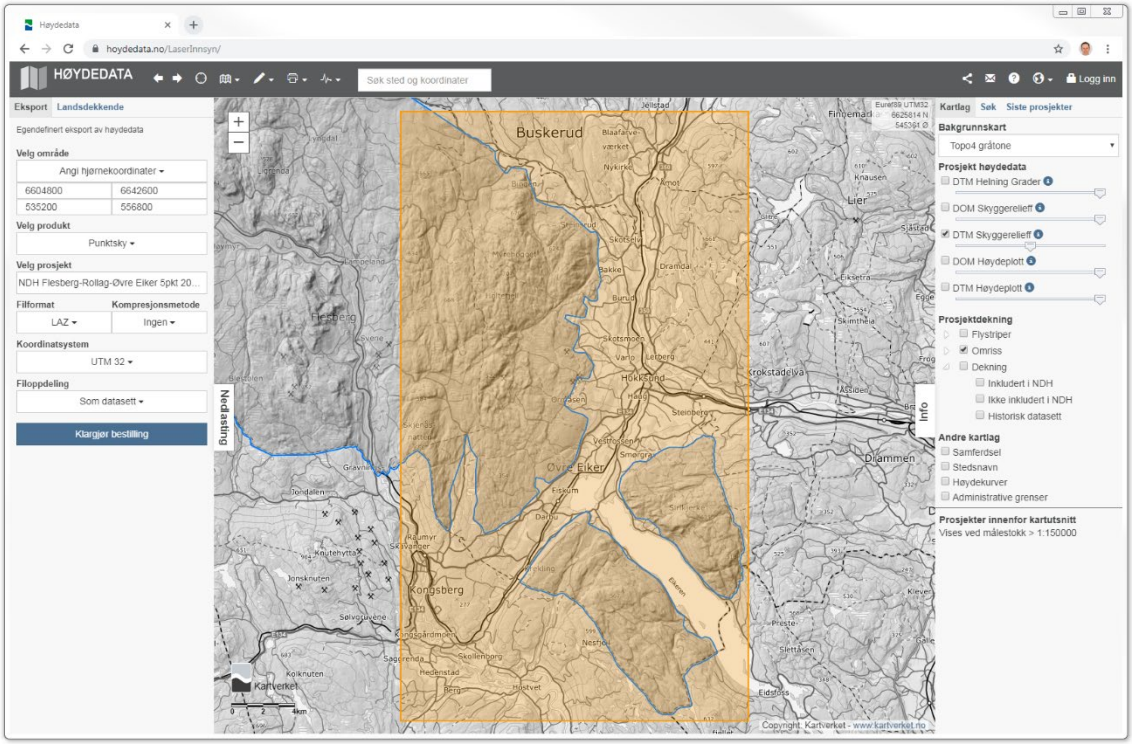



Figure 68. Øvre Eiker Flesberg 2017 dataset.

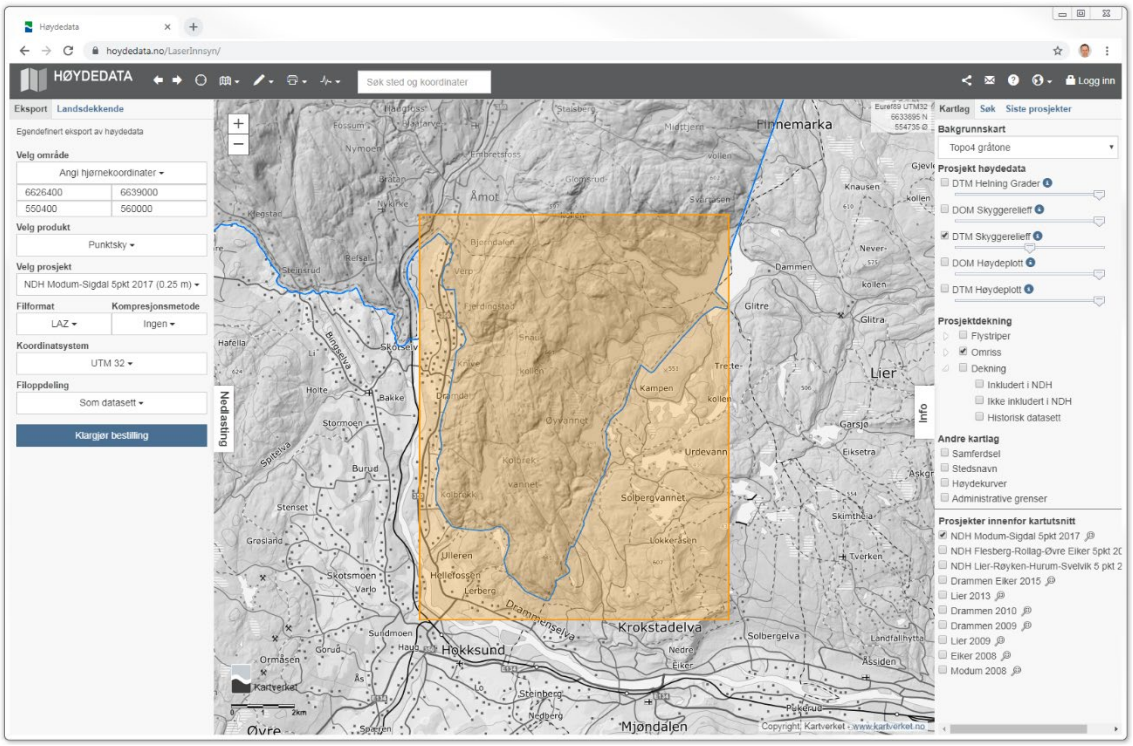



Figure 69. Øvre Eiker Modum 2017 dataset.

# 4 Methods

## 4.1 Preprocessing

The ALS point cloud data were converted to a digital terrain model (DTM) with 0.25 m pixel spacing. The DTM was converted to a simplified local relief model (LRM) by subtracting a smoothed version of the DTM. The LRM enhances local elevation differences while suppressing the general landscape topography (Hesse 2010). Thus, cultural heritage objects including grave mounds, pitfall traps and charcoal kilns may be visible.

For each cultural heritage object in the vector data, a 150 m × 150 m image was extracted from the LRM. The object's position within the subimage was selected at random. This was done in order to prevent the deep neural network from always predicting the object in the image centre. All cultural heritage objects within the subimage were included in the image annotation. Thus, each image contained one or more cultural heritage objects clearly visible.

## 4.2 Detection

For detection, the python code library *simple faster R-CNN* was downloaded from https://github.com/chenyuntc/simple-faster-rcnn-pytorch. For each detected object the R-CNN predicts a bounding box, a class label and a score value in the range 0.0 – 1.0. A few modifications had to be done:

1. The list of class labels was changed to match the class labels used in the image annotations.

2. Training was done on annotated LRM images containing cultural heritage objects.

3. The downloaded code crashed if there were no detected objects within an image. Thus, if-tests had to be added.

When these changes were made, the python code predicted the location and sizes of grave mounds (Figure 70), pitfall traps (Figure 71) and charcoal kilns (Figure 72) in LRM images of size 600 × 600 pixels.

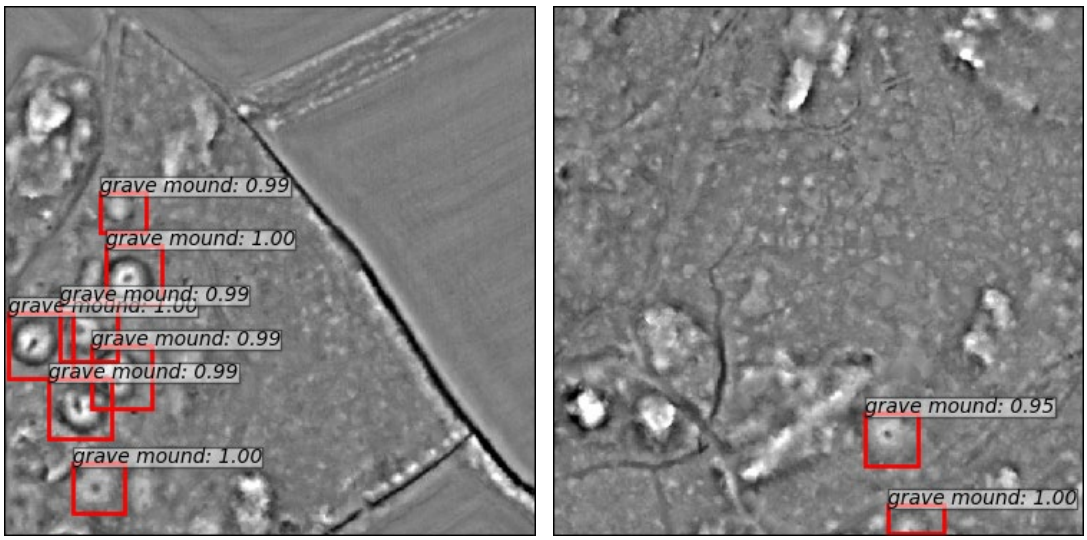



Figure 70. Predicted grave mound locations.

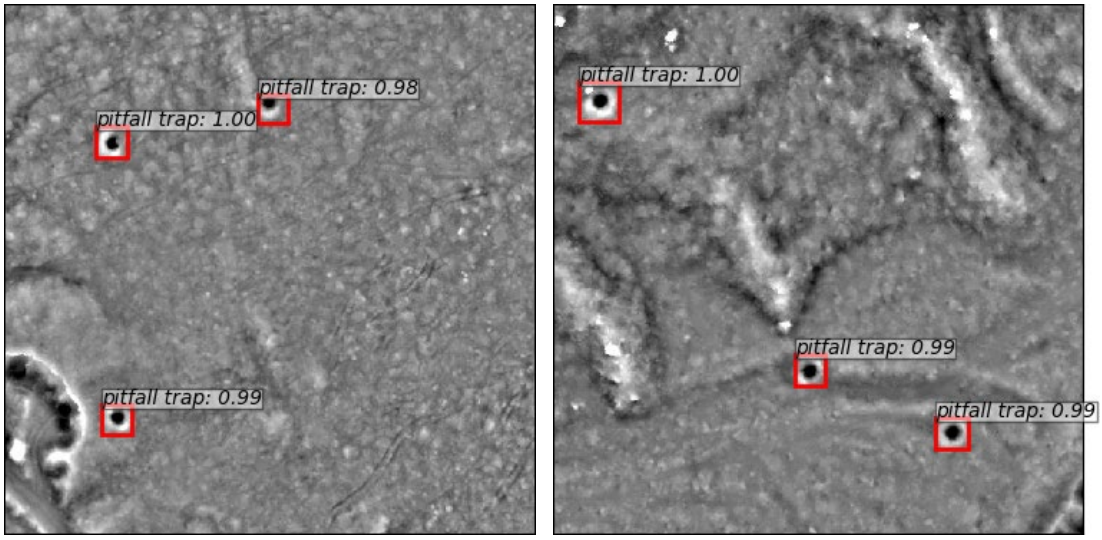



Figure 71. Predicted pitfall trap locations.

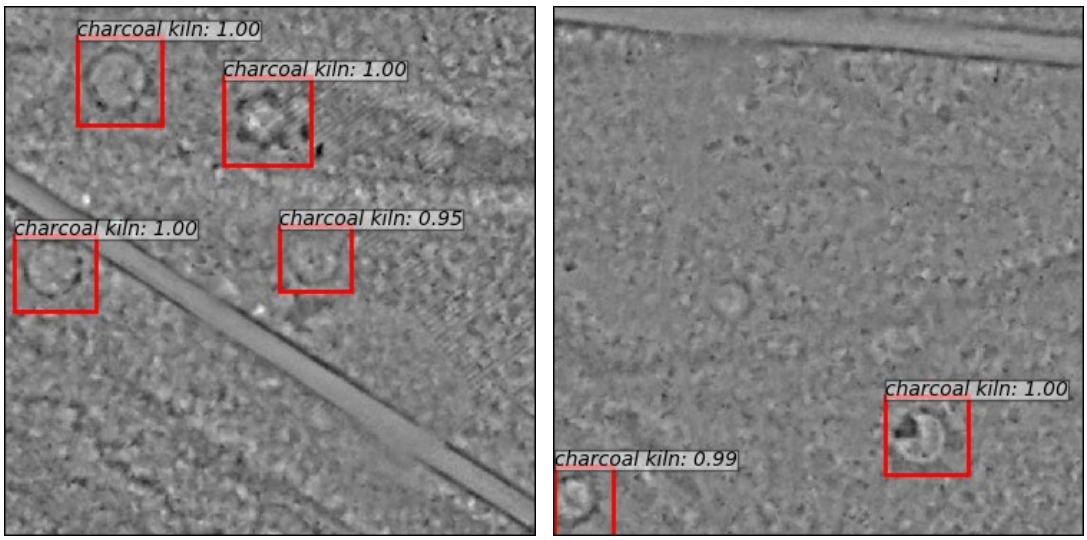



Figure 72. Predicted charcoal kiln locations.

### 4.3 Processing chain

The preprocessing and detection methods were integrated into a python script that may be called from QGIS or started from the Linux command line. The input was a collection of LAS files, and the output was two ESRI shape files for each object type; centre points in one file and object outlines in another file. Each object outline was obtained by converting the predicted bounding box to a circle.

## 5 Running the source code

### 5.1 jocuda

The machine jocuda has a GPU.

```
ssh jocuda
```

To mount a disk from the regular file system:

```
mkdir jodata2

sshfs -o uid=1000,gid=1000 trier@jo2.ad.nr.no:/nr/samba/jodata2 ~/jodata2

mkdir pro

sshfs -o uid=1000,gid=1000 trier@jo2.ad.nr.no:/nr/samba/jo/pro ~/pro

mkdir jodata9

sshfs -o uid=1000,gid=1000 trier@jo2.ad.nr.no:/nr/samba/jodata9 ~/jodata9
```

```
mkdir jodata10

sshfs -o uid=1000,gid=1000 trier@jo2.ad.nr.no:/nr/samba/jodata10 ~/jodata10
```

## 5.2 Python 3 virtual environment

First time:

```
virtualenv -p python3 .env
```

Alternatively:

```
virtualenv -p python2 .env

trier@jocuda1:~/cultsearcher/sfrcnn_p2$
virtualenv -p /usr/bin/python2.7 .env
```

Each time:

```
source .env/bin/activate
```

To end:

```
deactivate
```

## 5.3 Source code

Python source code:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier

/nr/samba/jo/pro/cultsearcher2018/usr/trier/src/cultsearcher/det
ectionline

/nr/samba/jo/pro/cultsearcher2018/usr/trier/src/cultsearcher/gui
```

### 5.3.1 jonrpy

To install jonrpy:

```
(.env)
trier@jo1:/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sf
rcnn$ pip install -e jonrpy
```

## 5.4 How to run

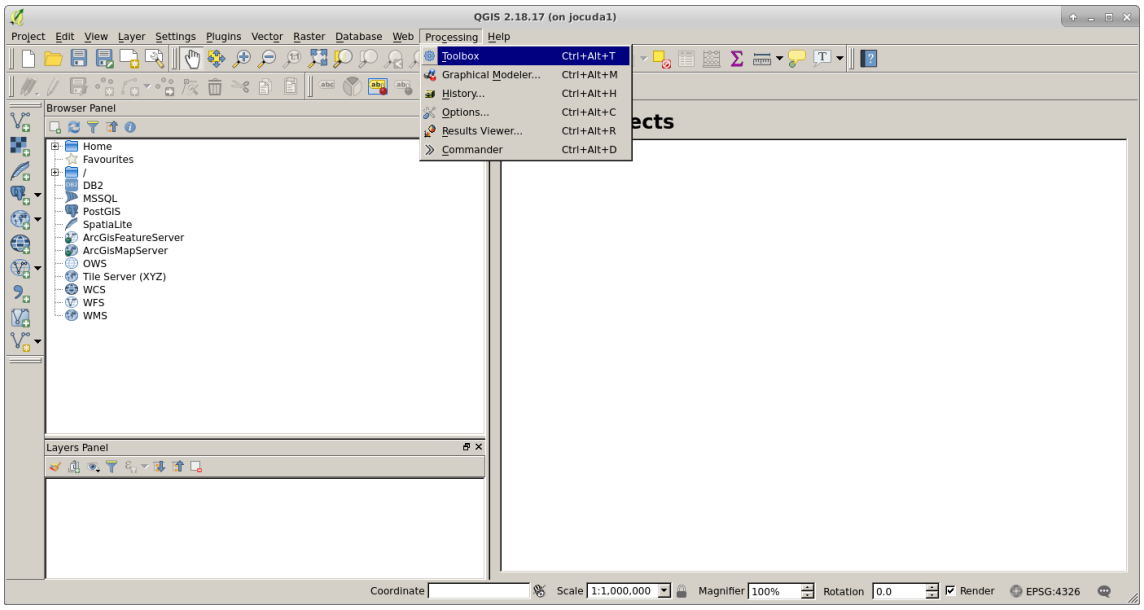
From command line (no line breaks):

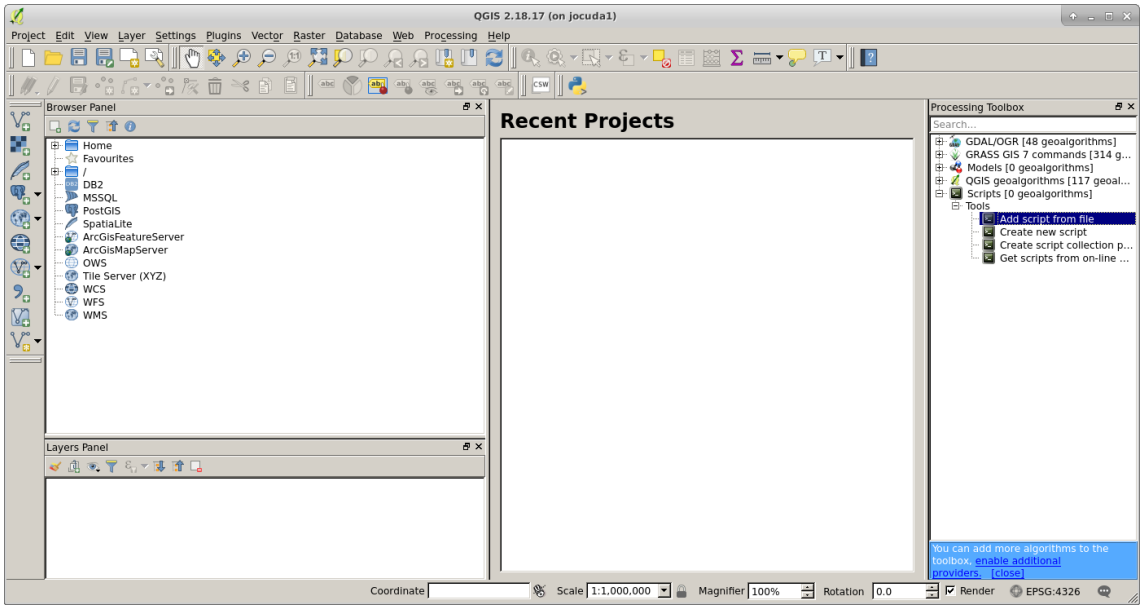
```
(.env) trier@jo1:/nr/samba/jo/pro/cultsearcher2018/usr/trier/
jo1_src/sfrcnn/simple-faster-rcnn-pytorch-master$
python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/lidar/oppland/dovre/LAS/folldal_2018_utm33
dovre_folldal_2018_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
larvik_2017_test_hele/LAS/larvik_2017 larvik_2017_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/ols
tappen_test nordfron_olstappen_2010_test



(.env) trier@jocuda1:~/cultsearcher/sfrcnn/simple-faster-rcnn-
pytorch-master$ python3 main.py --utm-zone 33
/home/trier/jodata10/lidar/oppland/lesja_2013_utm33/LAS/lesja_te
st lesja_2013_test

python3 main.py --heaps --utm-zone 33
/home/trier/jodata2/data/lidar/vestfold/larvik_2017/LAS/bokeskog
en bokeskogen

python3 main.py --utm-zone 33
/home/trier/jodata2/data/lidar/vestfold/larvik_2017/LAS/test_1
test_1

(.env) trier@cuda:/opt/nr/cultsearcher/simple-faster-rcnn-
pytorch-master$ python3 main.py --utm-zone 33
/opt/nr/cultsearcher/lidar/vestfold/larvik_2017/LAS/test_1
test_1
```

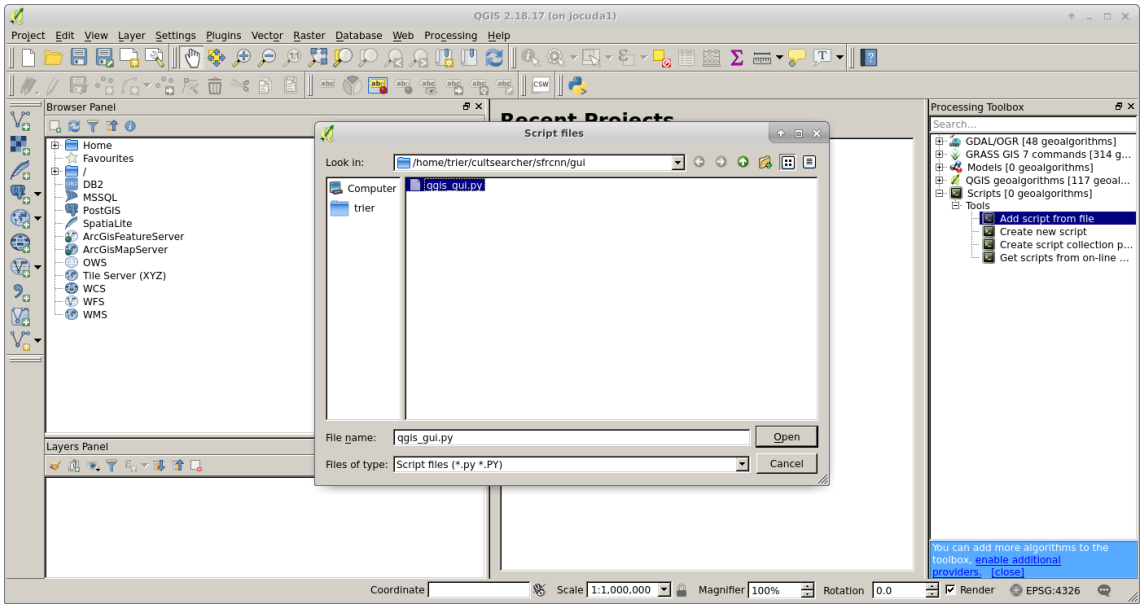
From QGIS:

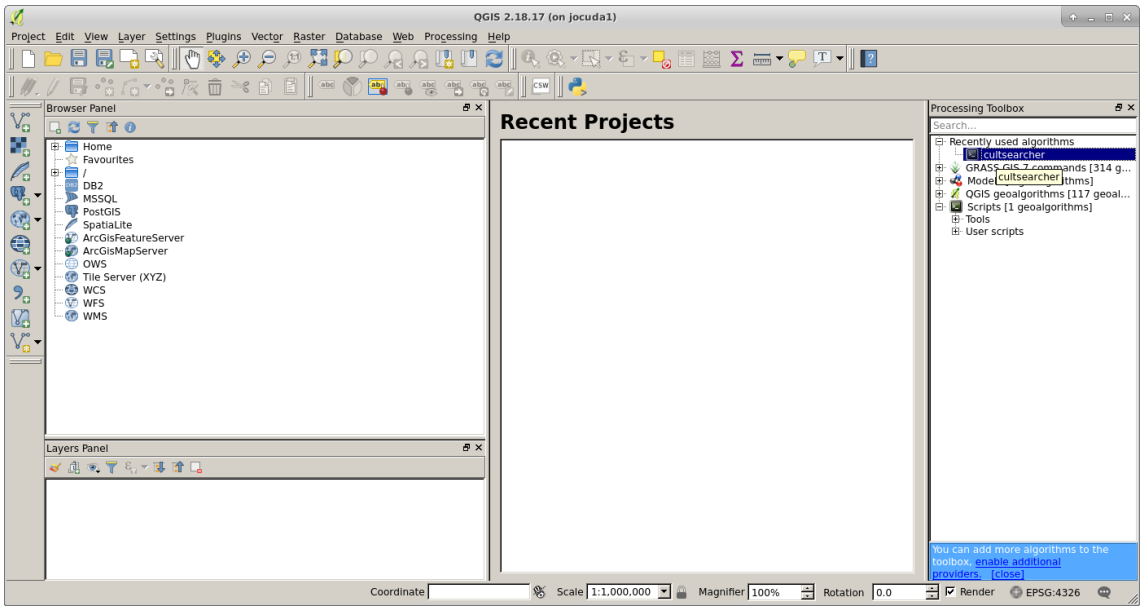
```
(.env) trier@jocuda1:~/cultsearcher/sfrcnn/gui$ qgis
```

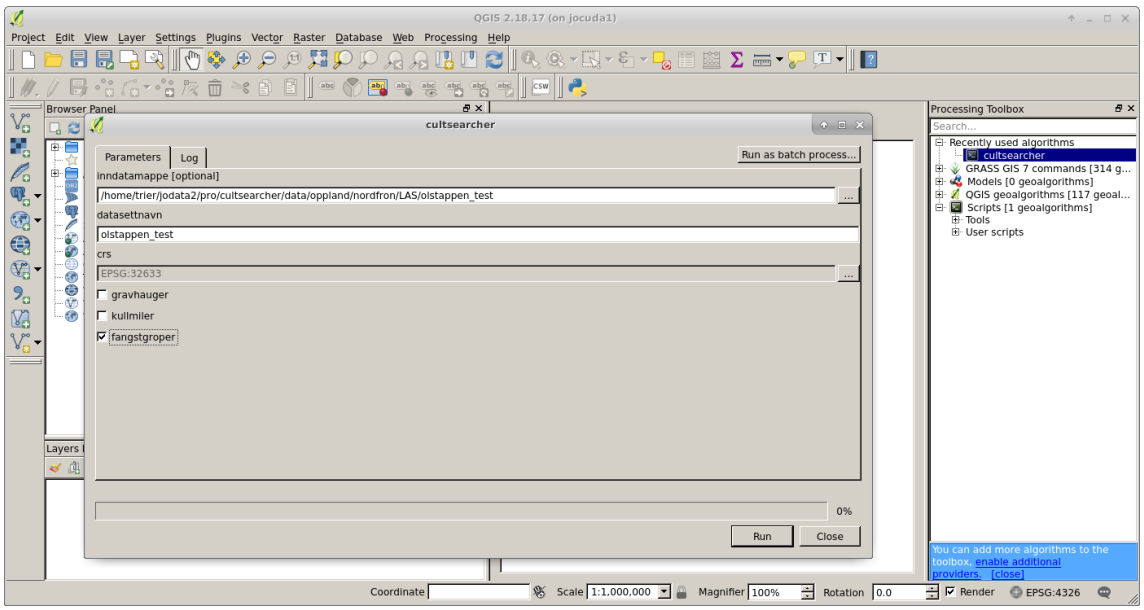

QGIS 2.18
Las Palmas de G.C.
Restoring loaded plugins

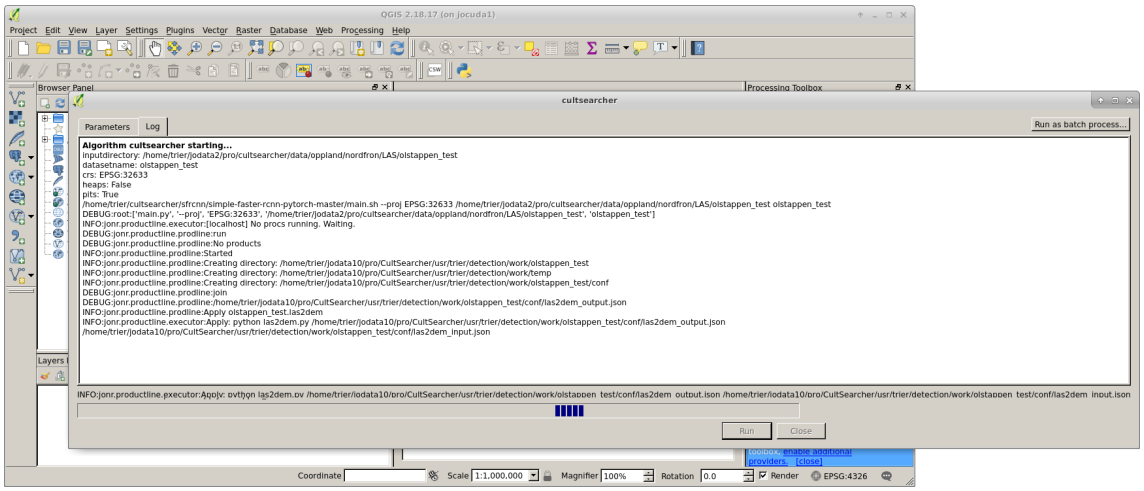

QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Toolbox Ctrl+Alt+T
Graphical Modeler... Ctrl+Alt+M
History... Ctrl+Alt+H
Options... Ctrl+Alt+C
Results Viewer... Ctrl+Alt+R
Commander Ctrl+Alt+D
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
Coordinate
Scale 1:1,000,000
Magnifier 100%
Rotation 0.0
Render
EPSG:4326

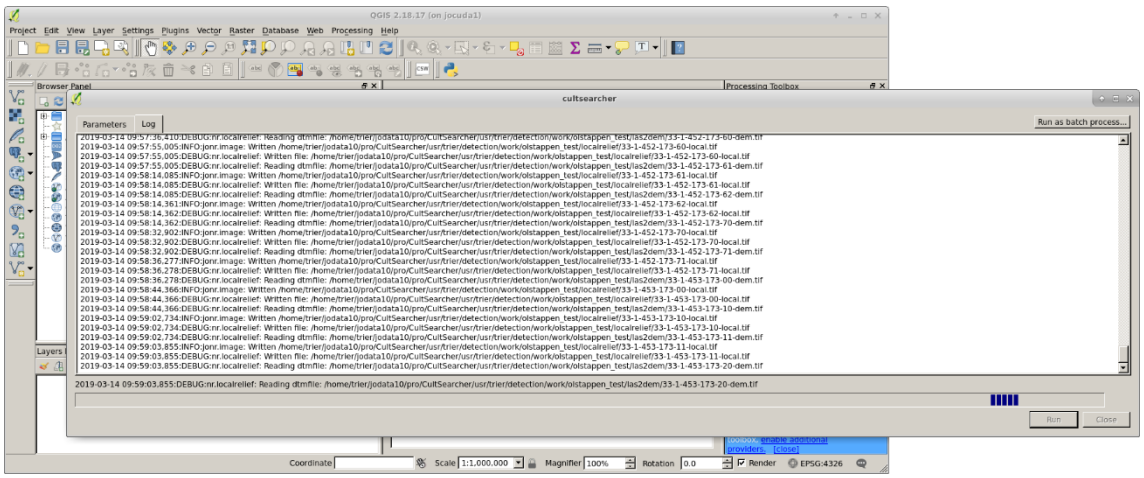

QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
Recent Projects
Processing Toolbox
Search...
GDAL/OGR [48 geoalgorithms]
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [0 geoalgorithms]
Tools
Add script from file
Create new script
Create script collection p...
Get scripts from on-line ...
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate
Scale 1:1,000,000
Magnifier 100%
Rotation 0.0
Render
EPSG:4326

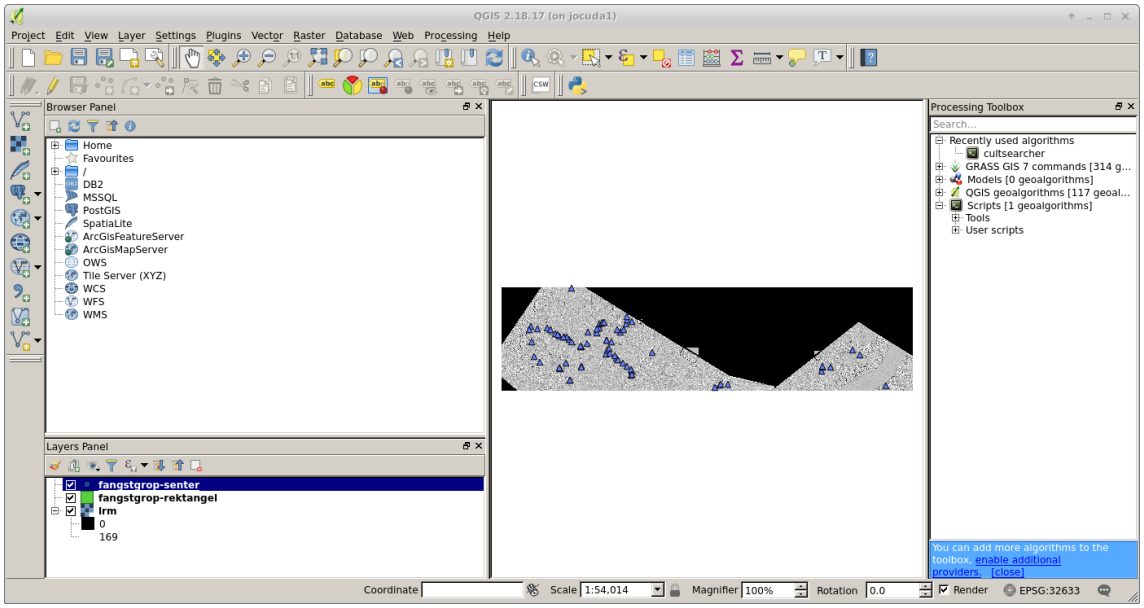

QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
Script files
Look in: /home/trier/cultsearcher/sfrcnn/gui
Computer
trier
qgis_gui.py
File name: qgis_gui.py
Open
Files of type: Script files (*.py *.PY)
Cancel
Processing Toolbox
Search...
GDAL/OGR [48 geoalgorithms]
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [0 geoalgorithms]
Tools
Add script from file
Create new script
Create script collection p...
Get scripts from on-line ...
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate
Scale 1:1,000,000
Magnifier 100%
Rotation 0.0
Render
EPSG:4326

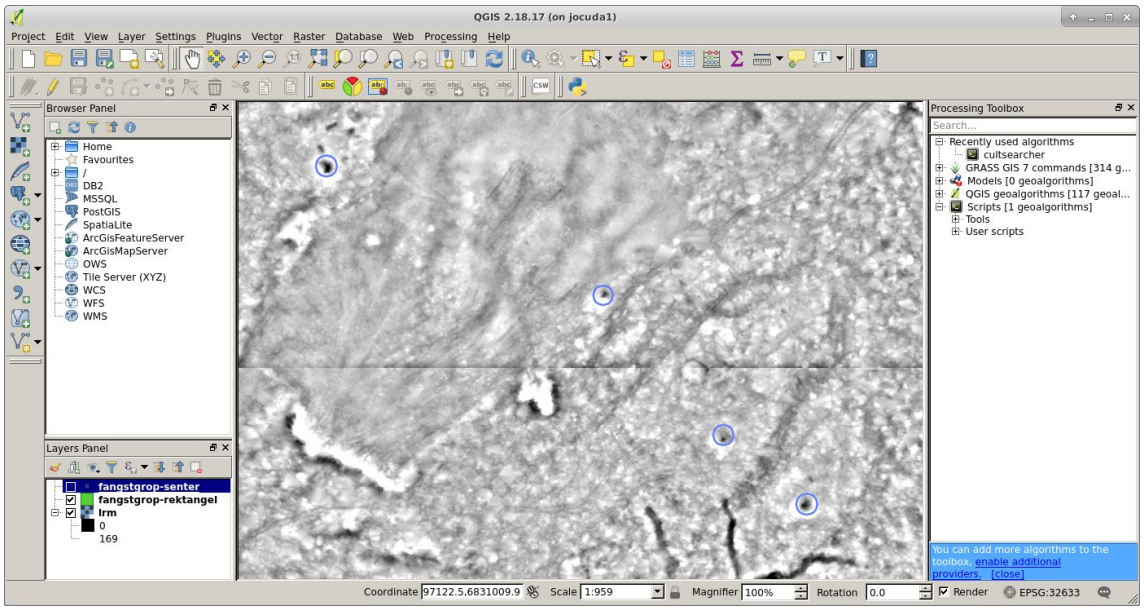

QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
Recent Projects
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
cultsearcher
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate
Scale 1:1,000,000
Magnifier 100%
Rotation 0.0
Render
EPSG:4326

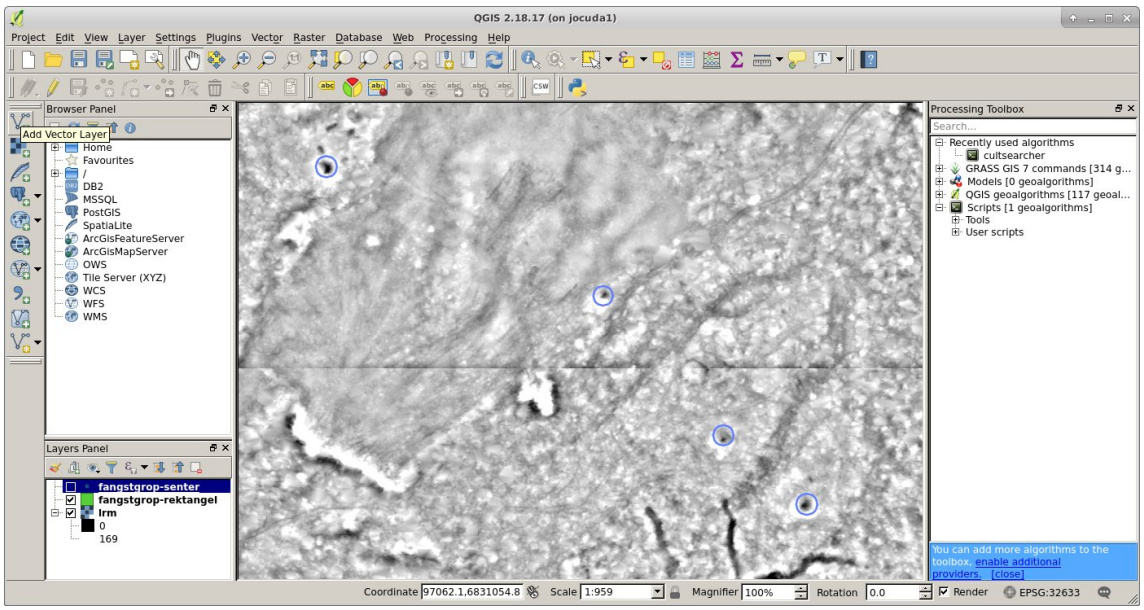

QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
cultsearcher
Parameters
Log
Run as batch process...
inndatamappe [optional]
/home/trier/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/olstappen_test
datasettnavn
olstappen_test
crs
EPSG:32633
gravhauger
kullmiler
fangstgroper
0%
Run
Close
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate
Scale 1:1,000,000
Magnifier 100%
Rotation 0.0
Render
EPSG:4326

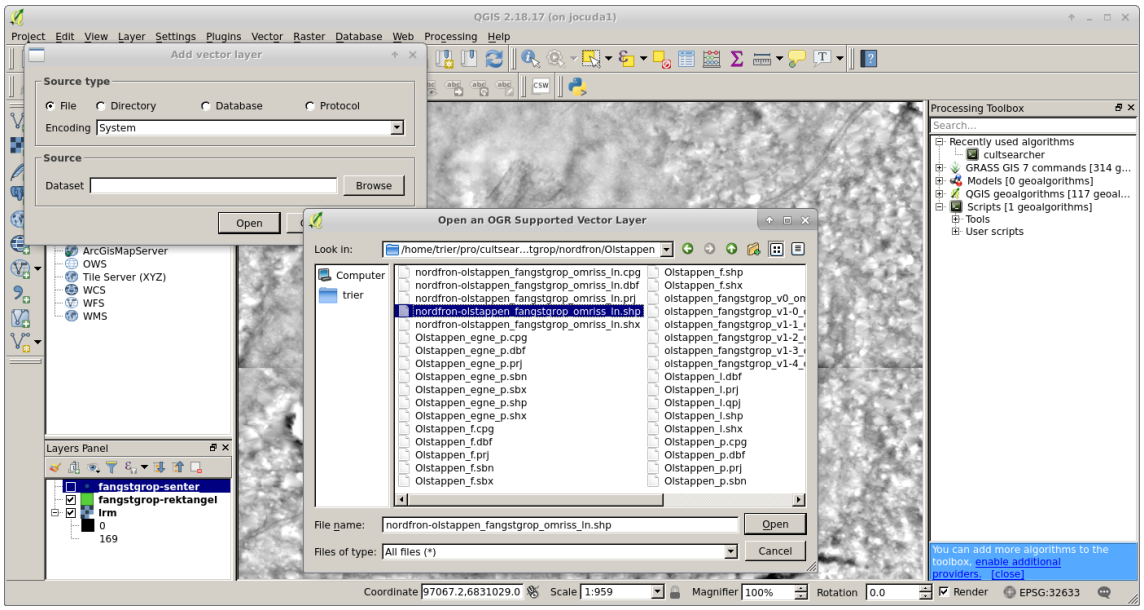

cultsearcher
Parameters
Log
Run as batch process...
Algorithm cultsearcher starting...
inputdirectory: /home/trier/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/olstappen_test
datasetname: olstappen_test
crs: EPSG:32633
heaps: False
pits: True
/home/trier/cultsearcher/sfrcnn/simple-faster-rcnn-pytorch-master/main.sh --proj EPSG:32633 /home/trier/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/olstappen_test olstappen_test
DEBUG:root:['main.py', '--proj', 'EPSG:32633', '/home/trier/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/olstappen_test', 'olstappen_test']
INFO:jonr.productline.executor:[localhost] No procs running. Waiting.
DEBUG:jonr.productline.prodline:run
DEBUG:jonr.productline.prodline:No products
INFO:jonr.productline.prodline:Started
INFO:jonr.productline.prodline:Creating directory: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test
INFO:jonr.productline.prodline:Creating directory: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/temp
INFO:jonr.productline.prodline:Creating directory: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/conf
DEBUG:jonr.productline.prodline:join
DEBUG:jonr.productline.prodline:/home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/conf/las2dem_output.json
INFO:jonr.productline.prodline:Apply olstappen_test.las2dem
INFO:jonr.productline.executor:Apply: python las2dem.py /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/conf/las2dem_output.json
/home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/conf/las2dem_input.json
Run
Close

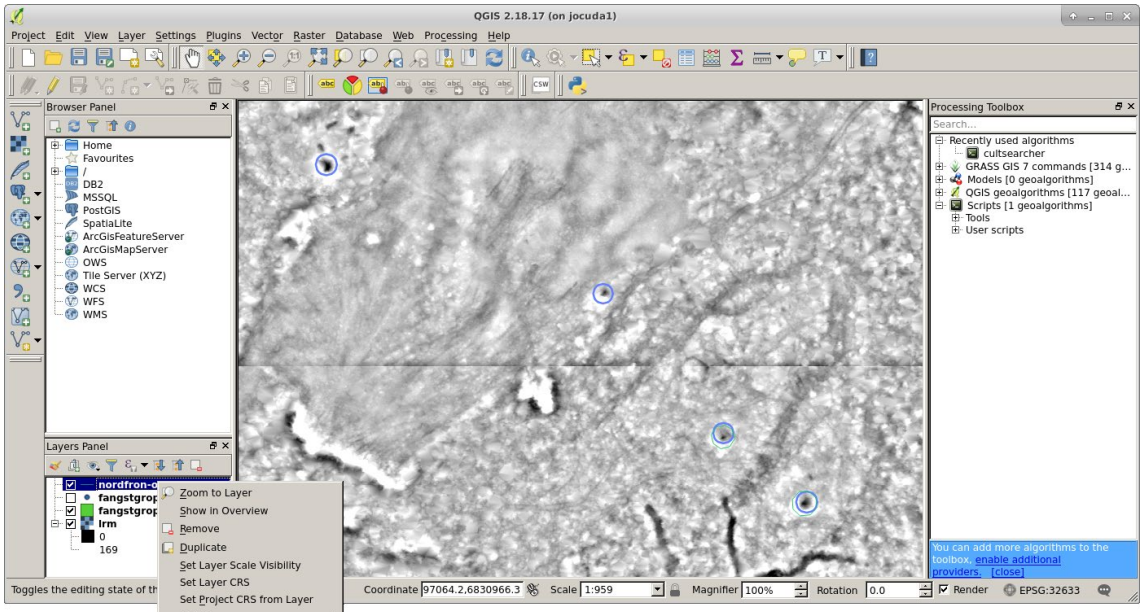

cultsearcher
Parameters
Log
Run as batch process...
2019-03-14 09:58:36,277:INFO:jonr.image: Written /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/localrelief/33-1-452-173-71-local.tif
2019-03-14 09:58:36,278:DEBUG:nr.localrelief: Written file: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/localrelief/33-1-452-173-71-local.tif
2019-03-14 09:58:36,278:DEBUG:nr.localrelief: Reading dtmfile: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/las2dem/33-1-453-173-00-dem.tif
2019-03-14 09:58:44,366:INFO:jonr.image: Written /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/localrelief/33-1-453-173-00-local.tif
2019-03-14 09:58:44,366:DEBUG:nr.localrelief: Written file: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/localrelief/33-1-453-173-00-local.tif
2019-03-14 09:58:44,366:DEBUG:nr.localrelief: Reading dtmfile: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/las2dem/33-1-453-173-10-dem.tif
2019-03-14 09:59:02,734:INFO:jonr.image: Written /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/localrelief/33-1-453-173-10-local.tif
2019-03-14 09:59:02,734:DEBUG:nr.localrelief: Written file: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/localrelief/33-1-453-173-10-local.tif
2019-03-14 09:59:02,734:DEBUG:nr.localrelief: Reading dtmfile: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/las2dem/33-1-453-173-11-dem.tif
2019-03-14 09:59:03,855:INFO:jonr.image: Written /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/localrelief/33-1-453-173-11-local.tif
2019-03-14 09:59:03,855:DEBUG:nr.localrelief: Written file: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/localrelief/33-1-453-173-11-local.tif
2019-03-14 09:59:03,855:DEBUG:nr.localrelief: Reading dtmfile: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/las2dem/33-1-453-173-20-dem.tif
2019-03-14 09:59:03,855:DEBUG:nr.localrelief: Reading dtmfile: /home/trier/jodata10/pro/CultSearcher/usr/trier/detection/work/olstappen_test/las2dem/33-1-453-173-20-dem.tif
Run
Close

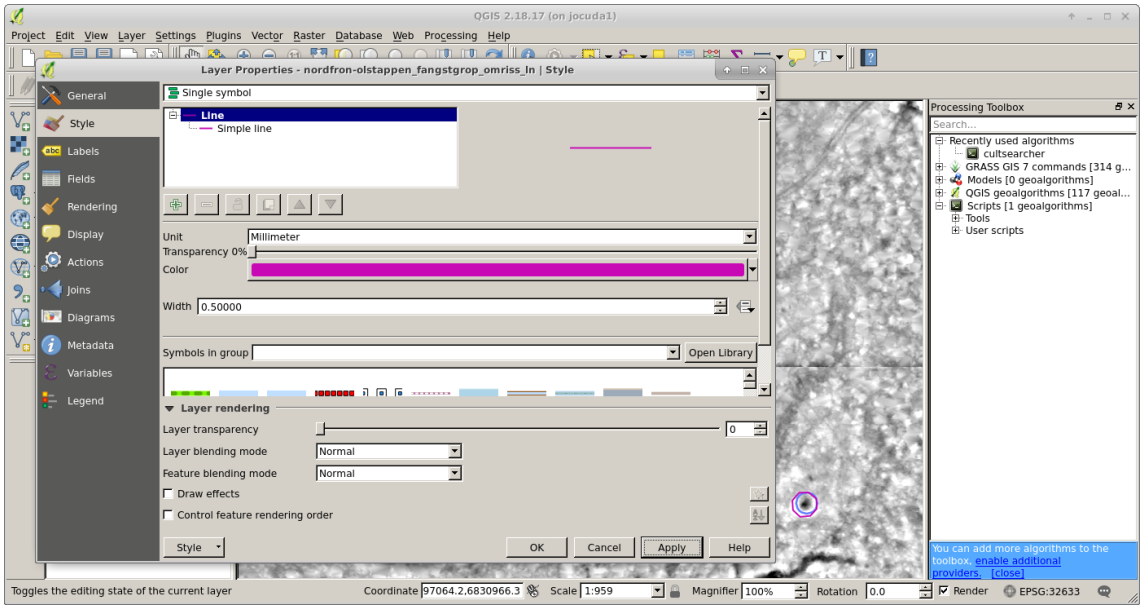

QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
fangstgrop-senter
fangstgrop-rektangel
lrm
0
169
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate
Scale 1:54,014
Magnifier 100%
Rotation 0.0
Render
EPSG:32633

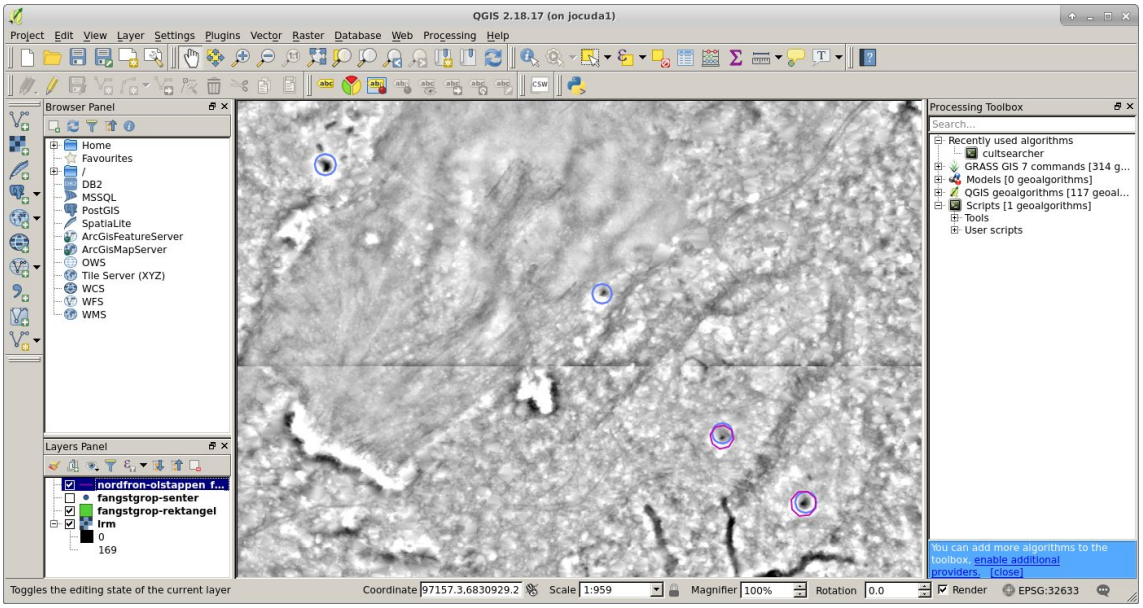

QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
fangstgrop-senter
fangstgrop-rektangel
lrm
0
169
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate 97122.5,6831009.9
Scale 1:959
Magnifier 100%
Rotation 0.0
Render
EPSG:32633


QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Add Vector Layer
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
fangstgrop-senter
fangstgrop-rektangel
lrm
0
169
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate 97062.1,6831054.8
Scale 1:959
Magnifier 100%
Rotation 0.0
Render
EPSG:32633


QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Add vector layer
Source type
File
Directory
Database
Protocol
Encoding System
Source
Dataset
Browse
Open
Open an OGR Supported Vector Layer
Look in: /home/trier/pro/cultsear...tgrop/nordfron/Olstappen
Computer
trier
nordfron-olstappen_fangstgrop_omriss_ln.cpg
nordfron-olstappen_fangstgrop_omriss_ln.dbf
nordfron-olstappen_fangstgrop_omriss_ln.prj
nordfron-olstappen_fangstgrop_omriss_ln.shp
nordfron-olstappen_fangstgrop_omriss_ln.shx
Olstappen_egne_p.cpg
Olstappen_egne_p.dbf
Olstappen_egne_p.prj
Olstappen_egne_p.sbn
Olstappen_egne_p.sbx
Olstappen_egne_p.shp
Olstappen_egne_p.shx
Olstappen_f.cpg
Olstappen_f.dbf
Olstappen_f.prj
Olstappen_f.sbn
Olstappen_f.sbx
Olstappen_f.shp
Olstappen_f.shx
olstappen_fangstgrop_v0_on
olstappen_fangstgrop_v1-0_
olstappen_fangstgrop_v1-1_
olstappen_fangstgrop_v1-2_
olstappen_fangstgrop_v1-3_
olstappen_fangstgrop_v1-4_
Olstappen_l.dbf
Olstappen_l.prj
Olstappen_l.qpj
Olstappen_l.shp
Olstappen_l.shx
Olstappen_p.cpg
Olstappen_p.dbf
Olstappen_p.prj
Olstappen_p.sbn
File name: nordfron-olstappen_fangstgrop_omriss_ln.shp
Open
Files of type: All files (*)
Cancel
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate 97067.2,6831029.0
Scale 1:959
Magnifier 100%
Rotation 0.0
Render
EPSG:32633


QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
Zoom to Layer
Show in Overview
Remove
Duplicate
Set Layer Scale Visibility
Set Layer CRS
Set Project CRS from Layer
Styles
Open Attribute Table
Toggle Editing
Save As...
Save As Layer Definition File...
Filter...
Show Feature Count
Properties
Rename
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
You can add more algorithms to the toolbox, enable additional providers. [close]
Coordinate 97064.2,6830966.3
Scale 1:959
Magnifier 100%
Rotation 0.0
Render
EPSG:32633


QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Layer Properties - nordfron-olstappen_fangstgrop_omriss_ln | Style
General
Style
Labels
Fields
Rendering
Display
Actions
Joins
Diagrams
Metadata
Variables
Legend
Single symbol
Line
Simple line
Unit
Millimeter
Transparency 0%
Color
Width
0.50000
Symbols in group
Open Library
Layer rendering
Layer transparency
Layer blending mode
Normal
Feature blending mode
Normal
Draw effects
Control feature rendering order
Style
OK
Cancel
Apply
Help
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
You can add more algorithms to the toolbox, enable additional providers. [close]
Toggles the editing state of the current layer
Coordinate 97064.2,6830966.3
Scale 1:959
Magnifier 100%
Rotation 0.0
Render
EPSG:32633


QGIS 2.18.17 (on jocuda1)
Project Edit View Layer Settings Plugins Vector Raster Database Web Processing Help
Browser Panel
Home
Favourites
/
DB2
MSSQL
PostGIS
SpatiaLite
ArcGisFeatureServer
ArcGisMapServer
OWS
Tile Server (XYZ)
WCS
WFS
WMS
Layers Panel
nordfron-olstappen_f...
fangstgrop-senter
fangstgrop-rektangel
lrm
0
169
Processing Toolbox
Search...
Recently used algorithms
cultsearcher
GRASS GIS 7 commands [314 g...
Models [0 geoalgorithms]
QGIS geoalgorithms [117 geoal...
Scripts [1 geoalgorithms]
Tools
User scripts
You can add more algorithms to the toolbox, enable additional providers. [close]
Toggles the editing state of the current layer
Coordinate 97157.3,6830929.2
Scale 1:959
Magnifier 100%
Rotation 0.0
Render
EPSG:32633

## 5.5 Simple faster R-CNN

This seemed to be the best alternative for testing if R-CNN works for cultural heritage detection.

```
https://github.com/chenyuntc/simple-faster-rcnn-pytorch
```

CuPy must also be installed. The version of cupy-cuda must match the installed cuda version.

```
trier@jocuda1:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2018 NVIDIA Corporation
Built on Sat_Aug_25_21:08:01_CDT_2018
Cuda compilation tools, release 10.0, V10.0.130
```

Thus, we should install cupy-cuda100:

```
(.env) trier@jocuda1:~/cultsearcher/sfrcnn$ pip install cupy-
cuda100
```

Alternatively, use:

```
trier@jocuda1:~$ cd /usr/local
trier@jocuda1:/usr/local$ ls -l
total 40
drwxr-xr-x  2 root root 4096 Jul 25  2018 bin
lrwxrwxrwx  1 root root    9 Jul  2 12:35 cuda -> cuda-10.1
drwxr-xr-x  3 root root 4096 Jul  2 12:31 cuda-10.0
drwxr-xr-x 17 root root 4096 Jul  2 12:34 cuda-10.1
drwxr-xr-x  2 root root 4096 Jul 25  2018 etc
drwxr-xr-x  2 root root 4096 Jul 25  2018 games
drwxr-xr-x  2 root root 4096 Jul 25  2018 include
drwxr-xr-x  4 root root 4096 Oct  9  2018 lib
lrwxrwxrwx  1 root root    9 Jul 25  2018 man -> share/man
drwxr-xr-x  2 root root 4096 Jul 25  2018 sbin
drwxr-xr-x  6 root root 4096 Oct  1  2018 share
drwxr-xr-x  2 root root 4096 Jul 25  2018 src
trier@jocuda1:/usr/local$
```

In this case, cuda version 10.1 is used.

```
sudo apt-get install python3-tk
```

### 5.5.1 How to install cuda

On one occasion, we got the following error message:

```
cupy.cuda.runtime.CUDARuntimeError: cudaErrorNoDevice: no CUDA-
capable device is detected
```

Apparently, there is something missing. By executing this command:

```
nvidia-smi
```

we got:

```
NVIDIA-SMI has failed because it couldn't communicate with the
NVIDIA driver. Make sure that the latest NVIDIA driver is
installed and running.
```

This is not a complete description.

Erik Vassaasen had to help me with installing the necessary Nvidia packages.

```
sudo apt-get install cuda
```

```
nvidia-smi
```

### 5.5.2 Detection parameters

In the demo code it is mentioned that the dog can be detected if the threshold is set to 0.6 instead of 0.7, but the comment fails to say where to make the change. This must be done in the source code file:

~/cultsearcher/sfrcnn/simple-faster-rcnn-pytorch-master/model/faster_rcnn.py

Go to the function

```
    def use_preset(self, preset):
```

line 156:

```
            self.score_thresh = 0.7
```

Actually, 0.6 is still too high, so use 0.58:

```
            self.score_thresh = 0.58
```

Remember to change back to 0.7.

### 5.5.3 Try the demo code

In

```
~/cultsearcher/sfrcnn/simple-faster-rcnn-pytorch-
master/utils/vis_tool.py
```

Change on line 8:

```
# ODT: Commented out since I don't use notebook
#matplotlib.use('Agg')
```

### 5.5.4 Try the training code

```
(.env) trier@jocuda1:~/cultsearcher/sfrcnn/simple-faster-rcnn-
pytorch-master$ python train.py train --env='fasterrcnn-caffe' -
-plot-every=100 --caffe-pretrain
```

This crashed after 38 minutes, 5011+3699 iterations.

```
5011it [31:35,  2.63it/s]
3699it [07:55,  6.83it/s]Traceback (most recent call last):
```

To avoid crash, I edited the file:

```
~/cultsearcher/sfrcnn/simple-faster-rcnn-pytorch-master/utils/config.py
```

Changed num_workers from 8 to 0 in lines 14-19:

```
    ## ODT: Changed num_workers from 8 to 0
    #num_workers = 8
    num_workers = 0
    ## ODT: Changed num_workers from 8 to 0
    #test_num_workers = 8
    test_num_workers = 0
```

### 5.5.5 Visdom

I could not get his to work.

Open a browser and goto:

http://jocuda1:8097/

### 5.5.6 Pretrained models

trier@jocuda1:~/jodata2/pro/cultsearcher/pretrained/simple_faster_r-cnn$ pwd

/home/trier/jodata2/pro/cultsearcher/pretrained/simple_faster_r-cnn

trier@jocuda1:~/jodata2/pro/cultsearcher/pretrained/simple_faster_r-cnn$ ls -l

total 1606420

-rwxr--r-- 1 andersuw andersuw 548317845 Jan 30 13:25 chainer_best_model_converted_to_pytorch_0.7053.pth

-rwxr--r-- 1 andersuw andersuw 548328495 Jan 30 13:22 fasterrcnn_12211511_0.701052458187_torchvision_pretrain.pth.701052458187

-rwxr--r-- 1 andersuw andersuw 548321187 Jan 30 13:24 fasterrcnn_12222105_0.712649824453_caffe_pretrain.pth.712649824453

I first tried python 3 but it didn't work; I got a lot of warnings from `urllib3/connections.py` or something like that. Then I tried python 2. Now the warnings didn't appear.

```
(.env) trier@jocuda1:~/cultsearcher/sfrcnn_p2/simple-faster-
rcnn-pytorch-master$ python train.py train --env='fasterrcnn-
caffe' --plot-every=100 --caffe-pretrain
```

```
======user config========
{'caffe_pretrain': True,
 'caffe_pretrain_path': 'checkpoints/vgg16_caffe.pth',
 'data': 'voc',
 'debug_file': '/tmp/debugf',
```

```
 'env': 'fasterrcnn-caffe',
 'epoch': 14,
 'load_path': None,
 'lr': 0.001,
 'lr_decay': 0.1,
 'max_size': 1000,
 'min_size': 600,
 'num_workers': 8,
 'plot_every': 100,
 'port': 8097,
 'pretrained_model': 'vgg16',
 'roi_sigma': 1.0,
 'rpn_sigma': 3.0,
 'test_num': 10000,
 'test_num_workers': 8,
 'use_adam': False,
 'use_chainer': False,
 'use_drop': False,
 'voc_data_dir':
'/home/trier/jodata2/pro/cultsearcher/imgdataset/VOCdevkit/VOC20
07',
 'weight_decay': 0.0005}
==========end===========
load data
model construct completed
WARNING:root:Setting up a new session...
WARNING:visdom:Without the incoming socket you cannot receive
events from the server or register event handlers to your Visdom
client.
```

### 5.5.7 Generate training images

The method must make sure that the true objects are in different locations within the extracted image portions. Conversely, if the true objects are always in the image centre of the training images, then the trained network will not be able to detect true objects if they are not in the image centre.

Source code for generating training images:

```
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/extra
ct_frcnn_training_imgs_v3.pro
```

Execute the following commands on the ENVI prompt (no line breaks within each single command):

```
.COMPILE
/nr/samba/jo/pro/cultsearcher2018/usr/trier/eoTools/src/methods/
tools/file_tools.pro
```

```
.COMPILE
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/cult_
merge_files.pro

.COMPILE
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/extra
ct_frcnn_training_imgs.pro

extract_frcnn_training_images_v3
```

Alternative versions of the source code, for use with the alternative subdivision into training, validation and test subsets, are in the below two source code files. The second makes eight versions of each extracted image, by rotation and flipping.

```
extract_frcnn_training_imgs_v1.pro

extract_frcnn_training_imgs_v1_augment-8.pro
```

### 5.5.8 Training of neural network

Edit the file:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simpl
e-faster-rcnn-pytorch-master/utils/config.py
```

In line 13, you may need to change:

```
    ra_data_dir =
'/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_test_v3_0013/'
```

Edit the file:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simpl
e-faster-rcnn-pytorch-master/data/ra_dataset.py
```

In lines 144-147, you may need to update the list of class labels. E.g., if you have added a confusion class, this class name must be added.

```
RA_BBOX_LABEL_NAMES = (
    'gravhaug',
    'fangstgrop',
    'kullmile',
    'kollenaturlig')
```

Execute:

```
(.env) trier@jocuda1:~/cultsearcher/sfrcnn/simple-faster-rcnn-
pytorch-master$ python -W ignore train.py train --
env='fasterrcnn-caffe' --plot-every=100 --caffe-pretrain
```

### 5.5.9 Code changes to allow zero objects in an image

The downloaded code crashes if there are no detected objects within an image. In many object detection scenarios, this is a common situation.

The following code changes were done.

In the source code file `model/faster_rcnn.py`, at line 130:

```
class FasterRCNN(nn.Module):
…
    def forward(self, x, scale=1.):
…
        h = self.extractor(x)
        rpn_locs, rpn_scores, rois, roi_indices, anchor = \
            self.rpn(h, img_size, scale)

        ## ODT Added if-test
        if len(rois)>0:
            roi_cls_locs, roi_scores = self.head(
                h, rois, roi_indices)
        else:
            roi_cls_locs = []
            roi_scores = []
        ## ODT End added if-test
        ## ODT Old code:
        #roi_cls_locs, roi_scores = self.head(
        #    h, rois, roi_indices)
        ## ODT End old code

        return roi_cls_locs, roi_scores, rois, roi_indices
```

In the same source code file `model/faster_rcnn.py`, at line :

```
   def predict(self, imgs,sizes=None,visualize=False):
…
            roi_cls_loc, roi_scores, rois, _ = self(img, scale=scale)
            #import pdb; pdb.set_trace()
            ## ODT Added if-test on len(rois)
            if len(rois)>0:
                ## ODT The following old code is only executed if len(rois)>0
                # We are assuming that batch size is 1.
…
                ## ODT End of old code that is now executed only if
len(rois)>0

        self.use_preset('evaluate')
        self.train()
```

```
        #import pdb; pdb.set_trace()

        return bboxes, labels, scores
```

In the source code file `model/region_proposal_network.py`, line 130:

```
class RegionProposalNetwork(nn.Module):
…
    def forward(self, x, img_size, scale=1.):
…
        for i in range(n):
            roi = self.proposal_layer(
                rpn_locs[i].cpu().data.numpy(),
                rpn_fg_scores[i].cpu().data.numpy(),
                anchor, img_size,
                scale=scale)
            ## ODT Added if-test on len(roi)
            if len(roi)>0:
                batch_index = i * np.ones((len(roi),), dtype=np.int32)
                rois.append(roi)
                roi_indices.append(batch_index)
            ## ODT End of added if-test.
            ## ODT Old code:
            #batch_index = i * np.ones((len(roi),), dtype=np.int32)
            #rois.append(roi)
            #roi_indices.append(batch_index)
            ## ODT End of old code

        ## ODT Added if-tests on len(rois) and len(roi_indices)
        if len(rois)>0:
            rois = np.concatenate(rois, axis=0)
        if len(roi_indices)>0:
            roi_indices = np.concatenate(roi_indices, axis=0)
        ## ODT End of if-tests
        ## ODT old code:
        #rois = np.concatenate(rois, axis=0)
        #roi_indices = np.concatenate(roi_indices, axis=0)
        ## ODT End of old code
        return rpn_locs, rpn_scores, rois, roi_indices, anchor
```

### 5.5.10 Running detection on extracted test images

Run the code:

```
python demo_2_test.py
```

This code may be run in two modes. To demonstrate the detection results on one image at a time, use (line 51 in `demo_2_test.py`):

```
visualize_each_image = True
```

For each test image, the detection results are displayed first. Close the image window. Then the annotations for that image are displayed. Close the image window. Then the next test image is displayed, and so on.

The displayed labels are in English or Norwegian. The list of possible labels to display is specified on lines 46-51 in utils/vis_tool.py:

```
RA_BBOX_LABEL_NAMES = (
    'gravhaug',
    'fangstgrop',
    'kullmile',
    'kollenaturlig',
    'gropnaturlig')
```

The display of each image with labelled detections and annotations is nice to get an indication of whether the code works or not.

To, instead, run on all the test images without displaying them, but to get an estimate of detection performance, use (line 51 in `demo_2_test.py`):

```
visualize_each_image = False
```

Make sure that the input and output directories and file names are correct, as follows.

The parameters of the trained network were saved to a file each time the detection performance was improved during training. Make sure to use the correct file on line 27 in `demo_2_test.py`:

```
trainer.load('/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_sr
c/sfrcnn/simple-faster-rcnn-pytorch-
master/checkpoints/fasterrcnn_04041754_0.7982244621705039')
```

The location of the image database is specified on lines 39-45:

```
img_dir =
'/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_
test_v3_0006/JPEGImages/'



list_dir =
'/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_
test_v3_0006/ImageSets/Main/'

test_list_file = os.path.join(list_dir, 'test.txt')
```

```
annotation_dir =
'/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_
test_v3_0006/Annotations/'
```

The output location is specified on lines 47-48 in `demo_2_test.py`:

```
test_out_dir = '/nr/samba/jodata2/pro/cultsearcher/usr/trier/'

test_out_file = os.path.join(test_out_dir,
'RA_test_v3_0006.csv')
```

The class names must match those used in the training

### 5.5.11 Running detection on large areas

Section 5.4 describes how to start the processing on a collection of lidar LAS files. The processing consists of the following steps:

1. Conversion of LAS files to digital terrain model (DTM) raster files in the TIFF format.
2. Conversion of DTM raster files to local relief model (LRM) raster files.
3. Automatic detection of structures in the LRM files
4. Export of detection results and LRM raster files

Step 3 is the function `cultdetection()`, located in `cultdetection.py`. This function loops over all the LRM files and, for each LRM file, calls the function `detect_obejcts()` located in `detect_obejcts.py`

### 5.5.12 Compute detection statistics

The source code is in
`detection_statistics.py`
located in
`/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simple-faster-rcnn-pytorch-master`

# 6 Useful utilities

## 6.1 Conversion from LAS files to DTM, DSM, hillshade etc.

Source code:

```
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src
```

Run from command line to produce DTM:

```
./cmd_convert_las_to_dem.pl
/nr/samba/jodata2/pro/cultsearcher/data/trondelag/steinkjer/LAS/
graamyra_vist_2016
```

The output is written to:

```
/nr/samba/jodata2/pro/cultsearcher/data/trondelag/steinkjer/DEM/
graamyra_vist_2016
```

In order to change parameter settings, edit `cmd_las2vegt.pro` and/or `cmd_las2dem.pro`. Both files contain a number of alternative ways to call batch_convert_las2dem, of which all but one is commented out. Select the one that fits your purpose and comment out the remaining.

For the Norwegian datasets, the following was used in cmd_las2dem.pro:

```
batch_convert_las2dem, lasDir, /HILLSHADE, /EXTRA, TO_UTM=33, PIXELSIZE=0.25
```

## 6.2 Conversion from ENVI files to Geotiff

Source code:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/eoTools/src/methods/adhoc/convert_envi_to_geotiff.pro
```

In line 69, change

```
  imagedirs =
["/nr/samba/jodata2/pro/cultsearcher/data/trondelag/steinkjer/DEM/egge_2012"]
```

to list the folders containing the files to be converted. Also, change the file pattern in line 234:

```
     filePattern = imagedir + PATH_SEP() + '*' + suffix
```

preferably by changing lines 227-228:

```
  suffix = '_dsm'
  newSuffix = '_dsm_float.tif'
```

Run from the ENVI prompt by executing:

```
.compile /nr/samba/jo/pro/PilotArran/usr/trier/eoTools/src/methods/adhoc/convert_envi_to_geotiff.pro
.compile /nr/samba/jo/pro/PilotArran/usr/trier/eoTools/src/methods/tools/file_tools.pro
.compile /nr/samba/jo/pro/PilotArran/usr/trier/eoTools/src/methods/tools/basic_tools.pro
convert_envi_to_geotiff
```

# 7 Alternative neural network implementations

This section lists alternatives to the selected *simple faster RCNN*. The alternatives were not used for various reasons.

## 7.1 Python faster R-CNN

Warning! py-faster-rcnn has been deprecated. They advice to use Detectron, which includes Mask R-CNN.

https://github.com/rbgirshick/py-faster-rcnn

## 7.2 Detectron

This could be an alternative. It seems to be able to detect object outlines instead of just bounding boxes.

https://github.com/facebookresearch/Detectron

## 7.3 Mask R-CNN

This is now part of Detectron.

## 7.4 Faster R-CNN

I tried to install this but gave up

https://github.com/jwyang/faster-rcnn.pytorch/tree/pytorch-1.0

### 7.4.1 Installation

```
(.env) trier@jocuda1:~/cultsearcher/frrcnn/faster-rcnn.pytorch

pip install -r requirements.txt

cd lib

python setup.py build
```

### 7.4.2 Pretrained models

(.env) trier@jocuda1:~/cultsearcher/frrcnn/faster-rcnn.pytorch/data$

cp ~/jodata2/pro/cultsearcher/pretrained/vgg16/vgg16_caffe.pth .

cp ~/jodata2/pro/cultsearcher/pretrained/resnet101/resnet101_caffe.pth .

### 7.4.3 Data preparation

trier@jo2:/nr/samba/jodata2/pro/cultsearcher/imgdataset$

wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar

wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtest_06-Nov-2007.tar

wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCdevkit_08-Jun-2007.tar

tar xvf VOCtrainval_06-Nov-2007.tar

tar xvf VOCtest_06-Nov-2007.tar

tar xvf VOCdevkit_08-Jun-2007.tar

(.env) trier@jocuda1:~/cultsearcher/frrcnn/faster-rcnn.pytorch/data$

ln -s ~/jodata2/pro/cultsearcher/imgdataset/VOCdevkit VOCdevkit2007

### 7.4.4 Running the code

```
python trainval_net.py --dataset pascal_voc --net vgg16 --cuda
```

# 8 Results

## 8.1 Results on small test images

Table 8. Number of extracted images.

| class | train | val | test | total |
|---|---|---|---|---|
| grave mound | 720 | 349 | 89 | 1158 |
| pitfall trap | 1307 | 374 | 161 | 1842 |
| charcoal kiln | 908 | 230 | 109 | 1247 |
| total | 2935 | 953 | 359 | 4247 |

Table 9. Detection result on 359 small test images, each containing at least one cultural heritage object.

| | predicted class | | | | | correct | |
|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | back-ground | sum | count | rate |
| grave mound | 159 | 3 | 4 | 71 | 237 | 159 | 67 % |
| pitfall trap | 2 | 268 | 7 | 96 | 373 | 268 | 72 % |
| charcoal kiln | 0 | 0 | 181 | 6 | 187 | 181 | 97 % |
| background | 27 | 66 | 82 | 0 | 175 | | |
| sum objects | | | | | **797** | | |
| **true positives** | | | | | | **608** | **76 %** |
| false negatives | | | | **173** | | | 22 % |
| false positives | | | | | 175 | | 22 % |
| wrong class | | | | | 16 | | 2 % |
| sum predicted | 188 | 337 | 274 | | 799 | | |
| correct | 159 | 268 | 181 | | 608 | | |
| producer's accuracy | 85 % | 80 % | 66 % | | 76 % | | |

### 8.1.1 Implementation details

The script `demo_2_test.py` may be used to evaluate the detection performance on the test data. The following variables were set inside the source code of `demo_2_test.py`:

```
trainer.load('/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simple-faster-rcnn-pytorch-master/checkpoints/fasterrcnn_10101046_0.7610503293336596_3-classes_RA_test_v3_0013')

img_top_dir = '/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_test_v3_0013/'
```

```
test_out_file = os.path.join(test_out_dir,
'RA_test_v3_0013.csv')
```

Command to run:

```
python -W ignore demo_2_test.py
```

By running `demo_2_test.py`, the 359 test images (Table 8, column ‘test’) were fed through the deep neural network. Each test image contained at least one true cultural heritage object. There was one test image for each true cultural heritage object. However, in many cases, more than one cultural heritage object were located inside the same 150 m × 150 m test image. Thus, the test images contained duplicates of many cultural heritage objects. Including duplicates, the total number of true cultural heritage objects inside the test images was 797 (Table 9). Of these 76% were correctly detected, and for the specific classes, grave mound 67%, pitfall trap 72% and charcoal kiln 97%. 22% of the true cultural heritage objects were missed by the method, while 2% were detected with wrong class. 22% of the objects that the method predicted as being cultural heritage were in fact not. However, the latter figure may be an optimistic estimate of the amount of false positives that the method may provide. All the test images contained at least one cultural heritage object. In operational use, there may be large areas, within an ALS dataset, with no cultural heritage objects visible in the data. Thus, the potential for false positives is much larger.

## 8.2 Results with confusion classes added

For the purpose of trying to reduce the number of false positives, confusion classes were added to the training data as follows. Detection was run on the Larvik 2017 dataset, consisting of 113 image tiles of 800 m × 600 m. The detection results were manually checked. False positives of grave mound were labelled 'natural knoll', and false positives of pitfall trap were labelled 'natural pit'. Then, two new versions of the annotated image sets were generated:

1. With one confusion class, natural knoll, in addition to the three true classes, grave mound, pitfall trap and charcoal kiln.

2. With two confusion classes, natural knoll and natural pit, in addition to the three true classes.

For each of these annotated image sets, training was done as described earlier in Section 5.5.8.

By using one confusion class, natural knoll, the true detection rate (consumer's accuracy) was 75% (Table 10). With two confusion classes, natural knoll and natural pit, the true detection rate was 70% (Table 11). Without confusion classes, the true detection rate was 76% (Table 9).

By comparing the three versions also on producer's accuracy (Table 12), adding one or two confusion classes did not improve the producer's accuracy compared to having no confusion classes.

Table 10. Detection result on test images, with one confusion class.

| | predicted class | | | | | | correct | |
|---|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | natural knoll | background | sum | count | rate |
| grave mound | 150 | 3 | 0 | 26 | 58 | 237 | 150 | 63 % |
| pitfall trap | 13 | 270 | 4 | 2 | 84 | 373 | 270 | 72 % |
| charcoal kiln | 1 | 0 | 180 | 0 | 6 | 187 | 180 | 96 % |
| natural knoll | 4 | 0 | 0 | 37 | 12 | 53 | | |
| background | 11 | 119 | 85 | 115 | 0 | 330 | | |
| sum objects | | | | | | **797** | | |
| **true positives** | | | | | | | **600** | **75 %** |
| false negatives | | | | | **176** | | | |
| wrong class | | | | | | 21 | | 3 % |
| sum predicted | 179 | 392 | 269 | | | 840 | | |
| correct | 150 | 270 | 180 | | | 600 | | |
| producer's accuracy | 84 % | 69 % | 67 % | | | 71 % | | |

Table 11. Detection result on test images, with two confusion classes.

| | predicted class | | | | | | | correct | |
|---|---|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | natural knoll | natural pit | background | sum | count | rate |
| grave mound | 155 | 0 | 0 | 14 | 1 | 67 | 237 | 155 | 65 % |
| pitfall trap | 1 | 227 | 4 | 0 | 15 | 126 | 373 | 227 | 61 % |
| charcoal kiln | 0 | 0 | 179 | 0 | 0 | 8 | 187 | 179 | 96 % |
| natural knoll | 2 | 0 | 0 | 30 | 0 | 21 | 53 | | |
| natural pit | 0 | 0 | 0 | 0 | 7 | 5 | 12 | | |
| background | 19 | 47 | 140 | 75 | 65 | 0 | 346 | | |
| sum objects | | | | | | | **797** | | |
| **true positives** | | | | | | | | **561** | **70 %** |
| false negatives | | | | | | | 231 | | |
| wrong class | | | | | | | 5 | | 0,6 % |
| sum predicted | 177 | 274 | 323 | | | | 774 | | |
| correct | 155 | 227 | 179 | | | | 561 | | |
| producer's accuracy | 88 % | 83 % | 55 % | | | | 72 % | | |

Table 12. Summary of results with confusion classes added.

| | no confusion class | | one confusion class | | two confusion classes | |
|---|---|---|---|---|---|---|
| true class | consumer's accuracy | producer's accuracy | consumer's accuracy | producer's accuracy | consumer's accuracy | producer's accuracy |
| grave mound | 67 % | 85 % | 63 % | 84 % | 65 % | 88 % |
| pitfall trap | 72 % | 80 % | 72 % | 69 % | 61 % | 83 % |
| charcoal kiln | 97 % | 66 % | 96 % | 67 % | 96 % | 55 % |
| all classes | 76 % | 76 % | 75 % | 71 % | 70 % | 72 % |

### 8.2.1 Implementation details

To generate annotated image sets, with one and two confusion classes, respectively, the following ENVI/IDL source code was used.

For one confusion class:

```
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/extra
ct_frcnn_training_imgs_v4.pro
```

For two confusion classes:

```
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/extra
ct_frcnn_training_imgs_v5.pro
```

Then, training was done on these two image sets. For the case with two confusion classes, the following code change was used.

In

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simple-faster-rcnn-pytorch-master/utils/config.py
```

lines 21-22:

```
    ## image database with three object classes and two confusion classes:
    ra_data_dir = '/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_test_v5_0013/'
```

Then, training was done by running

```
python -W ignore train.py train --env='fasterrcnn-caffe' --plot-every=100 --caffe-pretrain
```

The neural network parameters were written to file:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simple-faster-rcnn-pytorch-master/checkpoints/fasterrcnn_10081549_0.5844880361754353
```

which we chose to rename as

```
fasterrcnn_10081549_0.5844880361754353_3-classes+2-confusion_RA_test_v5_0013
```

For the case with one confusion class, the following code change was used.

In

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simple-faster-rcnn-pytorch-master/utils/config.py
```

lines 21-22:

```
    ## image database with three object classes and two confusion classes:
    ra_data_dir = '/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_test_v4_0013/'
```

Then, training was done by running

```
python -W ignore train.py train --env='fasterrcnn-caffe' --plot-every=100 --caffe-pretrain
```

The neural network parameters were written to file:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simpl
e-faster-rcnn-pytorch-master/checkpoints/
fasterrcnn_10141216_0.6550685194936133
```

which we chose to rename as

```
fasterrcnn_10141216_0.6550685194936133_3-classes+1-
confusion_RA_test_v5_0013
```

Two experiments were run. The script `demo_4_test.py` used one confusion class, ‘natural knoll’. The script `demo_5_test.py` used two confusion class, ‘natural knoll’ and ‘natural pit’.

Commands to run:

```
python -W ignore demo_4_test.py
```

```
python -W ignore demo_5_test.py
```

## 8.3 Results on small test areas, corrected image extraction

We discovered that several of the labelled cultural heritage objects were missing in the set of extracted images (Table 13). Therefore, the extracted images were re-generated and double-checked. Then, training and testing was re-run.

Table 13. Number of extracted images, before correction.

| class | train | val | test | total |
|---|---|---|---|---|
| grave mound | 720 | 349 | 89 | 1158 |
| pitfall trap | 1307 | 374 | 161 | 1842 |
| charcoal kiln | 908 | 230 | 109 | 1247 |
| total | 2935 | 953 | 359 | 4247 |

Table 14. Number of extracted images, after double-checking.

| class | train | val | test | total |
|---|---|---|---|---|
| grave mound | 719 | 349 | 166 | 1234 |
| pitfall trap | 1306 | 374 | 162 | 1842 |
| charcoal kiln | 908 | 230 | 109 | 1247 |
| total | 2933 | 953 | 437 | 4323 |

One may observe that the number of objects are higher in Table 14 than in Table 3. The numbers in Table 3 are from the vector files, and count the number of objects that are inside each extent. However, Table 14 counts the number of extracted raster images of 600 by 600 pixels. These were extracted from LRM files with a small overlap. Thus, one vector object near an LRM file boundary may be included in more than one LRM file, thus resulting in more than one extracted raster image.

With the corrected image extraction, the main difference was that the number of grave mound images in the test subset was increased from 89 (Table 13) to 166 (Table 14). As a result, 406 out of 477 grave mound objects (85%, Table 15) were correctly predicted, compared to 159 of 237 (67%, Table 9). All in all, the correct classification rate (consumer's accuracy) increased from 76% (Table 9) to 84% (Table 15). At the same time, the producer's accuracy was reduced from 76% (Table 9) to 67% (Table 15).

Table 15. Detection result on 437 small test images, each containing at least one cultural heritage object.

| | predicted class | | | | | correct | |
|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | back-ground | sum | count | rate |
| grave mound | 406 | 7 | 0 | 64 | 477 | 406 | 85 % |
| pitfall trap | 6 | 292 | 2 | 86 | 386 | 292 | 76 % |
| charcoal kiln | 2 | 0 | 179 | 6 | 187 | 179 | 96 % |
| background | 184 | 125 | 104 | 0 | 413 | | |
| sum objects | | | | | **1050** | | |
| **true positives** | | | | | | **877** | **84 %** |
| false negatives | | | | **156** | | | 15 % |
| false positives | | | | | 413 | | 39 % |
| wrong class | | | | | 17 | | 2 % |
| sum predicted | 598 | 424 | 285 | | 1307 | | |
| correct | 406 | 292 | 179 | | 877 | | |
| producer's accuracy | 68 % | 69 % | 63 % | | 67 % | | |

### 8.3.1 Implementation details

To generate annotated image sets, the following ENVI/IDL source code was used.

```
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/
extract_frcnn_training_imgs_v3_0020.pro
```

Then, training was done on this image set. The following code change was used.

In

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simpl
e-faster-rcnn-pytorch-master/utils/config.py
```

Line 14:

```
    ra_data_dir =
'/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_
test_v3_0020/'
```

Then, training was done by running

```
python -W ignore train.py train --env='fasterrcnn-caffe' --plot-
every=100 --caffe-pretrain
```

The neural network parameters were written to file:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simpl
e-faster-rcnn-pytorch-master/checkpoints/
fasterrcnn_12181825_0.763320822866815
```

which we chose to rename as

```
fasterrcnn_12181825_0.763320822866815_3-classes_RA_test_v3_0020
```

The script `demo_2_test_v3_0020.py` is a modified version of `demo_2_test.py` with the following code changes.

In line 28:

```
trainer.load('/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_sr
c/sfrcnn/simple-faster-rcnn-pytorch-
master/checkpoints/fasterrcnn_12181825_0.763320822866815_3-
classes_RA_test_v3_0020')
```

In line 41:

```
img_top_dir =
'/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_
test_v3_0020/'
```

In line 52:

```
test_out_file = os.path.join(test_out_dir, 'RA_test_v3_0020.csv')
```

Command to run:

```
python -W ignore demo_2_test_v3_0020.py
```

## 8.4 Results on small test areas, alternative setup

Table 16. Number of extracted images, alternative setup.

| class | train | val | test | total |
|---|---|---|---|---|
| grave mound | 616 | 359 | 248 | 1223 |
| pitfall trap | 774 | 682 | 379 | 1835 |
| charcoal kiln | 907 | 230 | 110 | 1247 |
| total | 2297 | 1271 | 737 | 4305 |

Table 17. Detection result on 737 small test images, each containing at least one cultural heritage object

| | predicted class | | | | | correct | |
|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | back-ground | sum | count | rate |
| grave mound | 522 | 1 | 1 | 191 | 715 | 522 | 73 % |
| pitfall trap | 4 | 1009 | 0 | 228 | 1241 | 1009 | 81 % |
| charcoal kiln | 0 | 0 | 176 | 12 | 188 | 176 | 94 % |
| background | 137 | 154 | 22 | 0 | 313 | | |
| sum objects | | | | | **2144** | | |
| **true positives** | | | | | | **1707** | **80 %** |
| false negatives | | | | **431** | | | 20 % |
| false positives | | | | | 313 | | 15 % |
| wrong class | | | | | 6 | | 0,3 % |
| sum predicted | 663 | 1164 | 199 | | 2026 | | |
| correct | 522 | 1009 | 176 | | 1707 | | |
| producer's accuracy | 79 % | 87 % | 88 % | | 84 % | | |

### 8.4.1 Implementation details

To generate annotated image sets the following ENVI/IDL source code was used.

```
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/extra
ct_frcnn_training_imgs_v1.pro
```

In lines 66-67, make sure that the number of augmentation is set to 1:

```
  num_augmentations = 1

  ;;num_augmentations = 8
```

In line 946:

```
  out_image_dir =
"/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/
RA_test_v1_0008_alt-subdiv"
```

From the ENVI/IDL prompt, execute (no line breaks within each command):

```
.COMPILE
/nr/samba/jo/pro/cultsearcher2018/usr/trier/eoTools/src/methods/
tools/file_tools.pro

.COMPILE
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/cult_
merge_files.pro

.COMPILE
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/extra
ct_frcnn_training_imgs_v1.pro

extract_frcnn_training_imgs_v1
```

The test images have been created in a directory structure under:

```
/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_t
est_v1_0008
```

Then, training was done on this image set. The following code changes were used.

In

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simpl
e-faster-rcnn-pytorch-master/utils/config.py
```

lines 10-11:

```
    ## image database with three object classes and no confusion
classes, alternative setup:

    ra_data_dir =
'/home/trier/jodata10/pro/CultSearcher/imgdataset/kulturminner/
RA_test_v1_0008/'
```

Then, training was done by running

```
python -W ignore train.py train --env='fasterrcnn-caffe' --plot-
every=100 --caffe-pretrain
```

The neural network parameters were written to file:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simpl
e-faster-rcnn-pytorch-master/checkpoints/
fasterrcnn_11161914_0.7948160634148803
```

This file was renamed as

```
fasterrcnn_11161914_0.7948160634148803_3-
classes_RA_test_v1_0008_alt_subdiv
```

The script `demo_1_test.py` was used to run detection on the set of small test images.

Make sure the correct file with saved neural parameters is loaded. Change lines 21-22 if needed:

```
#Training with 3 classes and no confusion classes:
trainer.load('/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_sr
c/sfrcnn/simple-faster-rcnn-pytorch-
master/checkpoints/fasterrcnn_04041754_0.7982244621705039')
```

Command to run:

```
python -W ignore demo_1_test.py
```

## 8.5 Results on small test areas, alternative setup and eight rotation/flip combinations

Table 18. Number of extracted images, alternative setup and eight rotation/flip combinations.

| class | train | val | test | total |
|---|---|---|---|---|
| grave mound | 4928 | 2880 | 1984 | 9792 |
| pitfall trap | 6192 | 5464 | 3040 | 14696 |
| charcoal kiln | 7256 | 1840 | 864 | 9960 |
| total | 18376 | 10184 | 5888 | 34448 |

Training of the neural network was done on the extracted images with the eight rotation/flip combinations (Table 18, columns 'train' and 'val'). However, testing was done on the un-rotated and non-flipped images (Table 16, column 'test').

Table 19. Detection result on 737 small test images, each containing at least one cultural heritage object

| | predicted class | | | | | correct | |
|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | back-ground | sum | count | rate |
| grave mound | 603 | 0 | 3 | 109 | 715 | 603 | 84 % |
| pitfall trap | 6 | 1073 | 1 | 161 | 1241 | 1073 | 86 % |
| charcoal kiln | 1 | 0 | 180 | 7 | 188 | 180 | 96 % |
| background | 252 | 267 | 80 | 0 | 599 | | |
| positives | | | | | **2144** | | |
| **true positives** | | | | | | **1856** | **87 %** |
| false negatives | | | | **277** | | | 13 % |
| false positives | | | | | 599 | | 28 % |
| wrong class | | | | | 11 | | 0,5 % |
| sum predicted | 862 | 1340 | 264 | | 2466 | | |
| correct | 603 | 1073 | 180 | | 1856 | | |
| producer's accuracy | 70 % | 80 % | 68 % | | 75 % | | |

### 8.5.1 Implementation details

To generate annotated image sets the following ENVI/IDL source code was used.

```
/nr/samba/jo/pro/CultSearcher/Usr/Trier/CultSearcher09/Src/
extract_frcnn_training_imgs_v1.pro
```

On line 937:

```
  num_augmentations = 8
```

On line 946 (no line breaks):

```
  out_image_dir =
"/nr/samba/jodata10/pro/CultSearcher/imgdataset/
kulturminner/RA_test_augment-8_0001_alt-subdiv"
```

The resulting image set was used for training/validation. However, for testing, rotated/flipped versions of the images were not needed.

Then, training was done on the image set. The following code changes were used.

In

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simpl
e-faster-rcnn-pytorch-master/utils/config.py
```

line 13:

```
    ra_data_dir =
'/nr/samba/jodata10/pro/CultSearcher/imgdataset/kulturminner/RA_
test_augment-8_0001_alt-subdiv/'
```

Then, training was done by running

```
python -W ignore train.py train --env='fasterrcnn-caffe' --plot-
every=100 --caffe-pretrain
```

The neural network parameters were written to file:

```
/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/
simple-faster-rcnn-pytorch-master/checkpoints/
fasterrcnn_11172341_0.8056864563757405
```

This file was renamed to (no line breaks):

```
fasterrcnn_11172341_0.8056864563757405_3-
classes_RA_test_augment-8_0001_alt-subdiv
```

The script `demo_1_test.py` was used to run detection on the set of small test images.

Make sure the correct file with saved neural parameters is loaded. Change lines 21-22 if needed, no line breaks within the file path:

```
#Training with 3 classes and no confusion classes:
trainer.load('/nr/samba/jo/pro/cultsearcher2018/usr/trier/
```

```
jo1_src/sfrcnn/simple-faster-rcnn-pytorch-master/
checkpoints/fasterrcnn_11172341_0.8056864563757405_3-
classes_RA_test_augment-8_0001_alt-subdiv')
```

Make sure the correct output file name is used in line 52:

```
test_out_file = os.path.join(test_out_dir,
'RA_test_augment-8_0001_alt_subdiv.csv')
```

Command to run:

```
python -W ignore demo_1_test.py
```

## 8.6 Results on larger areas

Table 20. Test set for evaluation of detection method on large areas. The number of files refers to the number of 800 m × 600 m image tiles.

| object type | dataset | subset | object count | extent of dataset in UTM zone 33 N | | | | number of files |
|---|---|---|---|---|---|---|---|---|
| | | | | west | east | south | north | |
| charcoal kiln | Lesja 2013 | test | 95 | 144 800 | 154 400 | 6 916 000 | 6 922 800 | 87 |
| grave mound | Brumunddal 2016 part 1 | test | 50 | 269 600 | 283 200 | 6 736 200 | 6 753 000 | 358 |
| grave mound | Brumunddal 2016 part 2 | test | 23 | 260 000 | 280 000 | 6 753 600 | 6 774 600 | 755 |
| grave mound | Larvik 2017 | test | 57 | 220 800 | 226 400 | 6 553 200 | 6 565 200 | 113 |
| pitfall trap | Dovre 2013 | test | 29 | 190 400 | 204 000 | 6 878 400 | 6 897 000 | 94 |
| pitfall trap | Dovre 2017 | test | 15 | 190 400 | 196 800 | 6 882 000 | 6 897 000 | 139 |
| pitfall trap | Dovre Folldal 2018 | test | 3 | 233 600 | 234 400 | 6 891 600 | 6 892 200 | 1 |
| pitfall trap | Nordfron 2013 | test | 6 | 191 200 | 195 200 | 6 831 000 | 6 832 200 | 12 |
| pitfall trap | Nordfron Olstappen 2010 | test | 41 | 195 200 | 202 400 | 6 830 400 | 6 832 200 | 21 |
| Total number of files | | | | | | | | 1580 |

The detection module was run on collections of 800 m × 600 m image files (Table 20). The true detection rate was 81% (Table 21). This was better than on the small test images (76%, Table 9). However, the set of small test images contained duplicates of many of the true objects. If the duplicates had been removed, then we would expect the true detection rate to be 81% also on the set of small test images.

A more important difference is that the number of false positives is now more than ten times the number of labelled true objects in the test data. Or to put it another way, the producer's accuracy is 5%. This means that out of the 5269 predicted objects, only 5% are true objects with correctly predicted class. 95% of the predicted objects do not match any labelled objects. This could potentially limit the usefulness of the automated detection method.

Confusion between classes is a minor problem, and occurs for 6 of the 319 true objects (2%).

Table 21. Detection results on large areas.

| | predicted class | | | | | correct | |
|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | background | sum | count | rate |
| grave mound | **95** | 2 | 3 | 30 | 130 | 95 | 73 % |
| pitfall trap | 1 | **72** | 0 | 21 | 94 | 72 | 77 % |
| charcoal kiln | 0 | 0 | **90** | 5 | 95 | 90 | 95 % |
| background | 1746 | 2076 | 1184 | 0 | 5006 | | |
| sum objects | | | | | **319** | | |
| **true positives** | | | | | | **257** | **81 %** |
| false negatives | | | | **56** | | | 18 % |
| false positives | | | | | 5006 | | 1569 % |
| wrong class | | | | | 6 | | 2 % |
| sum predicted | 1842 | 2150 | 1277 | | 5269 | | |
| correct | 95 | 72 | 90 | | 257 | | |
| producer's accuracy | 5 % | 3 % | 7 % | | 5 % | | |

Table 22. Confusion matrix for individual large areas of labelled test data. CK=charcoal kiln, GM=grave mound, PT=pitfall trap, Bg=background.

| | true class -> predicted class | | | | | | |
|---|---|---|---|---|---|---|---|
| dataset | CK->GM | CK->PT | CK->CK | CK->Bg | Bg->GM | Bg->PT | Bg->CK |
| Lesja 2013 | 0 | 0 | 90 | 5 | 75 | 61 | 807 |
| dataset | GM->GM | GM->PT | GM->CK | GM->Bg | Bg->GM | Bg->PT | Bg->CK |
| Brumunddal 2016 part 1 | 35 | 0 | 3 | 12 | 329 | 354 | 111 |
| Brumunddal 2016 part 2 | 15 | 1 | 0 | 7 | 866 | 1282 | 504 |
| Larvik 2017 | 45 | 1 | 0 | 11 | 420 | 153 | 40 |
| dataset | PT->GM | PT->PT | PT->CK | PT->Bg | Bg->GM | Bg->PT | Bg->CK |
| Dovre 2013 | 1 | 25 | 0 | 3 | 41 | 67 | 175 |
| Dovre 2017 | 0 | 10 | 0 | 5 | 87 | 197 | 185 |
| Dovre Folldal 2018 | 0 | 3 | 0 | 0 | 0 | 3 | 1 |
| Nordfron 2013 | 0 | 2 | 0 | 4 | 3 | 20 | 168 |
| Nordfron Olstappen 2010 | 0 | 32 | 0 | 9 | 5 | 34 | 11 |

Table 23. Detection rates for individual large areas of labelled test data.

| | | true positives | | false negatives | | wrong class | |
|---|---|---|---|---|---|---|---|
| dataset with charcoal kilns | sum objects | count | rate | count | rate | count | rate |
| Lesja 2013 | 95 | 90 | 95 % | 5 | 5 % | 0 | 0 % |
| datasets with grave mounds | | | | | | | |
| Brumunddal 2016 part 1 | 50 | 35 | 70 % | 12 | 24 % | 3 | 6 % |
| Brumunddal 2016 part 2 | 23 | 15 | 65 % | 7 | 30 % | 1 | 4 % |
| Larvik 2017 | 57 | 45 | 79 % | 11 | 19 % | 1 | 2 % |
| datasets with pitfall traps | | | | | | | |
| Dovre 2013 | 29 | 25 | 86 % | 3 | 10 % | 1 | 3 % |
| Dovre 2017 | 15 | 10 | 67 % | 5 | 33 % | 0 | 0 % |
| Dovre Folldal 2018 | 3 | 3 | 100 % | 0 | 0 % | 0 | 0 % |
| Nordfron 2013 | 41 | 32 | 78 % | 9 | 22 % | 0 | 0 % |
| Nordfron Olstappen 2010 | 6 | 2 | 33 % | 4 | 67 % | 0 | 0 % |

Table 24. Producer's accuracy for individual large areas of labelled test data.

| dataset with charcoal kilns | true positives | sum predicted | producer's accuracy | number of files | true objects per file | producer's accuracy / true objects per file |
|---|---|---|---|---|---|---|
| Lesja 2013 | 90 | 1033 | 9 % | 87 | 1,09 | 8 % |
| datasets with grave mounds | | | | | | |
| Brumunddal 2016 part 1 | 35 | 832 | 4 % | 358 | 0,14 | 30 % |
| Brumunddal 2016 part 2 | 15 | 2668 | 1 % | 755 | 0,03 | 18 % |
| Larvik 2017 | 45 | 659 | 7 % | 113 | 0,50 | 14 % |
| datasets with pitfall traps | | | | | | |
| Dovre 2013 | 25 | 309 | 8 % | 94 | 0,31 | 26 % |
| Dovre 2017 | 10 | 479 | 2 % | 139 | 0,11 | 19 % |
| Dovre Folldal 2018 | 3 | 7 | 43 % | 1 | 3,00 | 14 % |
| Nordfron 2013 | 32 | 82 | 39 % | 12 | 3,42 | 11 % |
| Nordfron Olstappen 2010 | 2 | 193 | 1 % | 21 | 0,29 | 4 % |

### 8.6.1 Implementation details

Changes in config.py

Lines 12-13:

```
workdir = "/nr/samba/jodata10/pro/CultSearcher/usr/trier/detection_v3/work",
resultdir = "/nr/samba/jodata10/pro/CultSearcher/usr/trier/detection_v3/results",
```

Line 46:

```
"model_file":
'/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simp
le-faster-rcnn-pytorch-
```

```
master/checkpoints/fasterrcnn_10101046_0.7610503293336596_3-
classes_RA_test_v3_0013'
```

Commands to run detection on the test sets:

```
python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/lidar/oppland/lesja_2013_utm33/LAS/lesja_test
lesja_2013_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
larvik_2017_test_hele/LAS/larvik_2017 larvik_2017_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
brumunddal_2016_test_hele/LAS/brumunddal_2016_test_hele
brumunddal_2016_test_1

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
brumunddal_2016_val_hele/LAS/brumunddal_2016_val_hele
brumunddal_2016_test_2

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/dovre_nordgudbrandsdal_2013_test_hele/LAS/dovre_2013
dovre_2013_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/lesja_vaga_test_hele/LAS/lesja_vaga_test_hele dovre_2017_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/lidar/oppland/dovre/LAS/folldal_2018_utm33
dovre_folldal_2018_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/nordfron_nordgudbrandsdal_2013/LAS/nordfron_2013_test
nordfron_2013_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/ols
tappen_test nordfron_olstappen_2010_test
```

Command to get detection statistics:

```
(.env)
trier@jo1:/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sf
rcnn/simple-faster-rcnn-pytorch-master$
python detection_statistics.py
```

The test set of large areas consists of the following directories:

Larvik 2017:
`/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/larvik-hele/LAS/larvik-hele`

Brumunddal 2016:
`/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/brumunddal_2016_test_hele/LAS/brumunddal_2016_test_hele`

`/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/brumunddal_2016_val_hele/LAS/brumunddal_2016_val_hele`

Nordfron Olstappen 2010:
`/nr/samba/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/olstappen_test`

Dovre 2013:
`/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgrop/dovre_nordgudbrandsdal_2013_test_hele/LAS/dovre_2013`

Dovre 2017:
`/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgrop/lesja_vaga_test_hele/LAS/lesja_vaga_test_hele`

Dovre Folldal 2018:
`/nr/samba/jodata10/lidar/oppland/dovre/LAS/folldal_2018_utm33`

Nordfron 2013:
`/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgrop/nordfron_nordgudbrandsdal_2013/LAS/nordfron_2013_test`

Lesja 2013:
`/nr/samba/jodata10/lidar/oppland/lesja_2013_utm33/LAS/lesja_test`

### 8.6.2 Overview maps of ALS test datasets

Overview maps of the test areas appear in Figure 73-Figure 81.

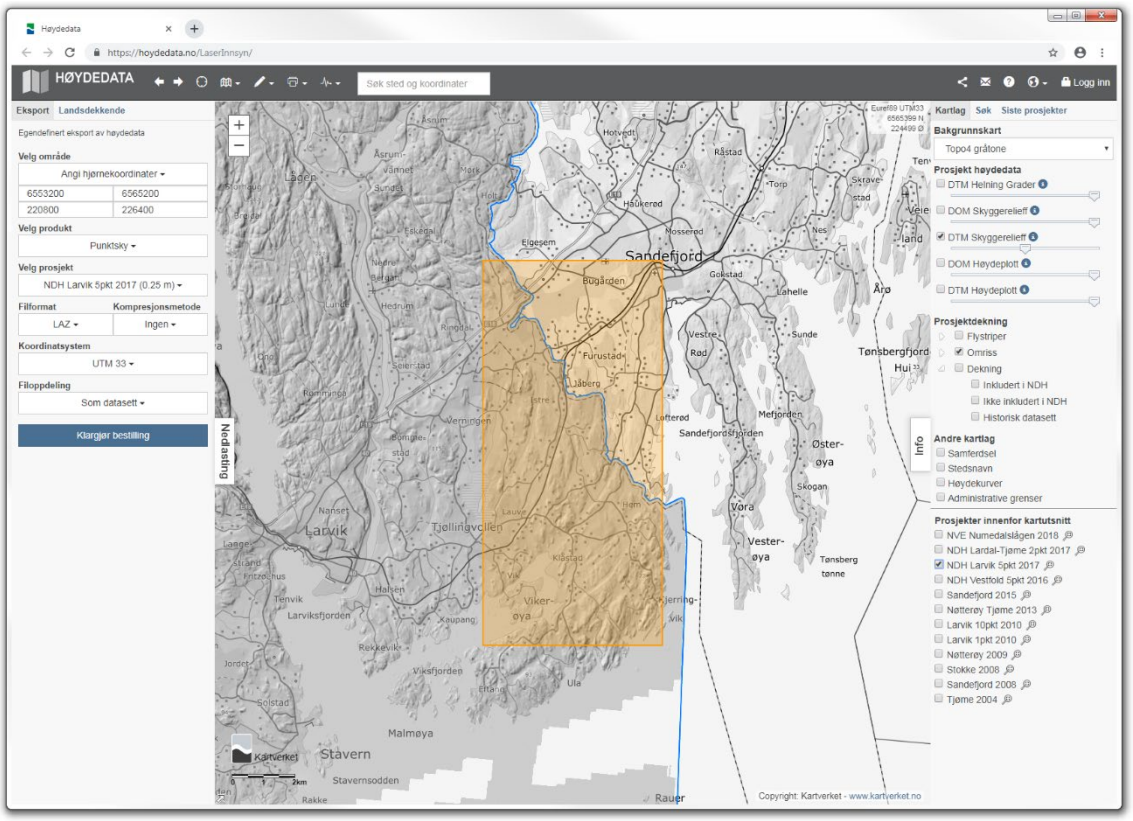



Figure 73. Larvik 2017 dataset, test subset.

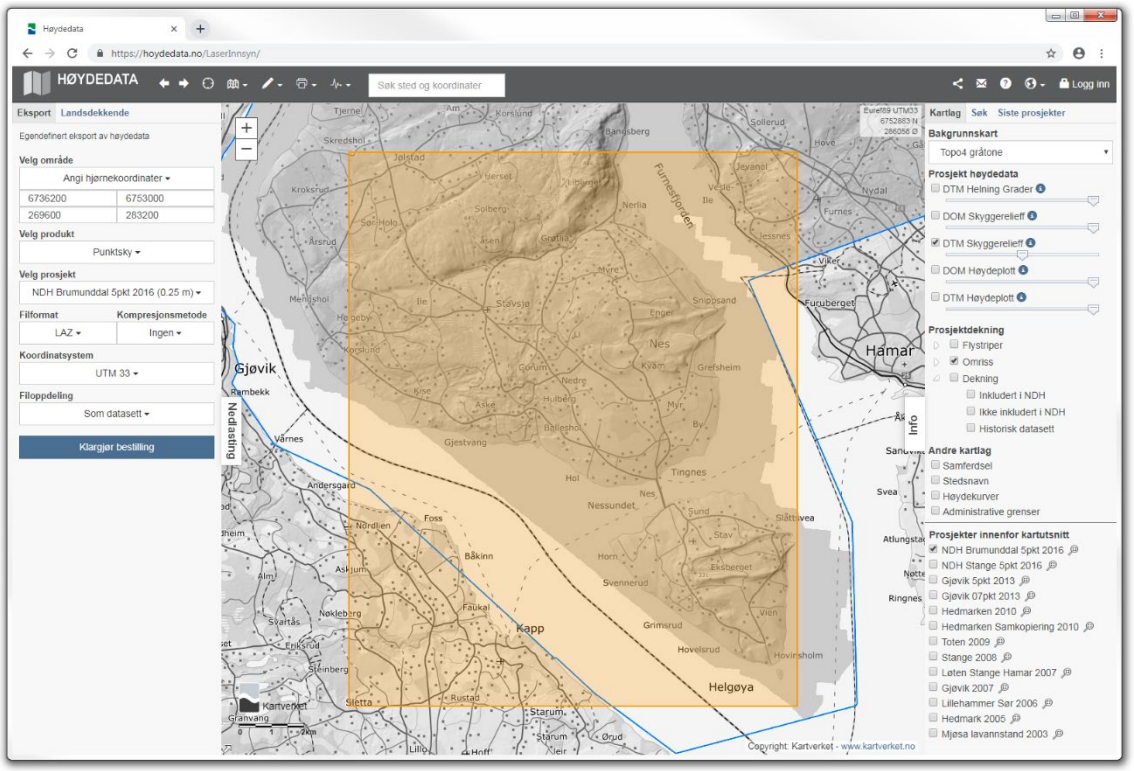



Figure 74. Brumunddal 2016 dataset, test subset, part 1.

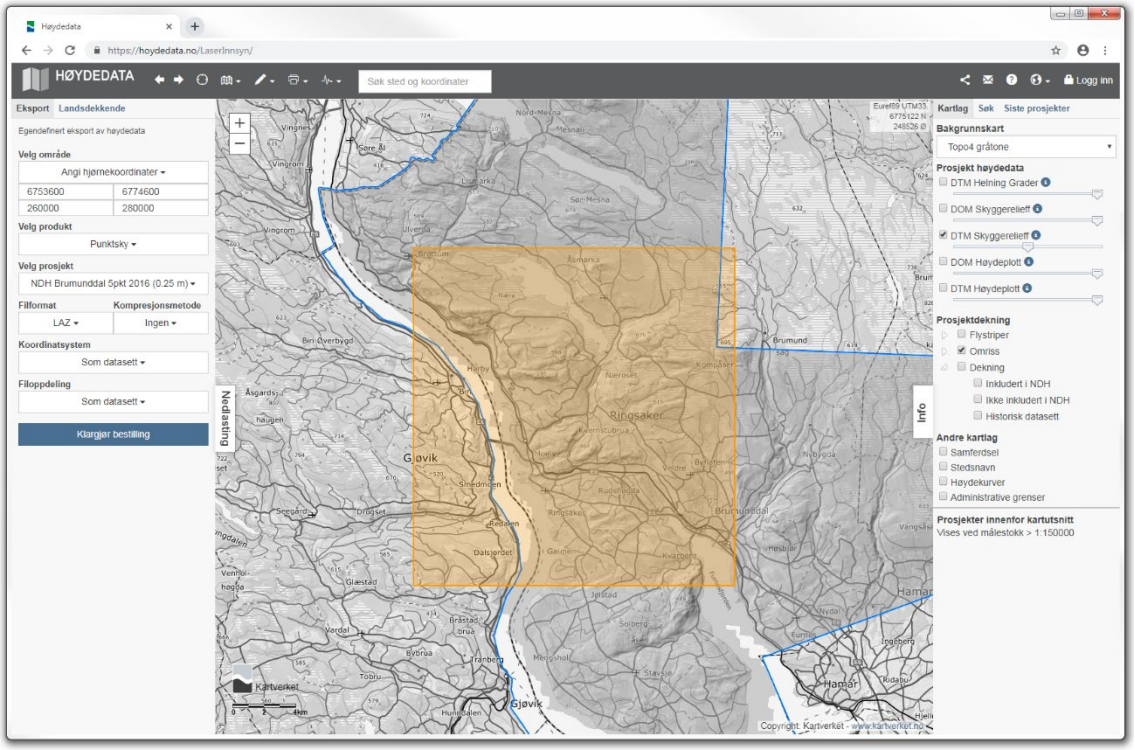



Figure 75. Brumunddal 2016 dataset, test subset, part 2.

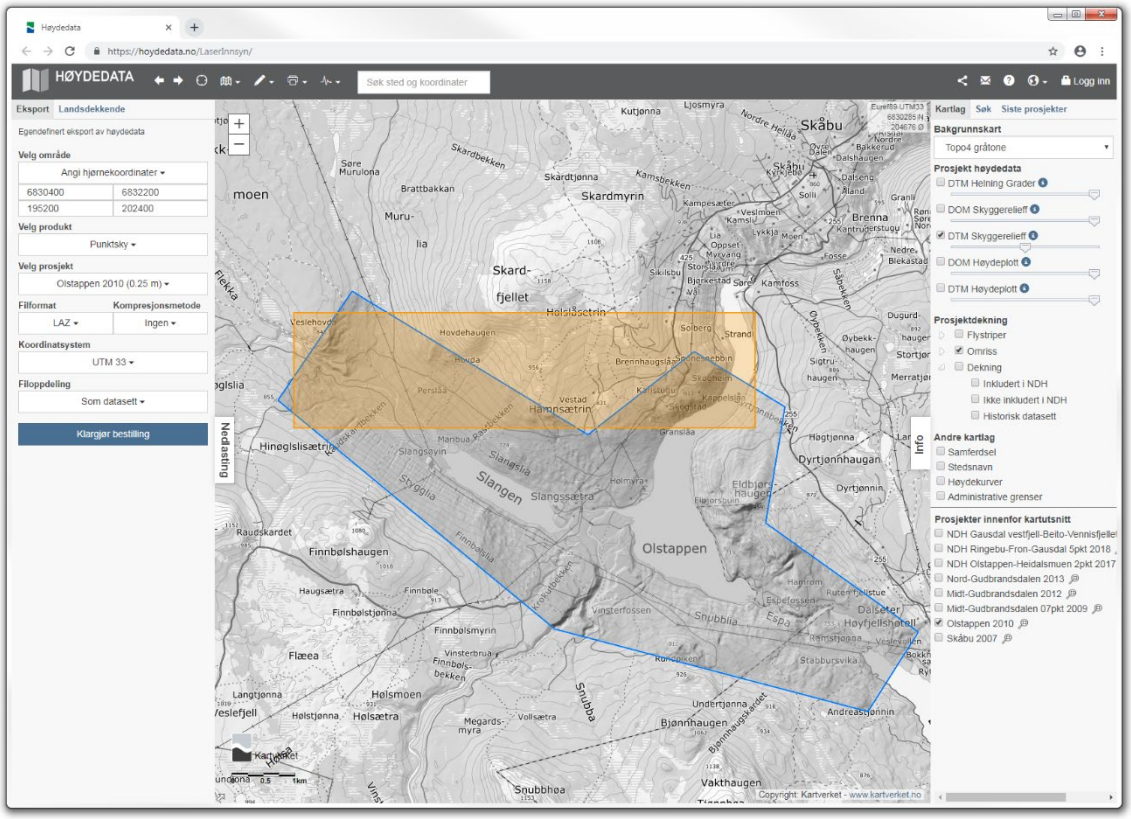



Figure 76. Nordfron Olstappen 2010 dataset, test subset.

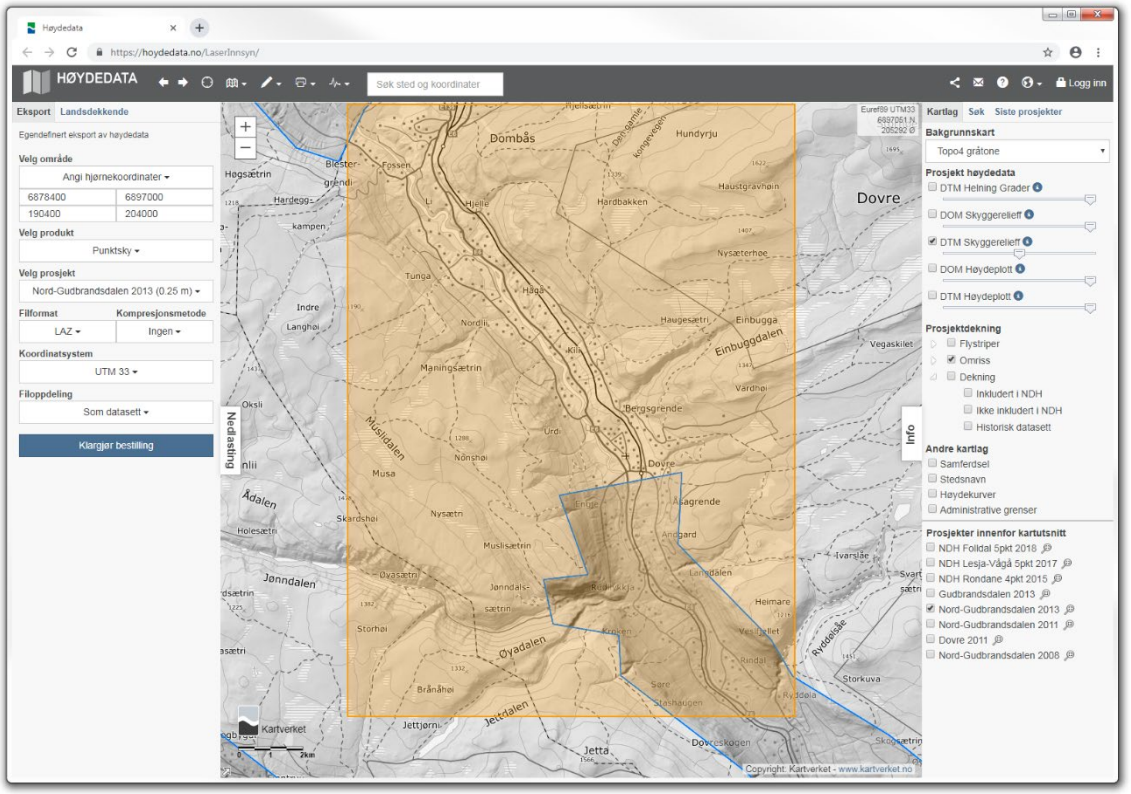



Figure 77. Dovre 2013 dataset, test subset.

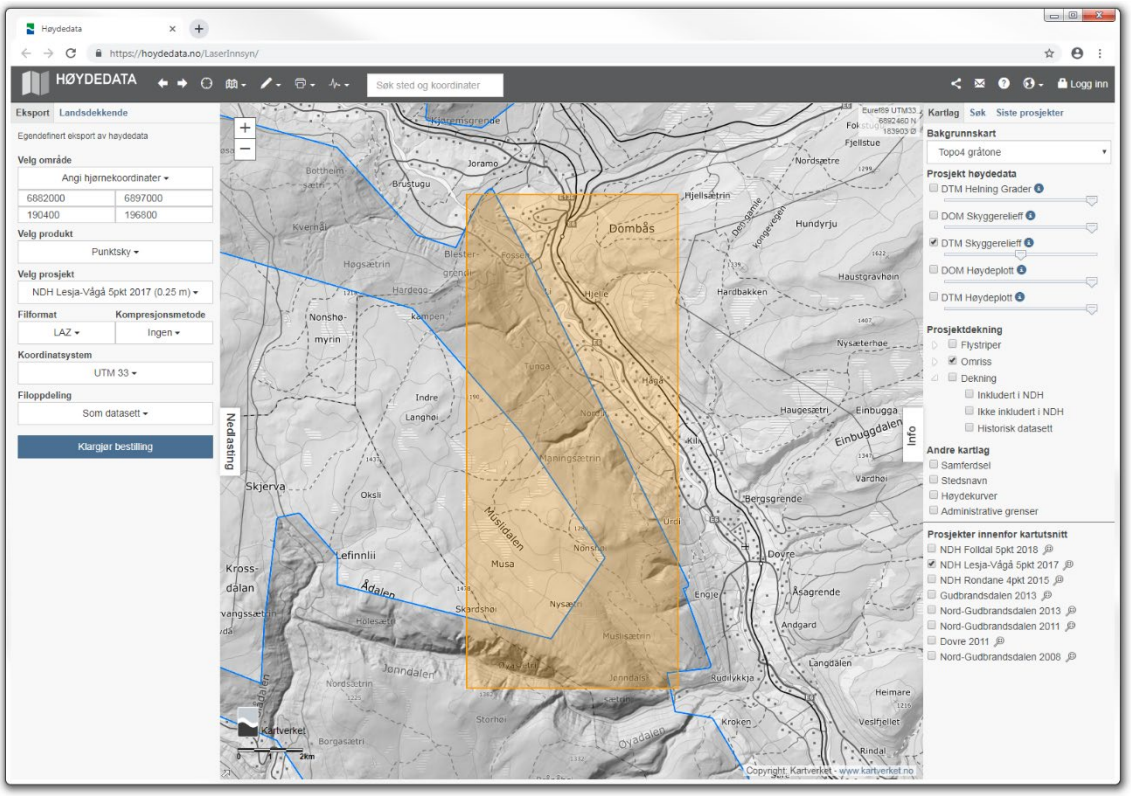



Figure 78. Dovre 2017 dataset, test subset.

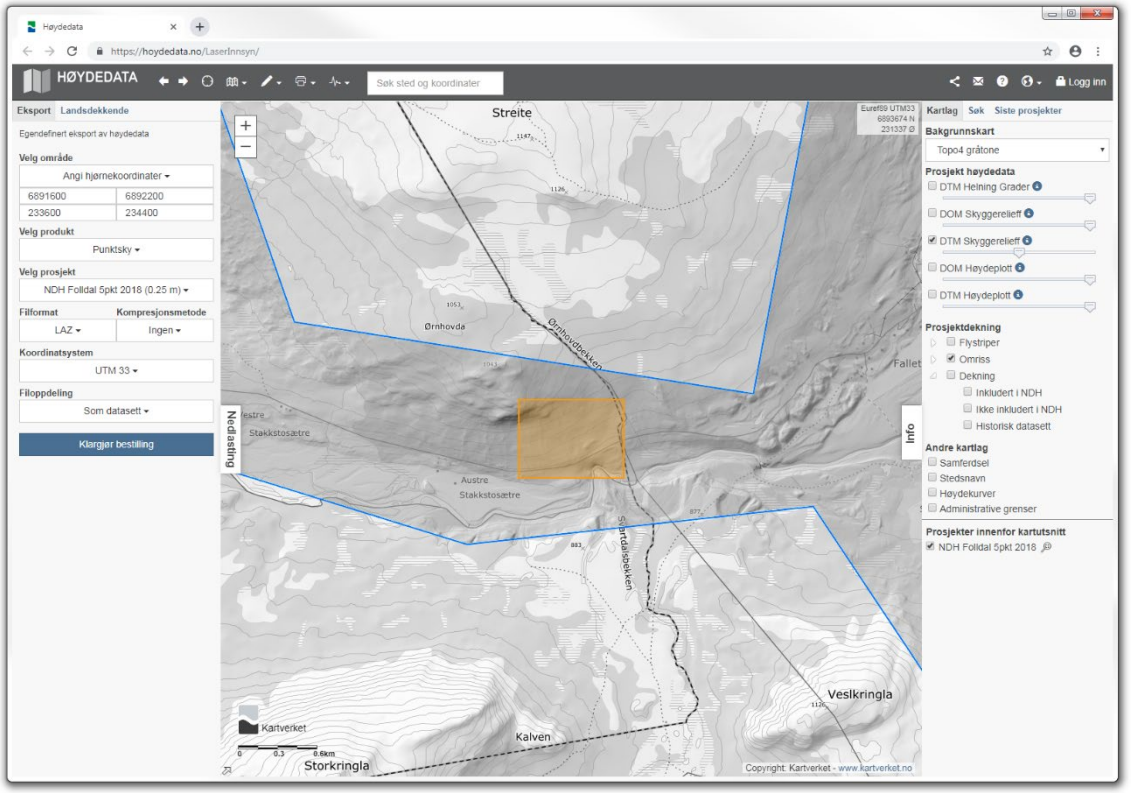



Figure 79. Dovre Folldal 2018 dataset, test subset.

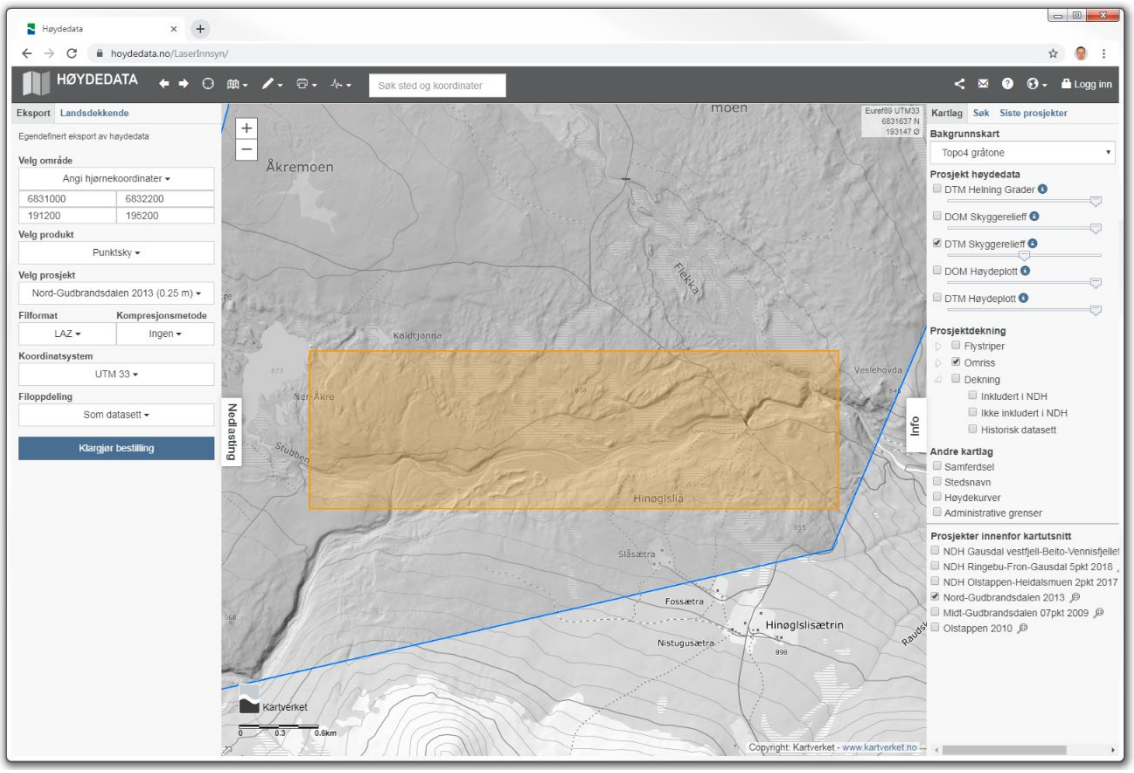



Figure 80. Nordfron 2013 dataset, test subset.

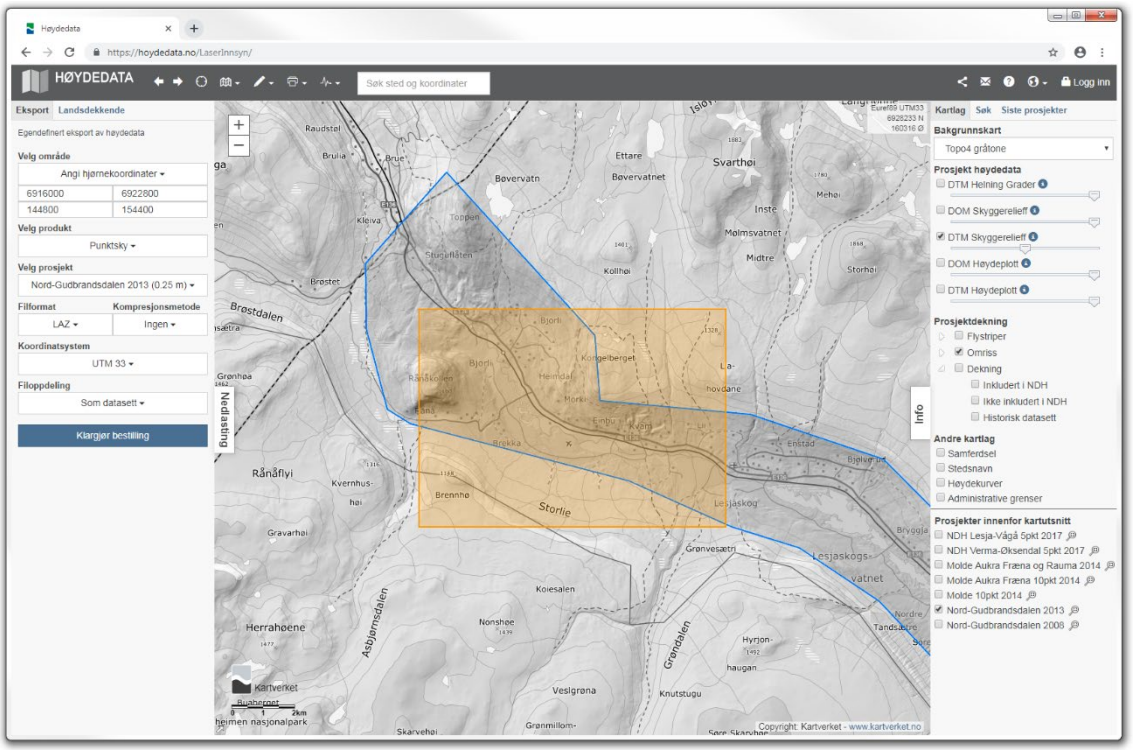



Figure 81. Lesja 2013 dataset, test subset.

## 8.7 Result on larger areas, alternative setup

Table 25. Test set for evaluation of detection method on large areas, alternative setup. The number of files refers to the number of 800 m × 600 m image tiles.

| object type | dataset | subset | object count | extent of dataset in UTM zone 33 N | | | | number of files |
|---|---|---|---|---|---|---|---|---|
| | | | | west | east | south | north | |
| charcoal kiln | Lesja 2013 | test | 95 | 144 800 | 154 400 | 6 916 000 | 6 922 800 | 87 |
| grave mound | Brumunddal 2016 part 1 | test | 50 | 269 600 | 283 200 | 6 736 200 | 6 753 000 | 358 |
| grave mound | Larvik 2017 | test | 57 | 220 800 | 226 400 | 6 553 200 | 6 565 200 | 113 |
| grave mound | Sarpsborg 2015 | test | 18 | 276 000 | 284 800 | 6 565 200 | 6 575 400 | 7 |
| grave mound | Steinkjer 2011 | test | 30 | 321 600 | 348 800 | 7 087 800 | 7 113 000 | 313 |
| grave mound | Steinkjer 2017 | test | 44 | 322 400 | 345 600 | 7 097 400 | 7 119 600 | 59 |
| pitfall trap | Dovre 2013 | test | 29 | 190 400 | 204 000 | 6 878 400 | 6 897 000 | 94 |
| pitfall trap | Dovre 2017 | test | 15 | 190 400 | 196 800 | 6 882 000 | 6 897 000 | 139 |
| pitfall trap | Dovre Folldal 2018 | test | 3 | 233 600 | 234 400 | 6 891 600 | 6 892 200 | 1 |
| pitfall trap | Dovre Grims-dalen 2010 | test | 155 | 219 200 | 231 200 | 6 893 400 | 6 899 400 | 36 |
| pitfall trap | Nordfron 2012 | test | 31 | 200 800 | 219 200 | 6 833 400 | 6 840 600 | 7 |
| pitfall trap | Nordfron 2013 | test | 6 | 191 200 | 195 200 | 6 831 000 | 6 832 200 | 12 |
| pitfall trap | Nordfron 2017 | test | 1 | 211 200 | 212 000 | 6 831 000 | 6 831 600 | 1 |
| pitfall trap | Nordfron 2018 | test 1 | 6 | 208 800 | 210 400 | 6 841 200 | 6 841 800 | 2 |
| | | test 2 | 9 | 197 600 | 199 200 | 6 821 400 | 6 822 600 | 2 |
| pitfall trap | Nordfron Olstappen 2010 | test | 41 | 195 200 | 202 400 | 6 830 400 | 6 832 200 | 21 |
| pitfall trap | Nordfron Venabu 2018 | test | 7 | 224 000 | 227 200 | 6 844 800 | 6 858 600 | 4 |
| Total number of files | | | | | | | | 1256 |

With an alternative subdivision of the datasets into training, validation and test subsets (Table 5, Table 25), the correct classification rate (consumer's accuracy) seemed to increase from 81% (Table 21) to 83% (Table 26). However, the two test data sets are not identical. At the same time, the producer's accuracy improved from 5% to 6%.

Table 26. Detection results on large areas.

| | predicted class | | | | | correct | |
|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | background | sum | count | rate |
| grave mound | **152** | 1 | 1 | 45 | 199 | 152 | 76 % |
| pitfall trap | 0 | **260** | 0 | 43 | 303 | 260 | 86 % |
| charcoal kiln | 0 | 0 | **83** | 12 | 95 | 83 | 87 % |
| background | 3770 | 2950 | 1270 | 0 | 7990 | | |
| sum objects | | | | | **597** | | |
| **true positives** | | | | | | **495** | **83 %** |
| false negatives | | | | **100** | | | 17 % |
| false positives | | | | | 7990 | | 1338 % |
| wrong class | | | | | 2 | | 0,3 % |
| sum predicted | 3922 | 3211 | 1354 | | 8487 | | |
| correct | 152 | 260 | 83 | | 495 | | |
| producer's accuracy | 4 % | 8 % | 6 % | | 6 % | | |

Table 27. Confusion matrix for individual large areas of labelled test data. CK=charcoal kiln, GM=grave mound, PT=pitfall trap, Bg=background.

| dataset | CK->GM | CK->PT | CK->CK | CK->Bg | Bg->GM | Bg->PT | Bg->CK |
|---|---|---|---|---|---|---|---|
| Lesja 2013 | 0 | 0 | 83 | 12 | 81 | 92 | 262 |
| dataset | GM->GM | GM->PT | GM->CK | GM->Bg | Bg->GM | Bg->PT | Bg->CK |
| Brumunddal 2016 | 41 | 0 | 0 | 9 | 435 | 330 | 29 |
| Larvik 2017 | 44 | 0 | 0 | 13 | 369 | 121 | 20 |
| Sarpsborg 2015 | 14 | 0 | 0 | 4 | 157 | 62 | 45 |
| Steinkjer 2011 | 17 | 0 | 1 | 12 | 1222 | 162 | 366 |
| Steinkjer 2015 | 36 | 1 | 0 | 7 | 781 | 520 | 247 |
| dataset | PT->GM | PT->PT | PT->CK | PT->Bg | Bg->GM | Bg->PT | Bg->CK |
| Dovre 2013 | 0 | 25 | 0 | 4 | 57 | 68 | 50 |
| Dovre 2017 | 0 | 12 | 0 | 3 | 66 | 206 | 58 |
| Dovre Folldal 2018 | 0 | 3 | 0 | 0 | 0 | 1 | 0 |
| Dovre Grimsdalen 2018 | 0 | 134 | 0 | 21 | 29 | 128 | 2 |
| Nordfron 2012 | 0 | 27 | 0 | 4 | 280 | 415 | 85 |
| Nordfron 2013 | 0 | 3 | 0 | 3 | 2 | 1 | 2 |
| Nordfron 2017 | 0 | 1 | 0 | 0 | 2 | 3 | 4 |
| Nordfron 2018 | 0 | 12 | 0 | 3 | 197 | 698 | 76 |
| Nordfron Olstappen 2010 | 0 | 36 | 0 | 5 | 6 | 29 | 7 |
| Nordfron Venabu 2018 | 0 | 7 | 0 | 0 | 86 | 114 | 17 |

## 8.8 Result on larger areas, alternative setup and eight rotation/flip combinations during training

By adding rotated and flipped versions of all images during training, but not during testing, the correct classification rate (consumer's accuracy) increased from 83% (Table 26) to 86% (Table 28). However, at the same time, the producer's accuracy was reduced from 6% to 3%.

Table 28. Detection results on large areas.

| | predicted class | | | | | correct | |
|---|---|---|---|---|---|---|---|
| true class | grave mound | pitfall trap | charcoal kiln | background | sum | count | rate |
| grave mound | **172** | 0 | 2 | 25 | 199 | 172 | 86 % |
| pitfall trap | 5 | **254** | 2 | 42 | 303 | 254 | 84 % |
| charcoal kiln | 0 | 0 | **89** | 6 | 95 | 89 | 94 % |
| background | 10669 | 4215 | 4279 | 0 | 19163 | | |
| sum objects | | | | | **597** | | |
| **true positives** | | | | | | **515** | **86 %** |
| false negatives | | | | **73** | | | 12 % |
| false positives | | | | | 19163 | | 3210 % |
| wrong class | | | | | 9 | | 1,5 % |
| sum predicted | 10846 | 4469 | 4372 | | 19687 | | |
| correct | 172 | 254 | 89 | | 515 | | |
| producer's accuracy | 2 % | 6 % | 2 % | | 3 % | | |

### 8.8.1 Implementation details

In `simple-faster-rcnn-pytorch-master/config.py` (which is imported by `main.py`), the following changes were made.

Lines 12-13:

```
    workdir =
"/nr/samba/jodata10/pro/CultSearcher/usr/trier/detection_augment
-8_alt-subdiv/work",

    resultdir =
"/nr/samba/jodata10/pro/CultSearcher/usr/trier/detection_augment
-8_alt-subdiv/results",
```

Line 49:

```
    "model_file":
'/nr/samba/jo/pro/cultsearcher2018/usr/trier/jo1_src/sfrcnn/simp
```

```
le-faster-rcnn-pytorch-
master/checkpoints/fasterrcnn_11172341_0.8056864563757405_3-
classes_RA_test_augment-8_0001_alt-subdiv'
```

Commands to run detection on the test sets:

```
python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/lidar/oppland/lesja_2013_utm33/LAS/lesja_test
lesja_2013_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
brumunddal_2016_test_hele/LAS/brumunddal_2016_test_hele
brumunddal_2016_test_1

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
larvik_2017_test_hele/LAS/larvik_2017 larvik_2017_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
sarpsborg_2015_test_hele/LAS/sarpsborg_2015_test_hele
sarpsborg_2015_test_hele

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
steinkjer_2011_test_hele/LAS/steinkjer_2011_test_hele
steinkjer_2011_test_hele

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/gravhaug/
steinkjer_2017_test_hele/LAS/steinkjer_2017_test_hele
steinkjer_2017_test_hele

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/dovre_nordgudbrandsdal_2013_test_hele/LAS/dovre_2013
dovre_2013_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/lesja_vaga_test_hele/LAS/lesja_vaga_test_hele dovre_2017_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/lidar/oppland/dovre/LAS/folldal_2018_utm33
dovre_folldal_2018_test
```

```
python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/grimsdalen_2010_test_hele/LAS/grimsdalen_2010_test_hele
grimsdalen_2010_test_hele

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/nordfron_midtgudbrandsdal_2012_test_hele/LAS/nordfron_midtgudb
randsdal_2012_test_hele
nordfron_midtgudbrandsdal_2012_test_hele/

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/nordfron_nordgudbrandsdal_2013_test/LAS/nordfron_nordgudbrands
dal_2013_test nordfron_nordgudbrandsdal_2013_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/nor
dfron_2017_test nordfron_2017_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/nordfron_2018_test_hele/LAS/nordfron_2018_test_hele
nordfron_2018_test_hele

python -W ignore main.py --utm-zone 33
/nr/samba/jodata2/pro/cultsearcher/data/oppland/nordfron/LAS/ols
tappen_test nordfron_olstappen_2010_test

python -W ignore main.py --utm-zone 33
/nr/samba/jodata10/pro/CultSearcher/usr/trier/testdata/fangstgro
p/nordfron_venabu_2018_test_hele/LAS/nordfron_venabu_2018_test_h
ele nordfron_venabu_2018_test_hele
```

Command to compute detection statistics:

```
python detection_statistics_augment-8_alt-subdiv.py
```

## 8.9 Results from new archaeological mapping

The method was applied to the entire area (457 km$^2$) of Øvre Eiker municipality, Buskerud County, Norway. This municipality is covered by three non-overlapping ALS datasets (Table 7). More than 1000 previously unknown charcoal kiln locations were mapped (e.g., Figure 82, Figure 83). Manual inspection (e.g., Figure 83) indicated that few (if any) false predictions were made. Therefore, all the predicted charcoal kilns were inserted into the Askeladden database of cultural heritage in Norway. These charcoal kiln locations may be viewed in the portal https://kulturminnesok.no/ (Figure 84-.

E.g., the five charcoal kilns in Figure 83 may be seen in the centre of Figure 84. By clicking on one of them, a detailed view is opened (Figure 85). The comment (in Norwegian) “Funnet med CultSearcher”, i.e., found by CultSearcher, indicates that the charcoal kiln was detected by the automated method.

By panning the map view to cover all of Øvre Eiker municipality, in several overlapping map portions (Figure 87-Figure 97), all the charcoal kiln locations, plus any other cultural heritage locations in the Askeladden database, may be seen.

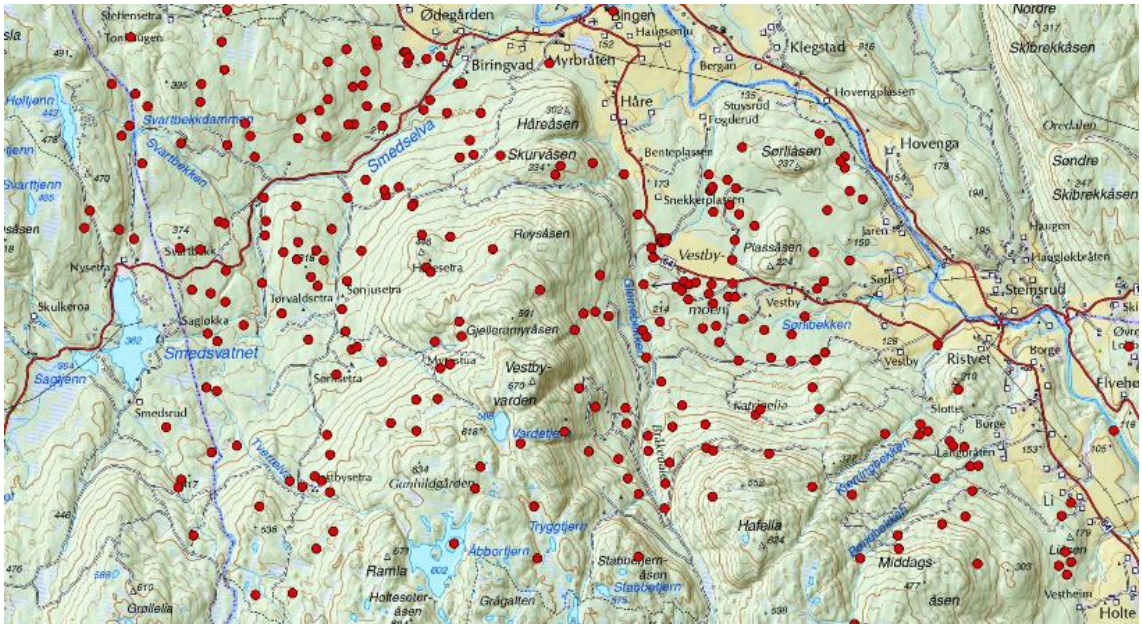



Figure 82. Detected charcoal kiln locations (red circles) for a forested area (pale green) south of Bingen in Øvre Eiker municipality.

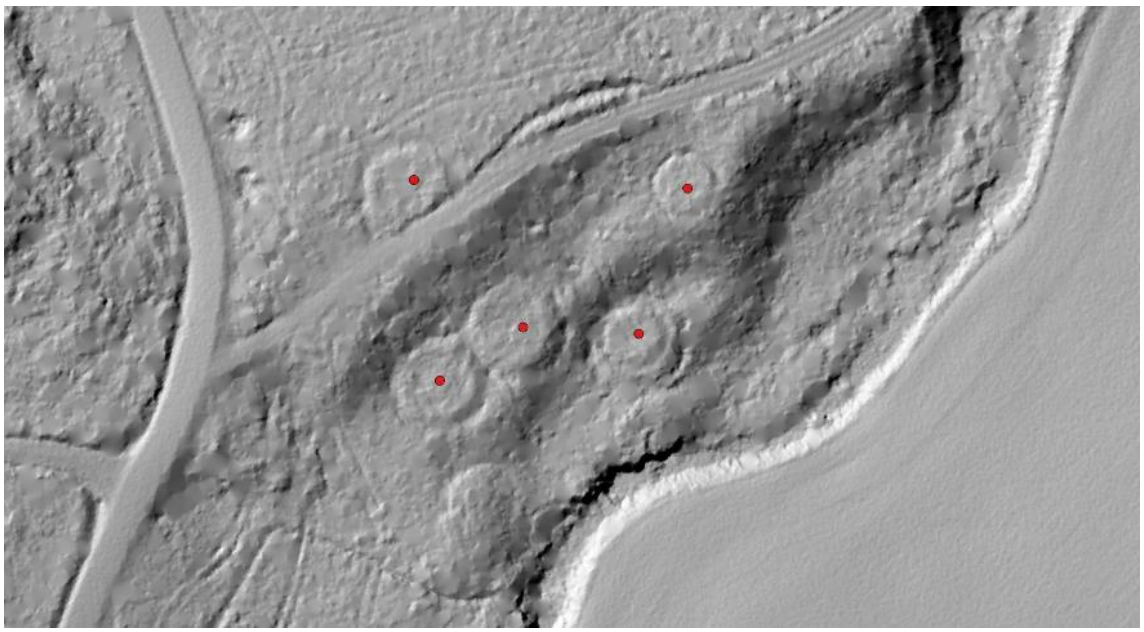


Figure 83. Five of the detected charcoal kilns, near Vestby, south of Bingen.

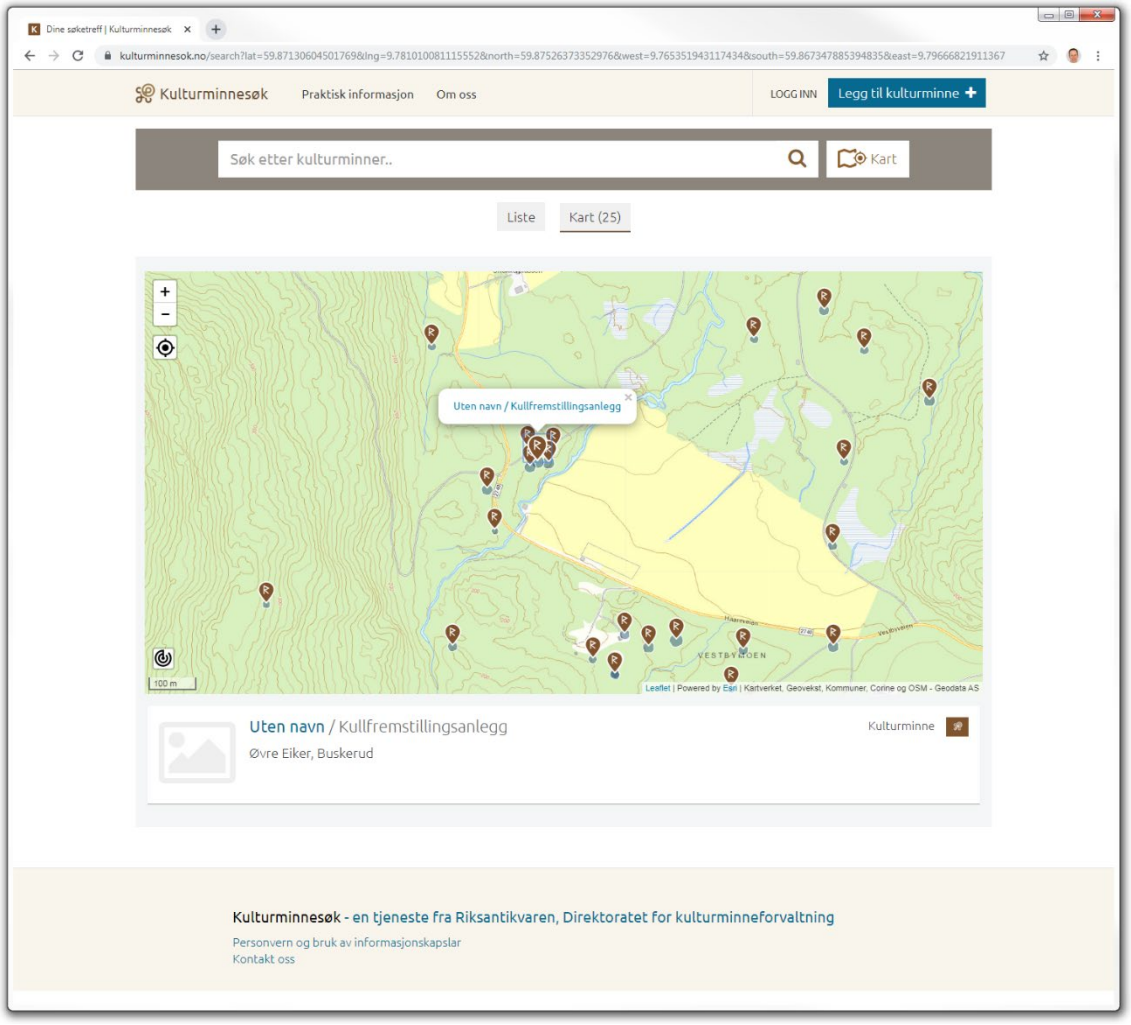



Figure 84. Cultural heritage locations near Vestby, Øvre Eiker.

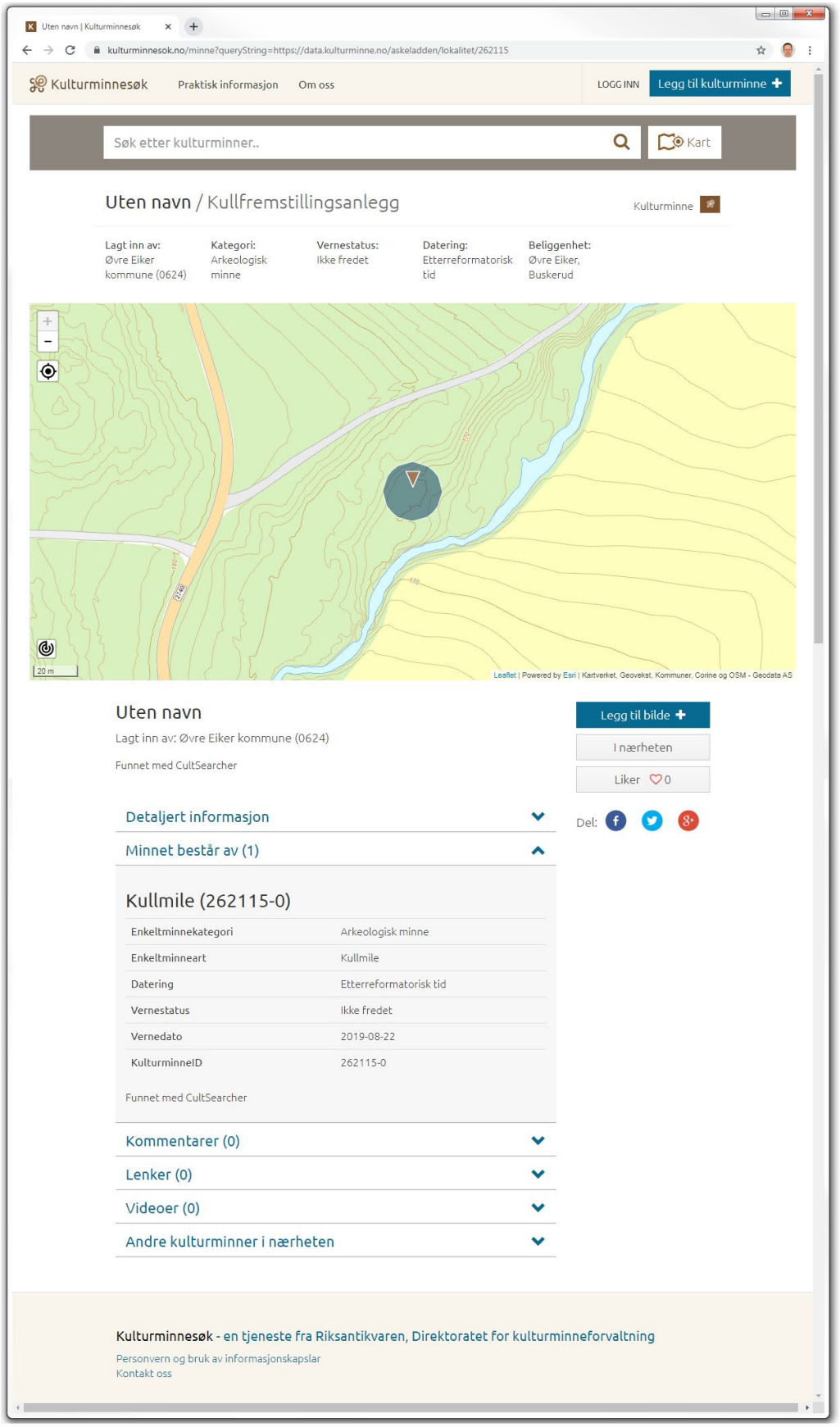



Figure 85. Details of database record of charcoal kiln.

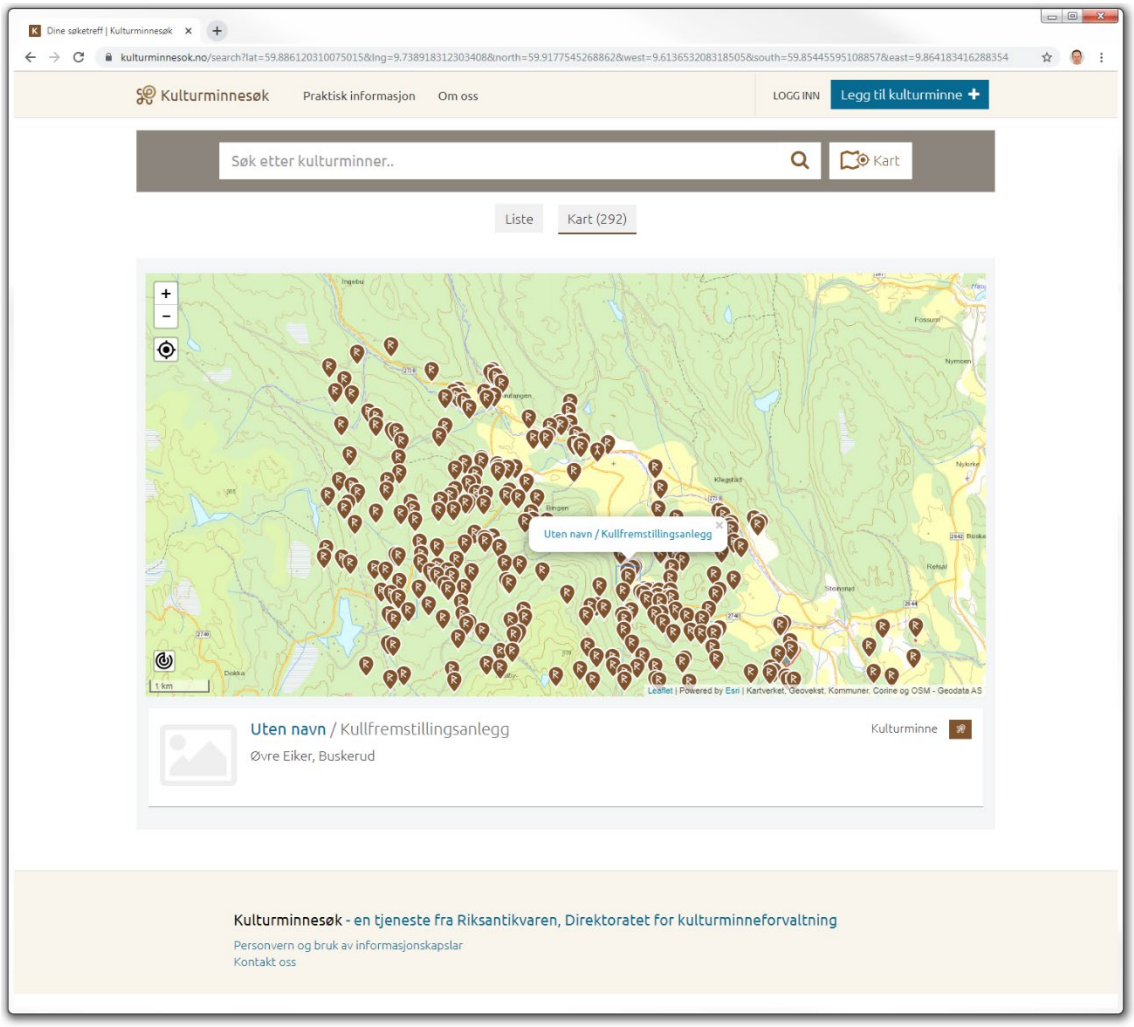



Figure 86. By zooming out from the detailed map view (Figure 84), the northwestern corner of Øvre Eiker municipality is displayed, with cultural heritage locations (brown symbols).

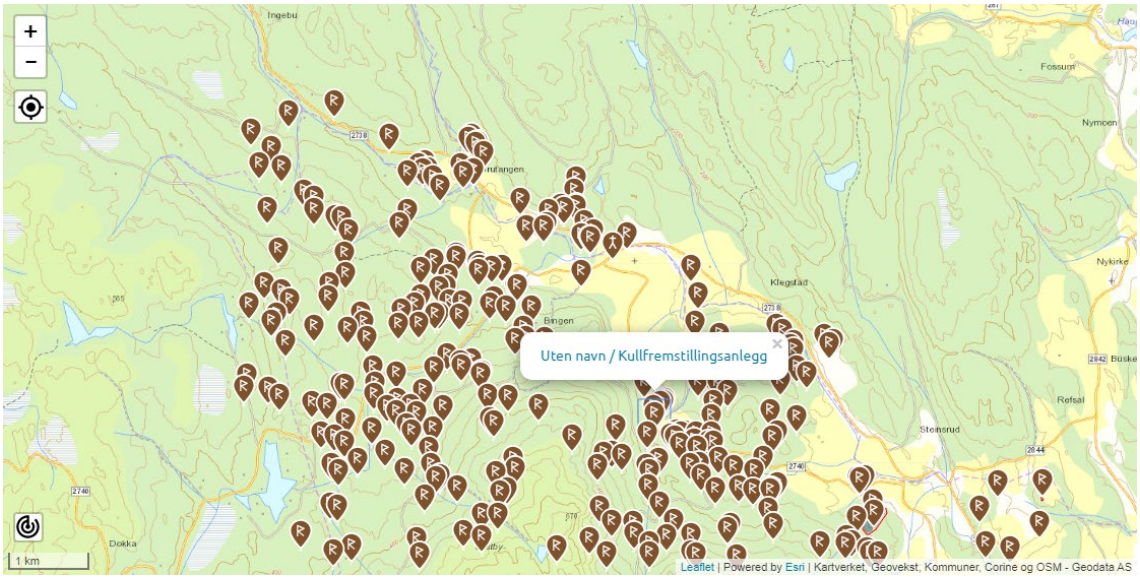



Figure 87. The map portion of Figure 86, i.e., part 1 of Øvre Eiker municipality..

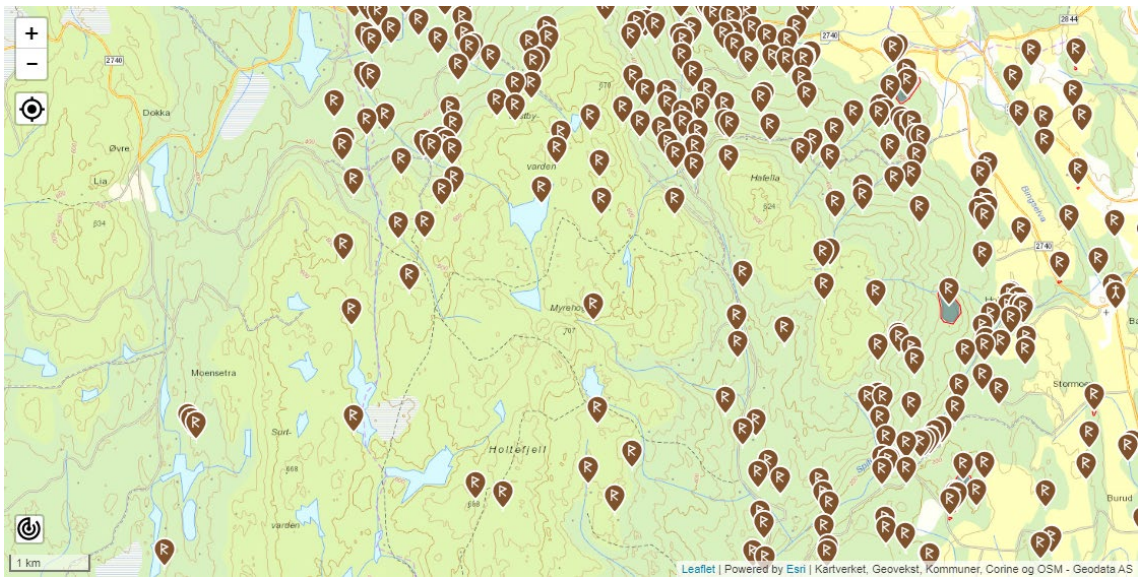



Figure 88. Part 2 of Øvre Eiker municipality, i.e., south of part 1.

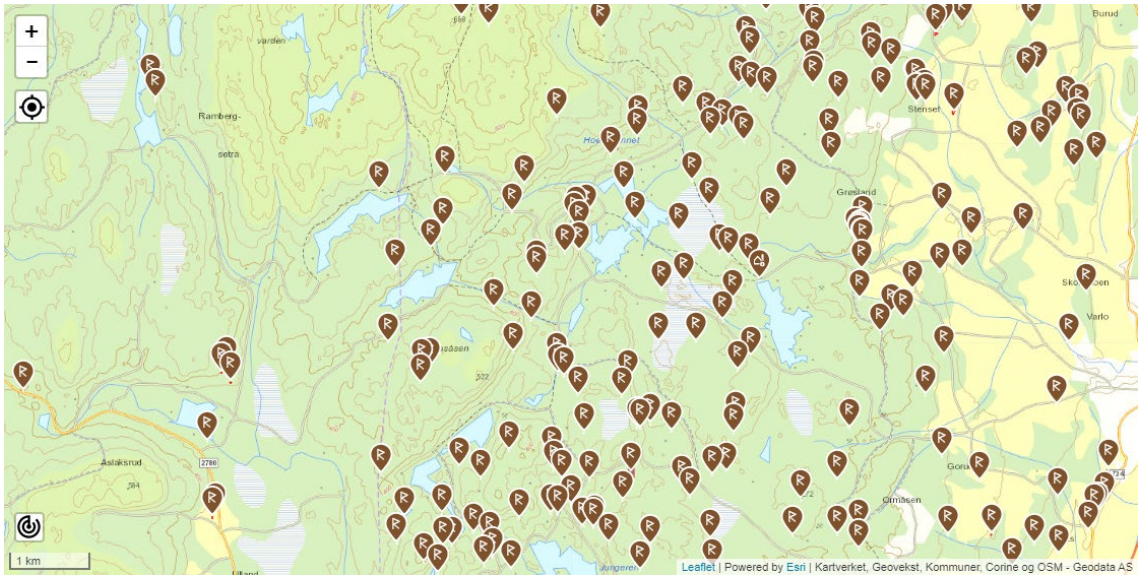



Figure 89. Part 3 of Øvre Eiker municipality, i.e., south of part 2.

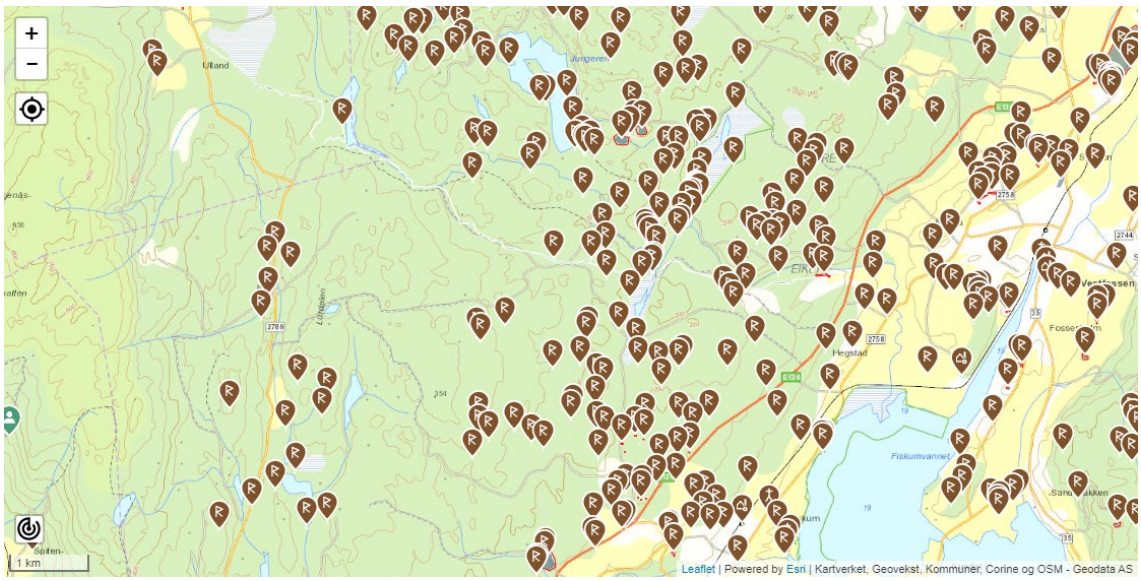



Figure 90. Part 4 of Øvre Eiker municipality, i.e., south of part 3.

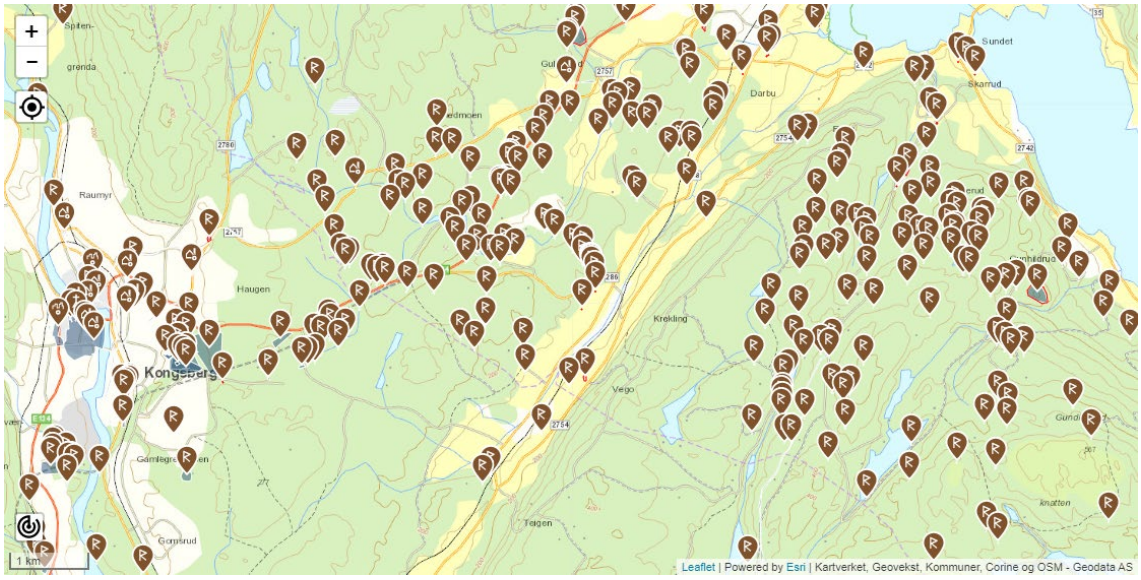



Figure 91. Part 5 of Øvre Eiker municipality, i.e., south of part 4.


Figure 92. Part 6 of Øvre Eiker municipality, i.e., southeast of part 5.

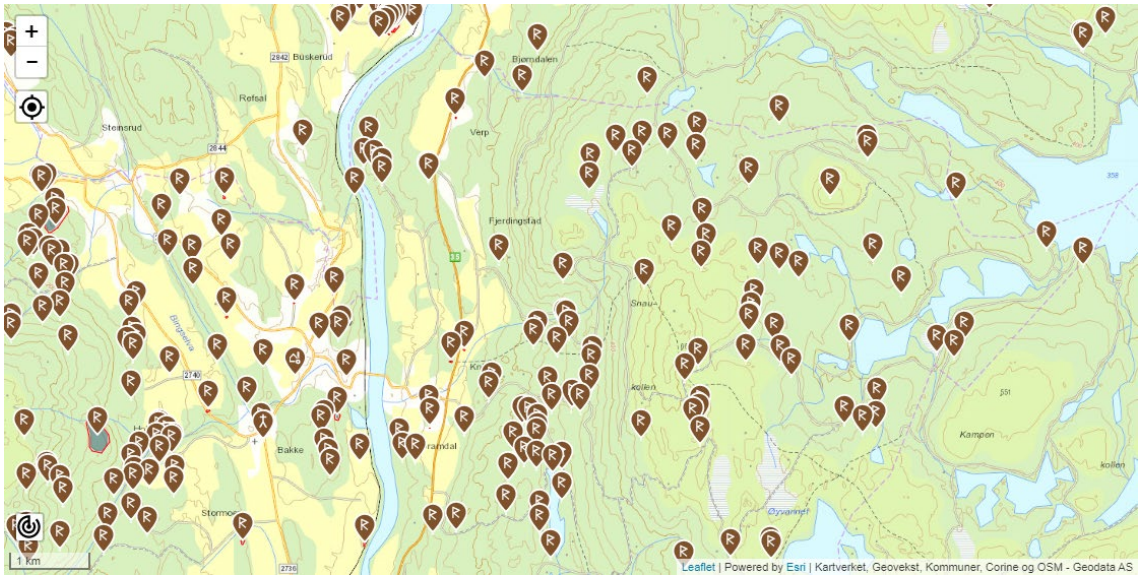



Figure 93. Part 7 of Øvre Eiker municipality, i.e., east of part 1 and part 2.

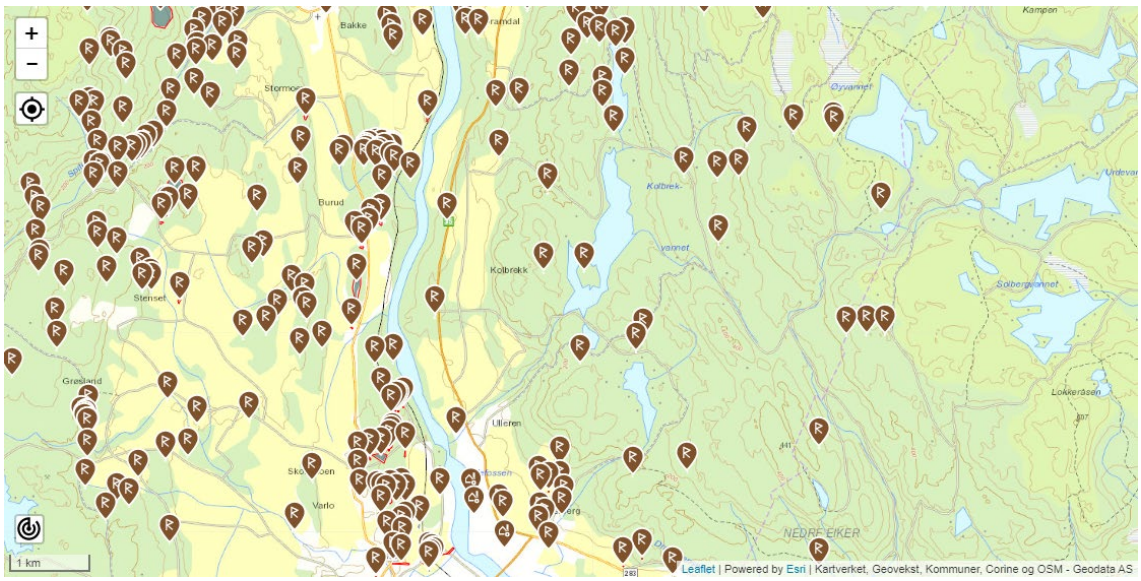



Figure 94. Part 8 of Øvre Eiker municipality, i.e., south of part 7.

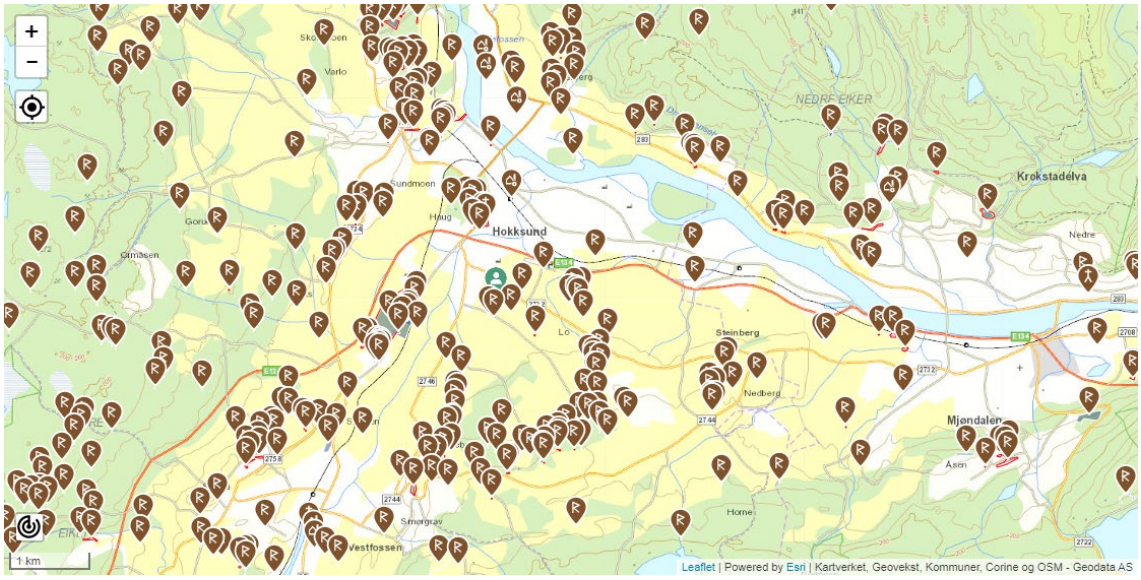



Figure 95. Part 9 of Øvre Eiker municipality, i.e., south of part 8.

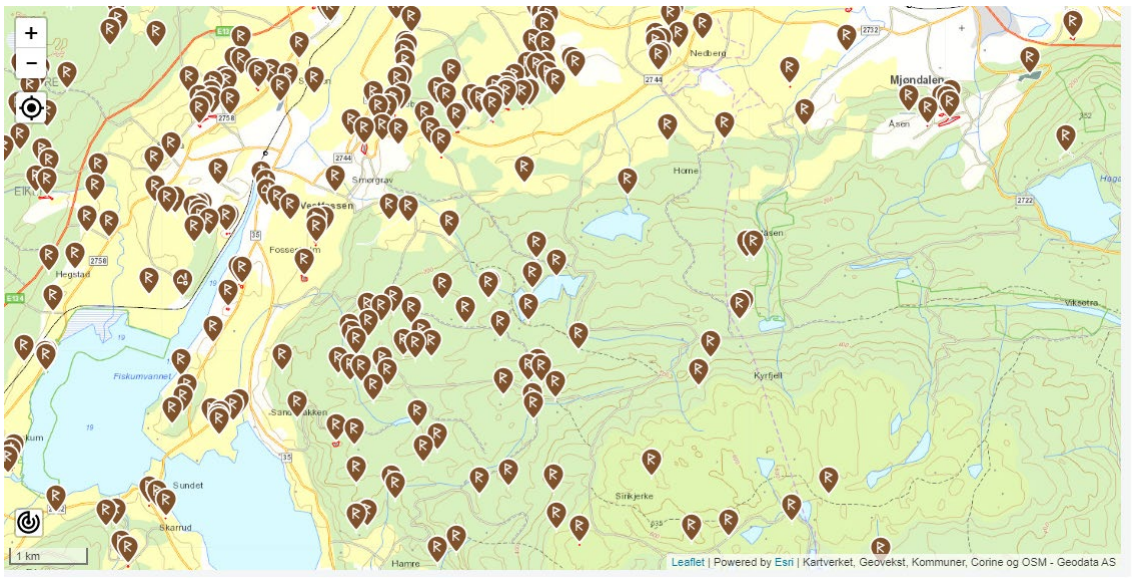



Figure 96. Part 10 of Øvre Eiker municipality, i.e., south of part 9.


Figure 97. Part 11 of Øvre Eiker municipality, i.e., south of part 10, east of part 5 and northeast of part 6.

# 9 Discussion and conclusions

The best classification performance was 86% correct classification (consumer's accuracy), i.e., the how many of the true cultural heritage objects were correctly predicted by the method. This was obtained on a test set of labelled lidar data not seen during training. At the same time, the producer's accuracy was 3%, i.e., how many of the objects predicted by the method were in fact true cultural heritage objects. Thus, the main potential for improvement is in reducing the large number of false predictions, i.e., increasing the producer's accuracy. This should be the focus for future improvements of the detection method.

The method has been used on a number of ALS datasets covering a variety of landscape types, including forest, mountain, urban, rural, agricultural and coastal areas. Although a detailed quantification of detection performance has not yet been performed, some trends were observed. The method performed better on charcoal kilns than on the other object types. In the inland, the method performed well on pitfall traps. This included many areas that are lacking detailed cultural heritage mapping. An unexpected bonus was that charcoal pits / tar pits were detected, albeit as pitfall traps. For grave mounds, the method was less successful. Confusion between natural knolls and grave mounds was the main problem. Still, the method may be useful by giving an overview of locations in the landscape with structures resembling grave mounds. These could then be checked visually by experienced archaeologist, who could spot which locations that need to be checked by field visits.

There are some recent projects that involve citizen volunteers to help identify which automatically detected structures are true archaeological remains. In the Chilterns in England (Morrison and Pevelier, 2019; https://chilternsbeacons.org/wp/), citizens use an internet portal to view different ALS visualisations of an area to identify and map archaeology. In the Veluwe area in the centre of the Netherlands (Lambers *et al.*, 2019; https://www.zooniverse.org/projects/evakap/heritage-quest), an internet portal is also used. Participants are asked to mark every potential barrow, charcoal kiln and Celtic field within a 300 m by 300 m subimage. Each individual image is checked by at least eight different users.

Our method is based on transfer learning, but in a setting that may not be optimal. We used a deep neural network that is pre-trained on natural scene images, followed by training on ALS visualisations with labelled cultural heritage remains. As the two types of image are quite different, there is a potential for improvement by pre-training the deep neural network on a large image set that is more similar to the ALS visualisations that we used.

Several researchers are suggesting similar approaches:

1. Deep clustering. Unsupervised learning of visual features is performed on remote sensing images. Dzeroski and Kokalj (2019) of Slovenia presents a project that will address this, starting in 2020. Two application areas are included:

a. archaeology and

b. land cover classification.

2. Insert synthetic objects of interest into images with ‘background’ terrain. Kücükdemirci and Sarris (2019) presents a project which uses pre-training of a U-net on synthetic images, for segmentation of archaeological structures in geophysical images. E.g., circles and straight lines are inserted into images of ‘background’ terrain.

Another issue related to object detection is that in the majority of landscapes, the absence of objects is much more frequent than the presence. Kramer et al. (2019) observes that the RetinaNet addresses this imbalance of foreground versus background.

An issue that is observed at terrain discontinuities, e.g., a cliff, is that the local relief model visualisation may hide archaeological objects that are close to the terrain discontinuity. A possible solution could be to use another ALS visualisation, e.g. openness.

Landauer and Hesse (2019) obtain very low false positive rates on a set of 29 000 labelled, possible charcoal kilns, with 95% detection rate. The labels ranged from 0 = ‘certainly not’ to 4 = ‘definetly yes’. For each 40 m by 40 m image of a possible charcoal kiln, the final label is the average of the labels provided by several human users. Of the 30 false positives (i.e., images labelled with 0, but detected as charcoal kiln by the deep neural network) 15 were in fact charcoal kilns and thus wrongly labelled 0.

# 10 Conference presentations

The project has been presented at three conferences in 2019:

- Nordic Remote Sensing Conference, 17-19 September 2019, Århus, Denmark.
- 24th Conference on Cultural Heritage and New Technologies, November 4-6 2019, Vienna, Austria.
- Artificial Intelligence, Machine Learning and Deep Learning in Archaeology, 7-8 November 2019, Rome, Italy.

## 10.1 Automated mapping of cultural heritage in Norway from airborne lidar data using Faster RCNN

*Øivind Due Trier*

This paper was presented at the Nordic Remote Sensing Conference, 17-19 September 2019, Århus, Denmark.

### 10.1.1 Abstract

We present a new method for automated mapping of historic monuments such as grave mounds, pitfall traps and charcoal kilns. The method is based on a region-proposal convolutional neural network called “simple faster R-CNN”. The network was pre-trained on a large database of natural scene images. Each image had annotations in the form of bounding boxes with associated class labels. Then the network was trained on images derived from airborne lidar data.

The lidar point cloud data was converted to a digital terrain model (DTM) by keeping all points that were labelled as ‘ground’. The DTM was then converted to a simplified local relief model by subtracting a smoothed version of the DTM. The local relief model enhances local detail in the DTM while suppressing the general landscape topography. Thus, cultural heritage remains such as grave mounds, pitfall traps and charcoal kilns are often visible.

Each geographic area was divided into disjoint areas for training, validation and testing. Training, validation and test images of sizes 150 m × 150 m were extracted from the local relief model data. Each image contained one or more cultural heritage objects clearly visible.

For the test images, the overall correct classification rate was 83%, and for the specific classes: grave mound 81%, pitfall trap 78% and charcoal platform 95%. 16% of the true cultural heritage objects were missed by the method. 1% of the cultural heritage objects were detected with wrong class. 21% of the objects that the method predicted as being cultural heritage were in fact not.

The new method was implemented in an automated processing chain that will be used by the Directorate for Cultural Heritage in Norway (Riksantikvaren) to improve the

national cultural heritage mapping. The main focus is on grave mounds and pitfall traps, since these are protected by Norwegian law.

### 10.1.2 Types of cultural heritage

The focus was on automated detection of three types of cultural heritage that occur frequently in many types of Norwegian landscape:

- Grave mounds (Figure 98Figure 99) from the Viking Age
- Pitfall traps (Figure 100) from deer hunting systems
- Charcoal kilns (Figure 101)

Figure 98. Grave mounds in Norway's largest Viking Age grave field at Vang, Oppdal municipality, Trøndelag County.

Figure 99. One of the larger grave mounds at Vang, Oppdal, Trøndelag.

Figure 100. Pitfall trap, Oppland County. Photo: Lars Holger Pilø, Oppland County Administration.

Figure 101. Charcoal kiln, Lesja, Oppland County.

### 10.1.3 Lidar – light detection and ranging

Airborne lidar data provides elevation measurements, both from the ground surface and vegetation such as trees. Since each (x, y, z) point in the lidar data has been labelled as ‘ground’ (i.e., terrain) or ‘other’ (including buildings, vegetation, etc.), the non-terrain points may be removed, and a very detailed digital terrain model (DTM) may be obtained (Figure 102).

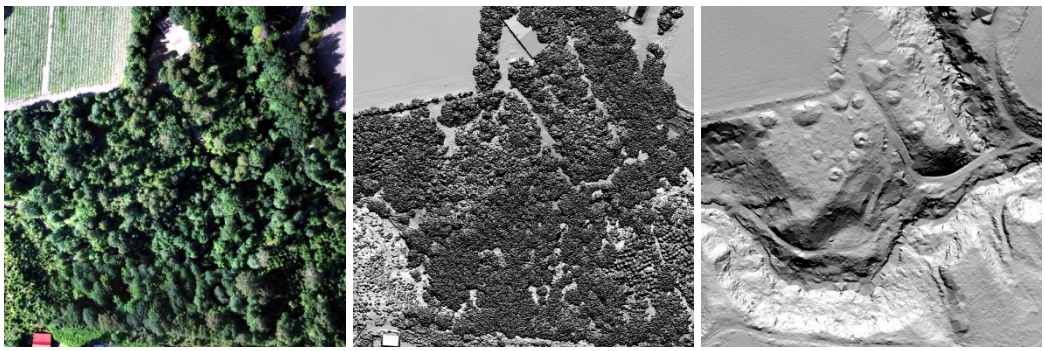


Figure 102. A forested area in Larvik municipality, Vestfold County. Left: air photo. Middle: digital surface model from airborne lidar data, first hits. Right: digital terrain model from airborne lidar data, ground hits.

### 10.1.4 Background

Existing cultural heritage mapping is incomplete. Some areas are mapped well, while some areas have only chance discoveries. The positional accuracy may be bad in old mapping.

There is now a major effort to create a new national elevation model for Norway. Accurate data is collected in the form of airborne laser scanning below the timber line. Automatic image matching is used above the timber line.

The Norwegian Computing Center has, over a number of years, developed semi-automatic detection methods for some types of cultural heritage objects. The Directorate for Cultural Heritage in Norway want to run the detection methods in-house

### 10.1.5 Challenges

The following challenges were identified:

1. Develop an automated processing chain
2. Reduce processing time
3. Reduce the number of false positives and false negatives
4. Develop detection methods that may be applied on all Norwegian landscapes

### 10.1.6 Recent developments

- Region proposal combined with convolutional neural network (CNN) classification: R-CNN (Girshick et al., CVPR 2014)
- Fast R-CNN (Girshick et al., ICCV 2015)
- Faster R-CNN (Ren et al., NIPS 2015; T-PAMI 2017)
- Mask R-CNN (He et al., ICCV 2017)
- Detectron (Girshick et al., 2018)

#### 10.1.6.1 Visualisation of airborne lidar data

In order to use the DTM data for automated object detection, a suitable visualization method had to be used. The local relief model is able to capture local detail while suppressing the general terrain elevation ().

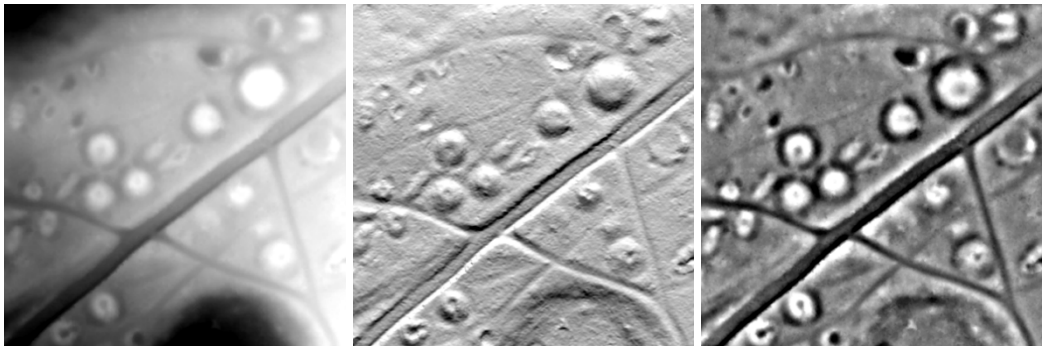


Figure 103. Lidar data from Bøkeskogen, Larvik municipality, Vestfold County. Several grave mounds are visible. Left: terrain elevation. Middle: hillshade. Right: local relief model.

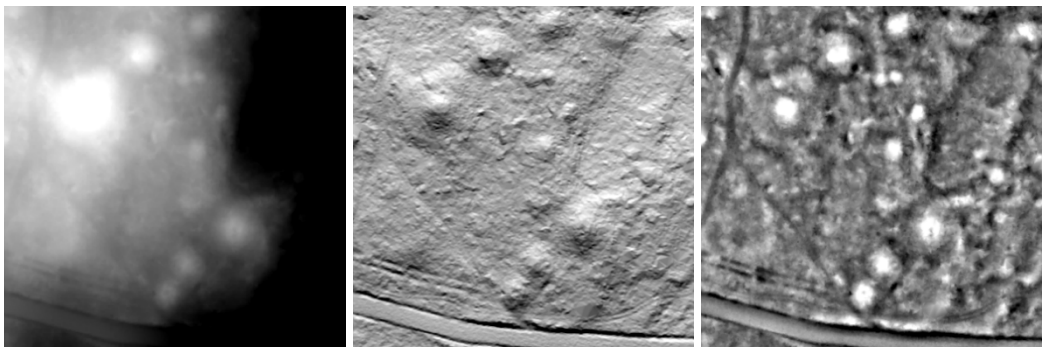


Figure 104. Lidar data from Omsland, Larvik municipality. Several grave mounds are visible. Left: terrain elevation. Middle: hillshade. Right: local relief model.

Figure 105. Lidar data from Nord-Fron municipality, Oppland County. A deer hunting system with pitfall traps is visible. Left: terrain elevation. Middle: hillshade. Right: local relief model.

### 10.1.7 Alternatives for R-CNN

We used: simple faster R-CNN:
https://github.com/chenyuntc/simple-faster-rcnn-pytorch.

Detectron:
https://github.com/facebookresearch/Detectron

Mask R-CNN (included in Detectron)

Python faster R-CNN:
py-faster-rcnn has been deprecated. They advice to use Detectron, which includes Mask R-CNN.

### 10.1.8 Modifications to code

List of class labels in python code must be updated to agree with the class labels used in the annotations of the image database:

- gravhaug (grave mound)
- fangstgrop (pitfall trap)
- kullmile (charcoal kiln / charcoal burning platform)

- any confusion classes, e.g., natural mound

Code crashed if no objects were found inside a 600 × 600 pixels subimage. Solution: add IF-tests

### 10.1.9 Examples

By running the code in display mode, detection results may be viewed to verify if the method is able to detect grave mounds (), pitfall traps () and charcoal kilns ().

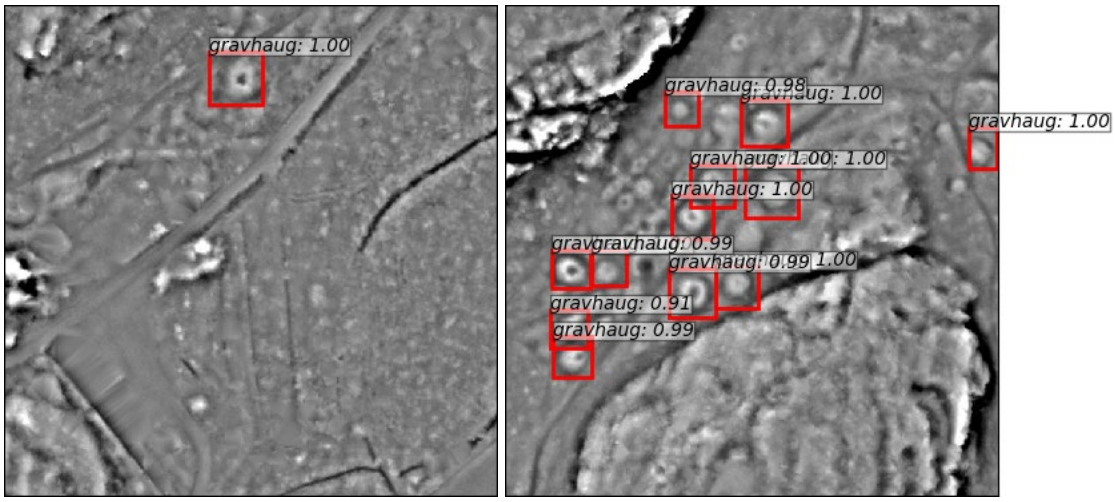



Figure 106. Examples of detected grave mounds (in Norwegian: gravhaug), Larvik municipality, Vestfold County.

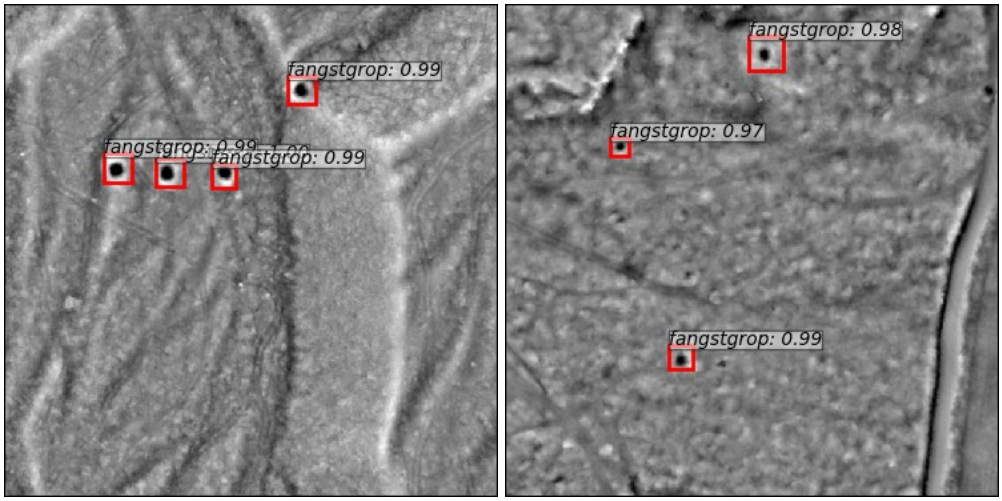



Figure 107. Examples of detected pitfall traps (in Norwegian: fangstgrop), Nord-Fron municipality, Oppland County.

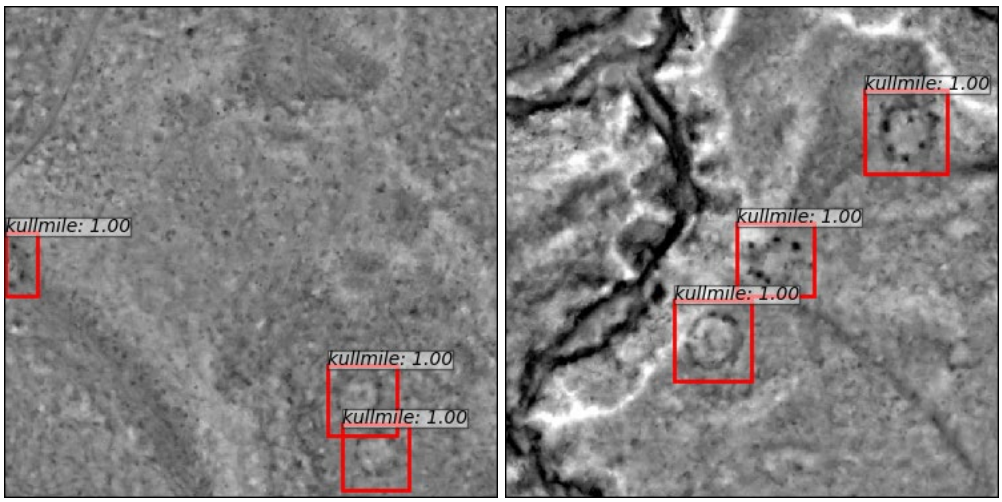



Figure 108. Examples of detected charcoal kilns (in Norwegain: kullmiler), Lesja municipality, Oppland County.

### 10.1.10 Training data

The training set was used to optimize the neural network parameters. The validation set was used to select the best set of neural network parameters encountered so far.

Training set: 2180 images – 2180 unique objects

- 11262 grave mounds
- 1840 pitfall traps
- 2170 charcoal platforms

Validation set: 1142 images – 1142 unique objects

- 685 grave mounds
- 2980 pitfall traps
- 345 charcoal platforms

### 10.1.11 Results

The detection results (Table 29) were obtained by running the method on 734 test images not seen during training. Each test image contained one or more cultural heritage objects, and 734 unique cultural heritage objects in total. So, Images overlap other images if they contain multiple cultural heritage objects. Thus, there were 2160 cultural heritage objects including duplicates.

82% of cultural heritage objects are detected with correct class. Less than 1% of the cultural heritage objects are detected but with wrong class. 18% of cultural heritage objects are not detected 17% of the predicted objects are false (background).

There were no images without cultural heritage objects. Thus, the reported number of false positives (17%) may be a too optimistic estimate for a large landscape.

Table 29. Detection results on 734 small test images of 150 m by 150 m, not seen during training, and each containing at least one cultural heritage object.

| | predicted object | | | | | correct detections | |
|---|---|---|---|---|---|---|---|
| true object | grave mound | pitfall trap | charcoal platform | background terrain | sum | count | percent |
| grave mound | 512 | 3 | 0 | 189 | 704 | 512 | 72,73 % |
| pitfall trap | 0 | 1086 | 0 | 188 | 1274 | 1086 | 85,24 % |
| charcoal platform | 0 | 0 | 168 | 14 | 182 | 168 | 92,31 % |
| background terrain | 146 | 177 | 33 | 0 | 356 | | |
| sum true objects | | | | | **2160** | | |
| **sum correct detections** | | | | | | **1766** | **81,76 %** |
| **wrong class** | | | | | **3** | | **0,14 %** |
| missing objects | | | | **391** | | | **18,10 %** |
| sum | | | | | | | 100,00 % |
| all detections | | | | | 2125 | | |
| false positives | | | | | **356** | | **16,75 %** |

The method was then used on all of Øvre Eiker municipality, an area with few recorded charcoal kilns; thus, no ground truth existed. This is the normal situation for the practical use of the method, in order to discover previously unknown cultural heritage locations. More than 1000 charcoal locations were predicted by the method (). A quick visual inspection (e.g., ) confirmed that the large majority, if not all, of the predicted charcoal locations were true. Thus, they were included into the Askeladden database of all cultural heritage locations in Norway.

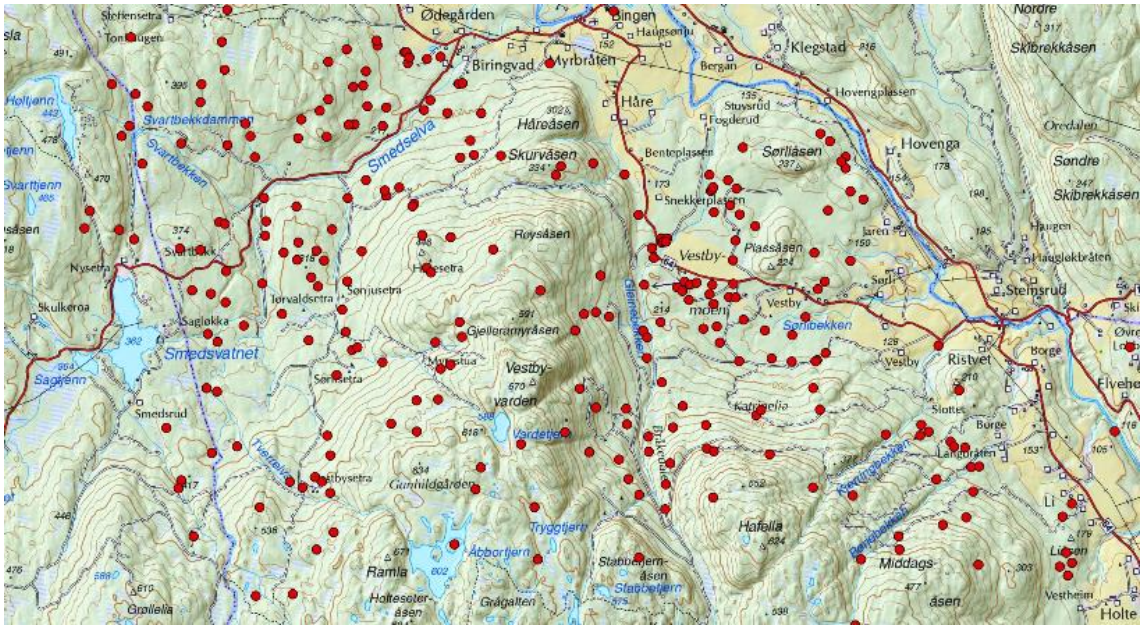



Figure 109. Some of the predicted charcoal kilns (red circles) in forested areas (pale green) in Øvre Eiker municipality.

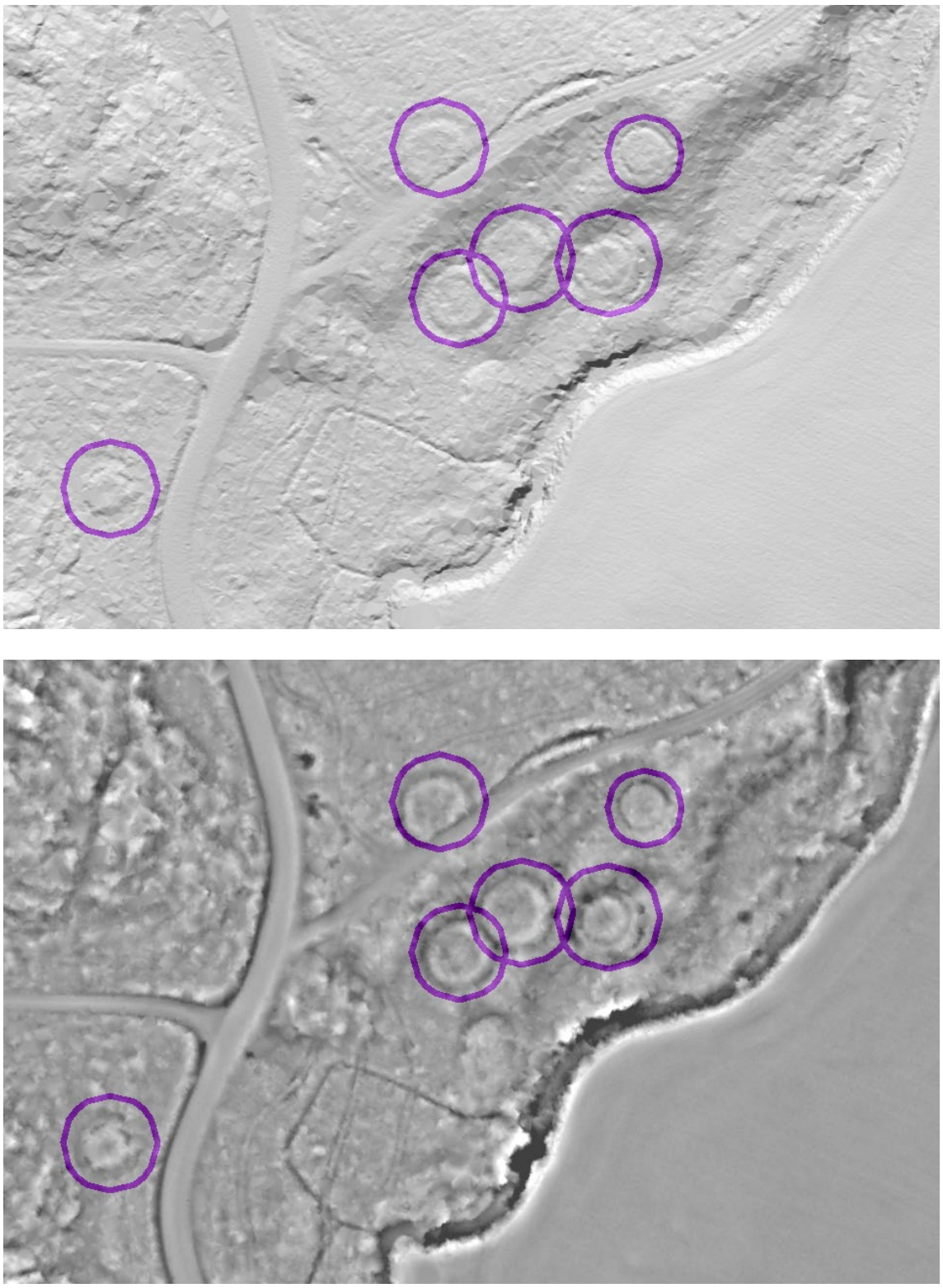


Figure 110. Visual inspection of six predicted charcoal kilns (purple circles) in Øvre Eiker municipality. Top: hillshade visualization of DTM. Bottom: Local relief visualization of DTM.

### 10.1.12 Future work

In order to estimate detection results for operational use, we will run automatic detection on entire LAS files and not only on small image portions which contain at least one cultural heritage object.

In order to avoid missing cultural heritage at image borders, detection must be run on overlapping images. Then, duplicate predictions must be eliminated.

The ground truth data must be valid for all the lidar data included in the test set. The predicted cultural heritage objects will be compared with the ground truth data. We expect to see more false positives. However, we expect the method to be able to detect roughly the same percentage of the true cultural heritage objects.

One possible workaround to reduce the number of false positives may be to add confusion classes. This may be done by running the detection method on training and validation areas. Then, false positives may be labelled with new class names:

- Natural mound
- Natural pit
- Natural platform

Also, it may be necessary to check if any false positives are actually true positives.

With the confusion class objects added to the training and validation sets, training will be re-run.


Figure 111. While on fieldwork in Øystre Slidre municipality, Oppland County, two archaeologists spotted this road sign at Lidar church.


Figure 112. Lidar church is located near Skammestein in Øystre Slidre municipality, between Fagernes and Beitostølen.

## 10.2 Detection of cultural heritage in airborne laser scanning data using Faster RCNN. Results on Norwegian data

*Øivind Due Trier*

This paper was presented at the 24th Conference on Cultural Heritage and New Technologies, 4-6 November 2019, Vienna, Austria.

**Keywords:** grave mounds; hunting systems; charcoal kilns; automated detection; lidar

### 10.2.1 Introduction

The existing cultural heritage mapping in Norway is incomplete. Some selected areas are mapped well, while the majority of areas only contain chance discoveries, often with bad positional accuracy.

Automated methods for detecting some types of cultural heritage objects from airborne laser scanning (ALS) data have previously been developed. These have contributed to increasing the number of areas that are mapped well. However, the methods have a number of issues that have prevented them from being used systematically on all available ALS datasets.

All of Norway will soon be covered by airborne laser scanning data for the purpose of creating a new national elevation model. The Directorate for Cultural Heritage in Norwaywants to use this opportunity to obtain a more complete and accurate mapping of cultural heritage in the landscape. The focus is on Iron Age grave mounds and deer hunting systems.

The following challenges were identified: (1) develop an automated processing chain, (2) reduce processing time, (3) reduce the number of false positives and false negatives, and (4) develop detection methods that may be applied on all Norwegian landscapes.

A recent development in deep neural networks for object detection in natural images is the region-proposing convolutional neural network (R-CNN; Girshick *et al*., 2014), which may also be used for cultural heritage detection in ALS data. Verschoof-van der Vaart and Lambers (2019) use Faster R-CNN (Ren *et al*., 2017) to detect prehistoric barrows and Celtic fields in ALS data from the Netherlands.

He *et al*. (2017) extend Faster R-CNN into Mask R-CNN by providing, for each detected object, an object mask in addition to the bounding box provided by Faster R-CNN.

### 10.2.2 Data

ALS point cloud data was downloaded from http://hoydedata.no. This internet site provides free access to all ALS data in Norway.

Vector maps of known locations of grave mounds and pitfall traps were provided as ESRI shape files by the Directorate for Cultural Heritage in Norway. Vector maps of charcoal kiln locations were provided by Oppland County Administration.

The data were split into three parts, named ‘training’, ‘validation’, and ‘test’. The neural network parameters were learned from the training data iteratively by minimising a loss function. The validation data were used to select the best set of neural network parameters. The test data were then used to estimate detection performance on data not seen during training.

### 10.2.3 Methods

#### 10.2.3.1 Preprocessing

The ALS point cloud data were converted to a digital terrain model (DTM) with 0.25 m pixel spacing. The DTM was converted to a simplified local relief model (LRM) by subtracting a smoothed version of the DTM. The LRM enhances local elevation differences while suppressing the general landscape topography. Thus, cultural heritage objects including grave mounds, pitfall traps and charcoal kilns may be visible.

For each cultural heritage object in the vector data, a 150 m × 150 m image was extracted from the LRM. The object’s position within the subimage was selected at random. This was done in order to prevent the deep neural network from always predicting the object in the image centre. All cultural heritage objects within the subimage were included in the image annotation. Thus, each image contained one or more cultural heritage objects clearly visible.

#### 10.2.3.2 Detection

For detection, the python code library *simple faster R-CNN* was downloaded from https://github.com/chenyuntc/simple-faster-rcnn-pytorch. For each detected object the R-CNN predicts a bounding box, a class label and a score value in the range 0.0 – 1.0. A few modifications had to be done. (1) The list of class labels was changed to match the class labels used in the image annotations. (2) The downloaded code crashed if there were no detected objects within an image. Thus, if-tests had to be added.

When these changes were made, the python code predicted the location and sizes of grave mounds (Figure 113), pitfall traps (Figure 114) and charcoal kilns (Figure 115) in LRM images of size 600 × 600 pixels.

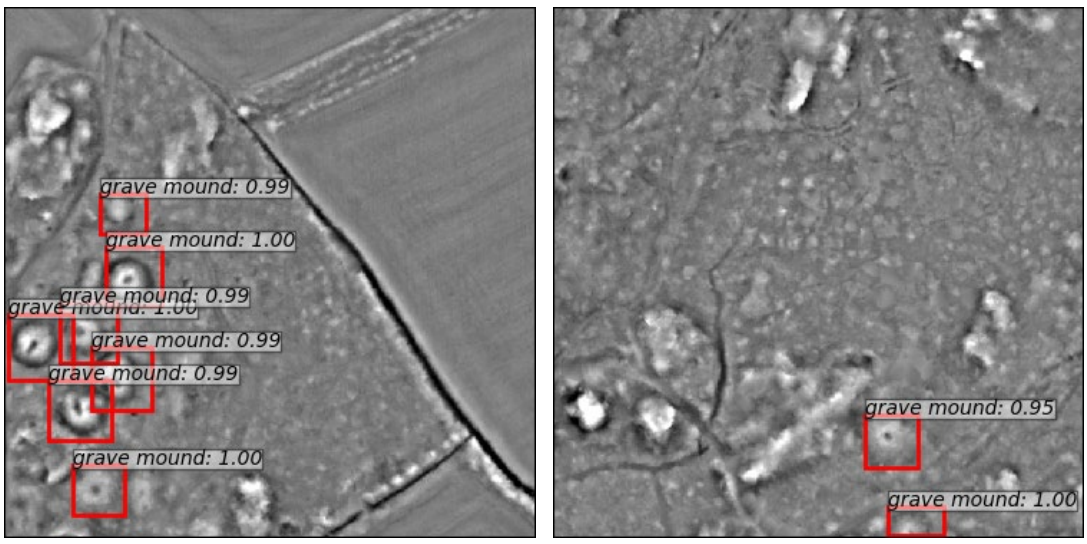



Figure 113. Predicted grave mound locations.

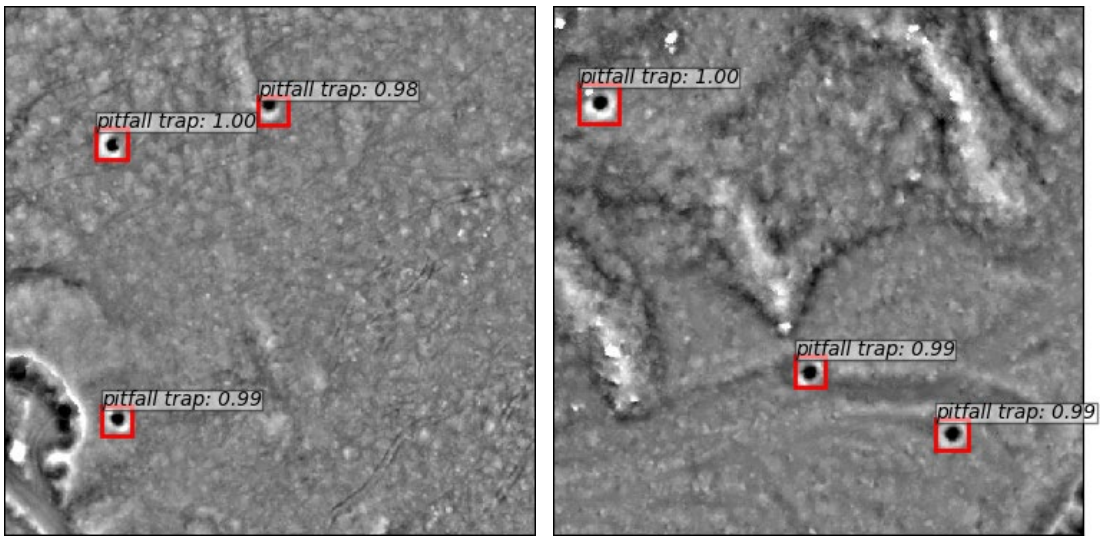



Figure 114. Predicted pitfall trap locations.

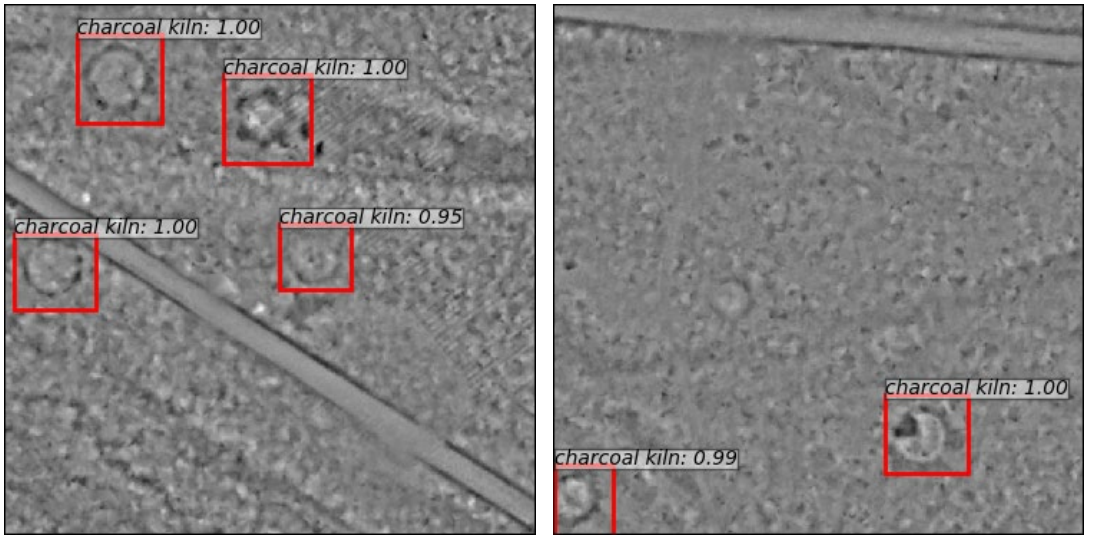



Figure 115. Predicted charcoal kiln locations.

#### 10.2.3.3 Processing chain

The preprocessing and detection methods were integrated into a python script that may be called from QGIS or started from the Linux command line. The input is a collection of LAS files, and the output is two ESRI shape files for each object type; centre points in one file and object outlines in another file. Each object outline is obtained by converting the predicted bounding box to a circle.

### 10.2.4 Results

By running on the test images, the overall correct classification rate was 83%, and for the specific classes, grave mound 81%, pitfall trap 78% and charcoal kiln 95%. 16% of the true cultural heritage objects were missed by the method, while 1% was detected with wrong class. 21% of the objects that the method predicted as being cultural heritage were in fact not. However, the latter figure may be an optimistic estimate of the amount of false positives that the method may provide. All the test images contained at least one cultural heritage object. In operational use, there may be large areas, within

an ALS dataset, with no cultural heritage objects visible in the data. Thus, the potential for false positives is much larger. Evaluation of the detection and classification performance in such a setting will be done in the near future.

## 10.3 Automated detection of grave mounds, deer hunting systems and charcoal burning platforms from airborne lidar data using faster-RCNN

*Øivind Due Trier and Kristian Løseth*

This paper was presented at the conference: Artificial Intelligence, Machine Learning and Deep Learning in Archaeology, 7-8 November 2019, Rome, Italy.

We present a new method for automated mapping of historic monuments such as grave mounds, pitfall traps and charcoal kilns. The method is based on a region-proposal convolutional neural network called “simple faster R-CNN”. The network was pre-trained on a large database of natural scene images. Each image had annotations in the form of bounding boxes with associated class labels. Then the network was trained on images derived from airborne lidar data.

The lidar point cloud data was converted to a digital terrain model (DTM) by keeping all points that were labelled as ‘ground’. The DTM was then converted to a simplified local relief model by subtracting a smoothed version of the DTM. The local relief model enhances local detail in the DTM while suppressing the general landscape topography. Thus, cultural heritage remains such as grave mounds, pitfall traps and charcoal kilns are often visible.

Each geographic area was divided into disjoint areas for training, validation and testing. Training, validation and test images of sizes 150 m × 150 m were extracted from the local relief model data. Each image contained one or more cultural heritage objects clearly visible.

For the test images, the overall correct classification rate was 83%, and for the specific classes: grave mound 81%, pitfall trap 78% and charcoal platform 95%. 16% of the true cultural heritage objects were missed by the method, while 1% were detected with wrong class. 21% of the objects that the method predicted as being cultural heritage were in fact not.

# 11 Newspaper story

The below newspaper story was published on 16 December 2019, and appeared in the printed version of Aftenposten on 17 December 2019, pages 24-25.

https://www.aftenposten.no/viten/i/kJPkeB/kunstig-intelligens-finner-skjulte-kulturminner

# Acknowledgements

This research was financed by the Directorate for Cultural Heritage in Norway (Riksantikvaren).

# References


Dzeroski, S. and Kokalj, Z. (2019). Machine learning, remote sensing and archaeology: tasks, tools, resources and needs. Presented at: *Artificial Intelligence, Machine Learning and Deep Learning in Archaeology*, 7-8 November 2019, Rome, Italy.

Girshick, R., Donahue, J., Darrell, T. and Malik, J., (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, Ohio, USA, 23-28 June 2014, pp. 580-587. DOI: 10.1109/CVPR.2014.81

Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K. (2018). *Detectron*. https://github.com/facebookresearch/detectron

He, K., Gkioxari, G., Dollar, P. and Girshick, R. (2017). Mask R-CNN. *The IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 22-29 Oct. 2017, pp. 2961-2969. DOI: 10.1109/ICCV.2017.322

Hesse, R. (2010). Lidar-derived local relief models – a new tool for archaeological prospection. *Archaeological Prospection* 17, pp. 67–72. DOI:10.1002/arp.374 https://onlinelibrary.wiley.com/doi/abs/10.1002/arp.374

Landauer, J. and Hesse, R. (2019). Machine learning for large area archaeological feature detection. Applying transfer learning to airborne lidar data. In: *24th Conference on Cultural Heritage and New Technologies*, November 4-6 2019, Vienna, Austria. https://www.chnt.at/wp-content/uploads/Machine-learning-for-large-area-archaeological-feature-detection.pdf

Kramer, I., Hare, J., and Cowley, D., (2019). Arran: a benchmark dataset for automated detection of archaeological sites on LiDAR data. Presented at: *Artificial Intelligence, Machine Learning and Deep Learning in Archaeology*, 7-8 November 2019, Rome, Italy.

Lambers, K., Verschoof-van der Vaart, W.B. and Bourgeois, Q.P.J. (2019). Integrating remote sensing, machine learning and citizen science in Dutch archaeological prospection. Remote Sensing 11, article no. 794. doi: 10/3390/rs11070794 https://www.mdpi.com/2072-4292/11/7/794/htm

Morrison, W. and Peveler, E. (2019). Beacons of the past. Visualising LiDAR on a large scale. In: *24th Conference on Cultural Heritage and New Technologies*, November 4-6 2019, Vienna, Austria. https://www.chnt.at/wp-content/uploads/Beacons-of-the-Past.pdf

Ren, S., He, K., Girshick, R. and Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(6), pp. 1137-1149. DOI: 10.1109/TPAMI.2016.2577031

Trier, Ø.D. and Pilø, L.H. (2012). Automatic detection of pit structures in airborne laser scanning data. *Archaeological Prospection* 19 (2), pp. 103-121. DOI: 10.1002/arp.1421

Trier, Ø.D., Pilø, L.H., and Johansen, H.M. (2015). Semi-automatic mapping of cultural heritage from airborne laser scanning data. *Semata* 27, pp. 159-186. Available at: http://www.usc.es/revistas/index.php/semata/article/view/2736

Trier, Ø.D., Zortea, M., and Tonning, C. (2015). Automatic detection of mound structures in airborne laser scanning data. *Journal of Archaeological Science: Reports* 2 (1), pp. 69-79. doi: 10.1016/j.jasrep.2015.01.005

Trier, Ø.D., Salberg, A.-B. and Pilø, L.H. (2018). Semi-automatic mapping of charcoal kilns from airborne laser scanning data using deep learning. In: Matsumoto, M and Uleberg, E. (eds). *CAA2016: Oceans of Data. Proceedings of the 44th Conference on Computer Applications and Quantitative Methods in Archaeology.* Oslo, Norway, 30 March-3 April 2016. Oxford: Archaeopress, pp. 219-231. Available at: http://archaeopress.com/ArchaeopressShop/Public/download.asp?id={6A565CFE-F617-4333-9818-4C13E78B7C1B}

Trier, Ø.D., Cowley, D.C. and Waldeland, A.U. (2019). Using deep neural networks on airborne laser scanning data: Results from a case study of semi-automatic mapping of archaeological topography on Arran, Scotland. *Archaeological Prospection* 26 (2), pp. 165–175. doi: 10.1002/arp.1731
https://onlinelibrary.wiley.com/doi/abs/10.1002/arp.1731

Verschoof-van der Vaart, W. B. and Lambers, K. (2019). Learning to look at LiDAR: The use of R-CNN in the automated detection of archaeological objects in LiDAR data from the Netherlands. *Journal of Computer Applications in Archaeology* 2(1), pp. 31–40. DOI: 10.5334/jcaa.32