



Kartverket

Sentinel-2 project and production of mosaic in NMA

*TerraNor remote sensing user meeting February 2020
Torgeir Ferdinand Klingenberg, Norwegian Mapping Authority*



Description of project

- WP-1 Project management, skills development and reporting to Norwegian Space Agency
- WP-2 Integrate data from the national ground segment in Geonorge.no
- WP-3 Testing use cases and functionality in satellittdata.no
- WP-4 Quality control of Sentinel-2 data and DEMs
- WP-5 Cloudless mosaic over Norway
- WP-6 Further develop the "Norway in Images" viewing service for Sentinel-2

WP1: Project management, skills development and reporting to Norwegian Space Agency

- Promote Copernicus data:
 - National Geodata Strategy: Action 13 - Utilizing Earth Observation Satellite Data
 - Geomatics Days 2019: Own "Satellite data" session
- National/global interaction:
 - Norway digital members – satellite meeting
 - EuroSDR Workshop: 4th Workshop on Copernicus / Sentinels

WP-2 Integrate data from the national ground segment in Geonorge.no

GEONORGE Search for map data and articles Map catalog (0) Articles (0)

Geonorge > Map Catalogue > Satellite data - Sentinel-2 - Cloudless Norway 2018 UInt-16

Satellite data - Sentinel-2 - Cloudless Norway 2018 UInt-16

Add to map	Download	Display coverage map	Help	Contact dataowner
Display product sheet	Show product specification	Display cartography	Webpage	Display productpage
Download metadata XML		Edit metadata		

A cloudless Sentinel-2 mosaic of Norway, assembled from Sentinel-2 scenes sensed between 30 June and 31 July 2018. The mosaic consist of the bands 2 through 8, 8A, 11, and 12. The datatype is UInt16, and the data basis is atmospherically corrected data (L2A). Vector data that contains the date of the mosaicked raster data is included in each tile.

- Viewing the latest data from satellitdata.no to Geonorge.no – work is in progress
- Other products available for download:
 - Cloudless mosaic 2015-2017 (RGB 8-bit)
 - Cloudless mosaic 2018 (RGB 8-bit)
- WMS service:
 - 2018 UInt-16, with download grid



Table Of Contents

- Layers
 - ✓ Sentinel-2 Norge: Mosaikk
 - ✓ Sentinel-2 Norge: Mosaikk
 - ✓ Rutenett
 - ✓ Mosaikk

Identify

Identify from: <Top-most layer>

- Rutenett
 - WMS Feature(s)

Location: 332 109,363 6 932 397,767 Meters

 **GEONORGE**

Sentinel-2 Skyfri mosaikk 2018 (L2A)
Kartverket

Skyfri mosaikk sammensatt fra Sentinel-2 data fra 30 juni til og med 31 juli 2018. Mosaikken består av båndene 2 til 8, 8A, 11 og 12. Datatypen er UInt16, og datagrunnlaget er atmosfærekorrigerede data (L2A). Det medfølger vektordata som inneholder dato for rasterdataene.

[Mer informasjon](#)

ID: 115

Image name: T32VPQ

Download: <https://nedlasting.geonorge.no/geonorge/FlyOgSatellittbilder/3>

Identified 1 feature

WMS service:

<https://wms.geonorge.no/skwms1/wms.sentinel2mosaikk?service=wms&version=1.3.0&request=getcapabilities>

WP-3 Testing use cases and functionality in satellittdata.no

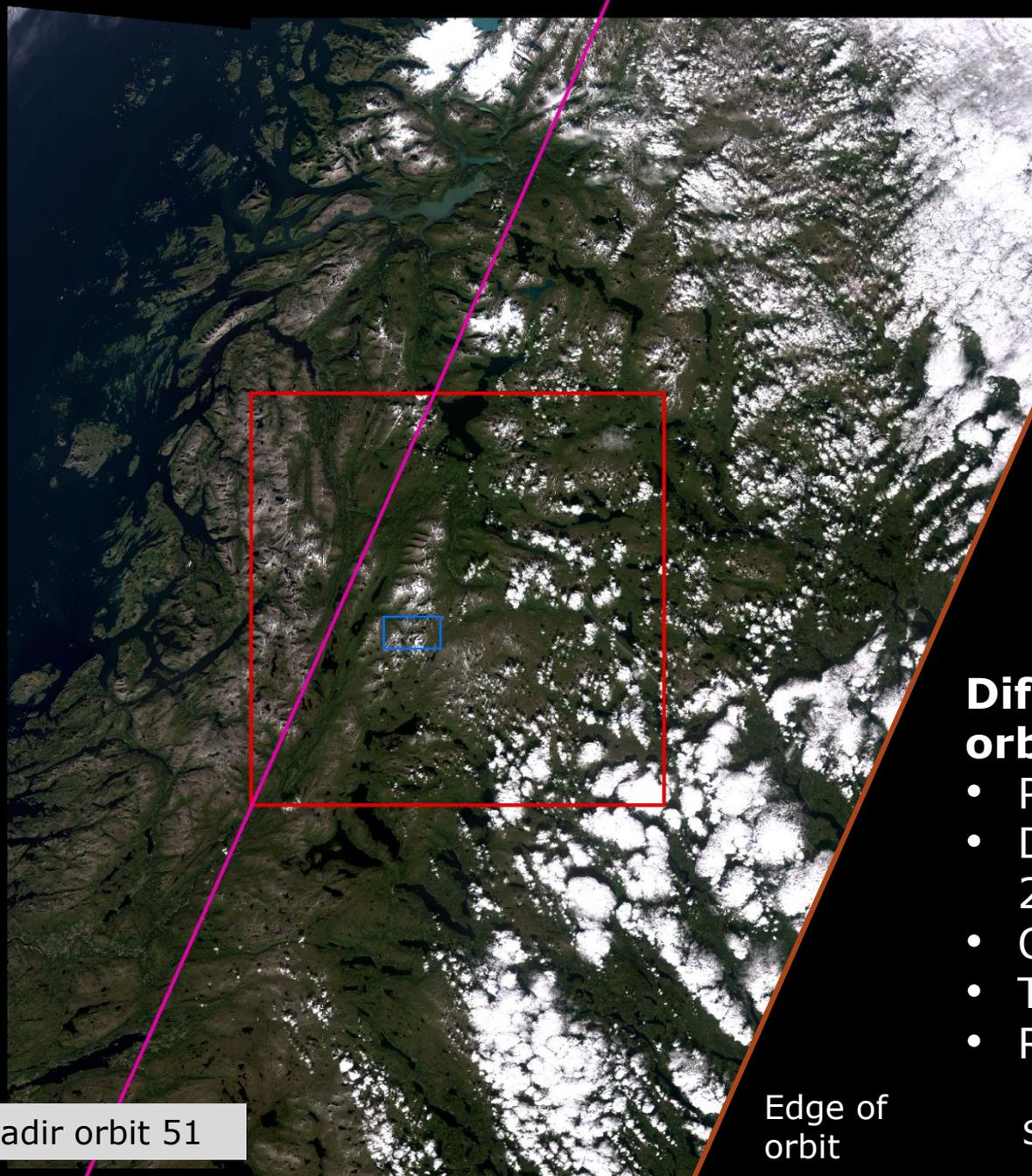
Main focus in this period is

- Focus on change detection and automatic cloudless mosaic
- Testing use cases and user interface in satellittdata.no
- Assists NOSA user survey

WP-4 Quality control of Sentinel-2 data and DEMs

Background info for this WP:

- Improve the geometric quality of Sentinel-2
 - Today: PlanetDEM90 is used for orthorectification of S2
 - Pilot project: National ground segment ~ DTM10 is used for orthorectification of S2
 - Look for the DTERRENGDATA suffix



Nadir orbit 51

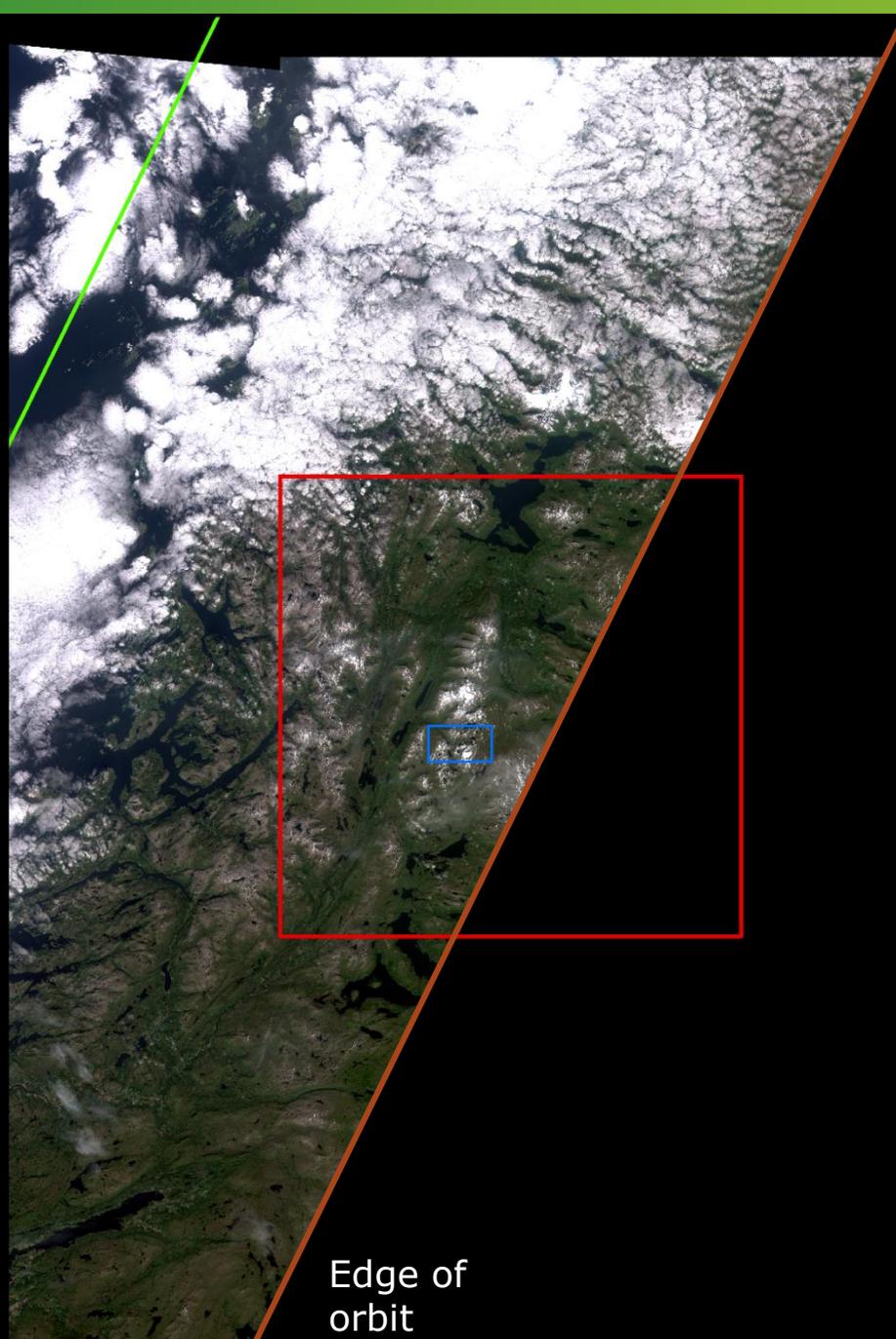
Edge of orbit

Different dates and orbits:

- PlanetDEM and DTM10
- Date: 2017-08-27 and 2017-09-03
- Orbit: R094 and R051
- Tile: T33WVN
- Region: Børgefjell

Sentinel-2 L1C 2017-08-27

Nadir orbit 94



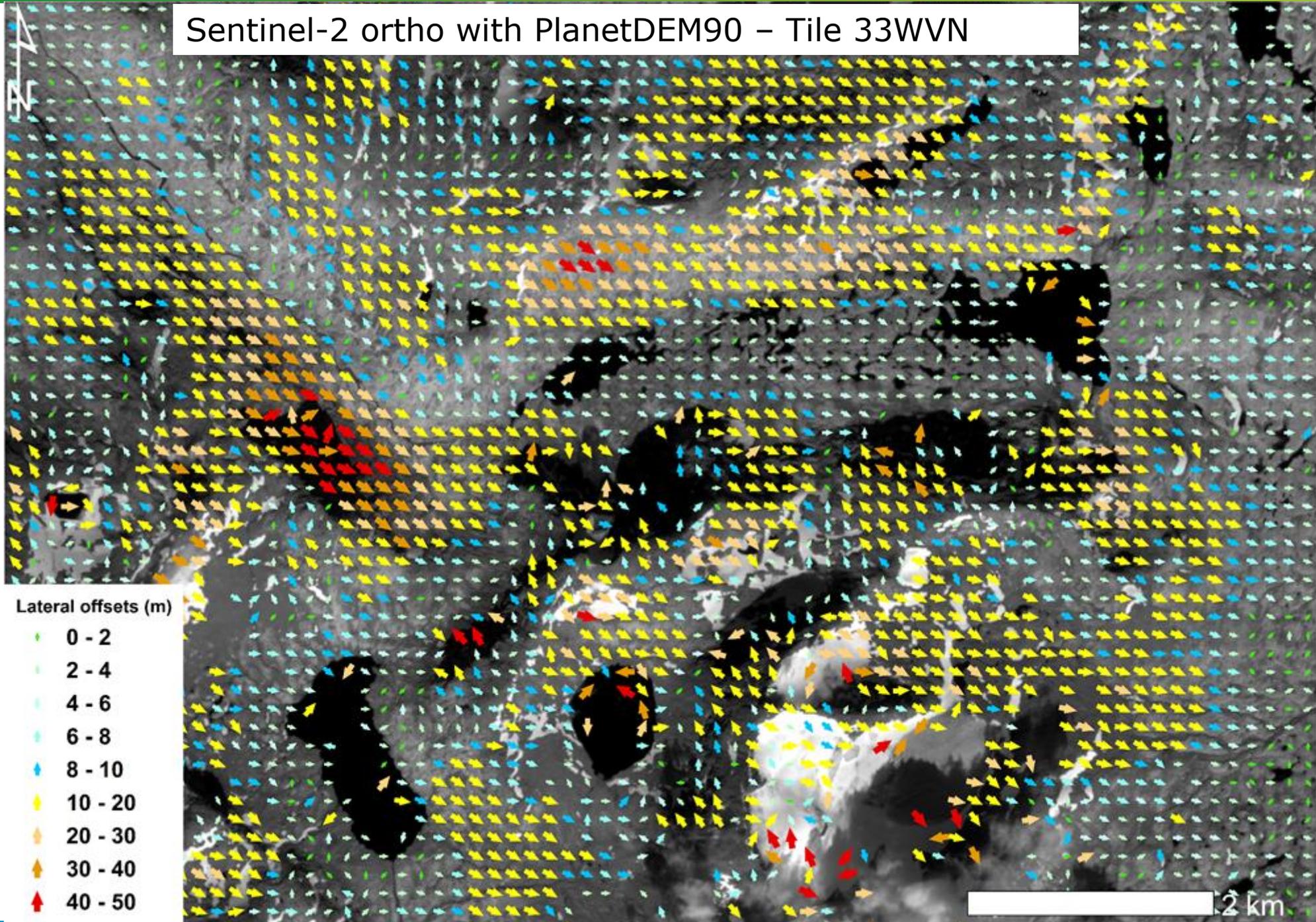
Different dates and orbits:

- PlanetDEM and DTM10
- Date: 2017-08-27 and 2017-09-03
- Orbit: R094 and R051
- Tile: T33WVN
- Region: Børgefjell

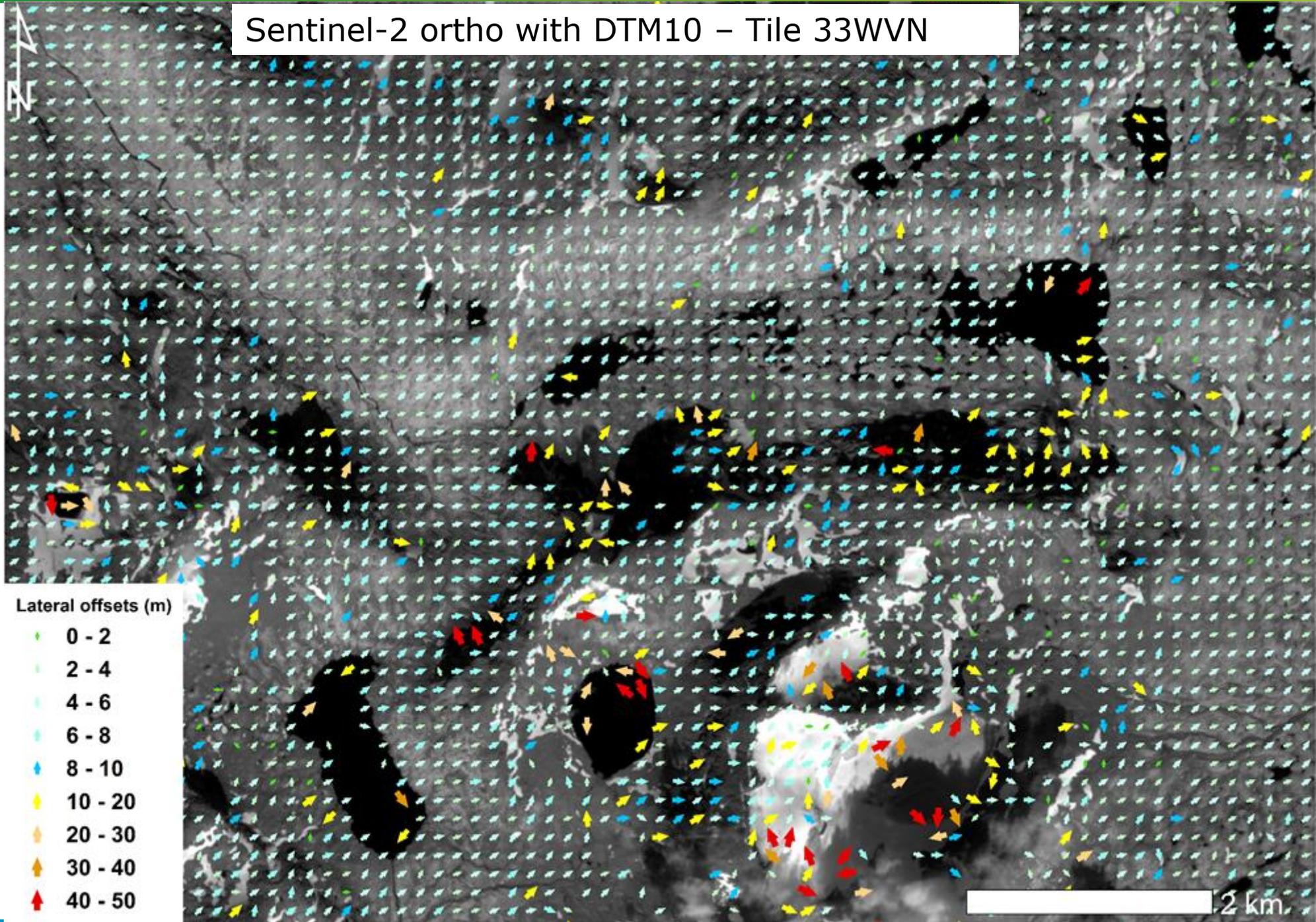
Edge of orbit

Sentinel-2 L1C 2017-08-27

Sentinel-2 ortho with PlanetDEM90 – Tile 33WVN



Sentinel-2 ortho with DTM10 – Tile 33WVN



WP-4 Quality control of Sentinel-2 data and DEMs

Background info for this WP:

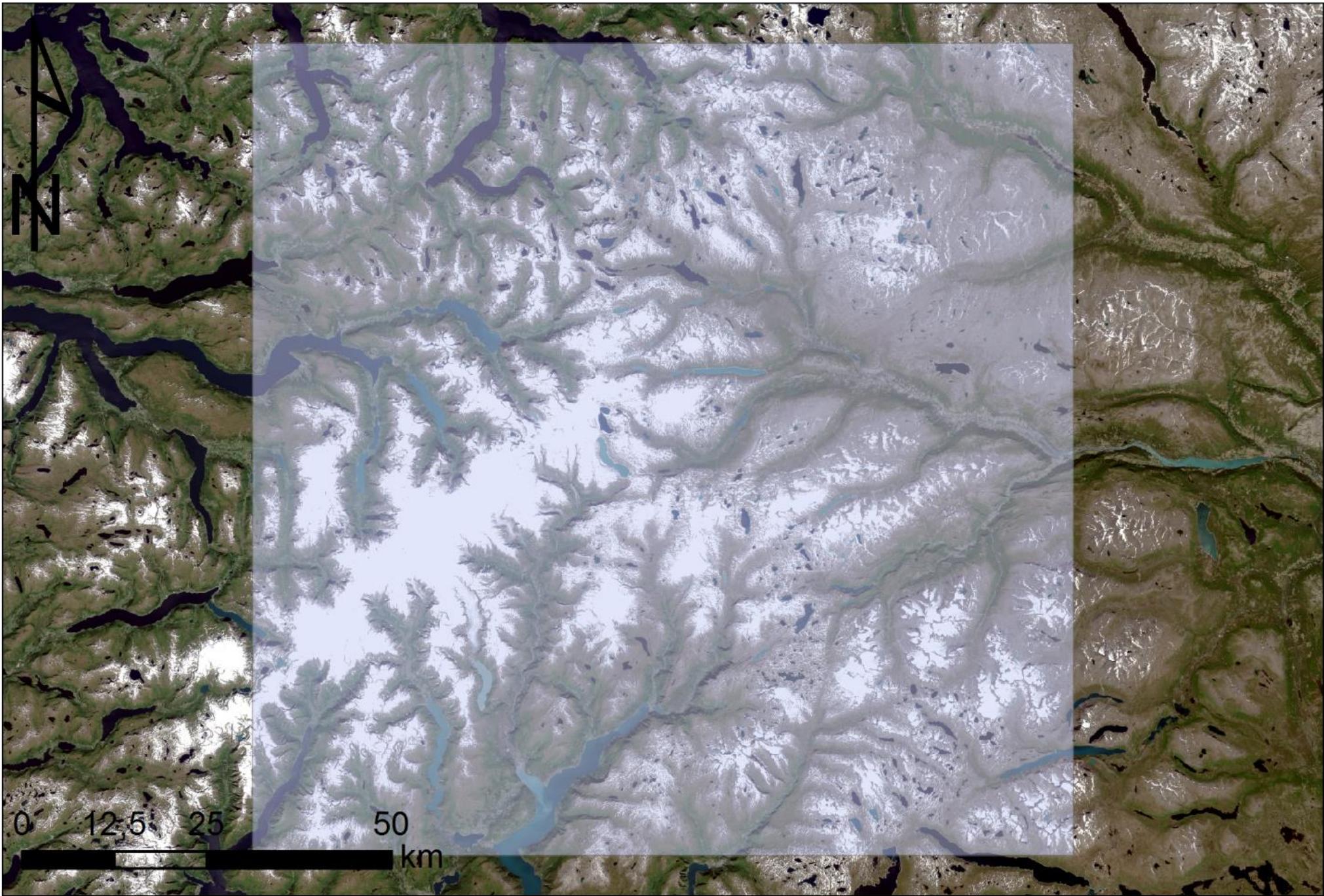
- Improve the geometric quality of Sentinel-2
 - Today: PlanetDEM90 is used for orthorectification of S2
 - Late 2020 new Copernicus DEM

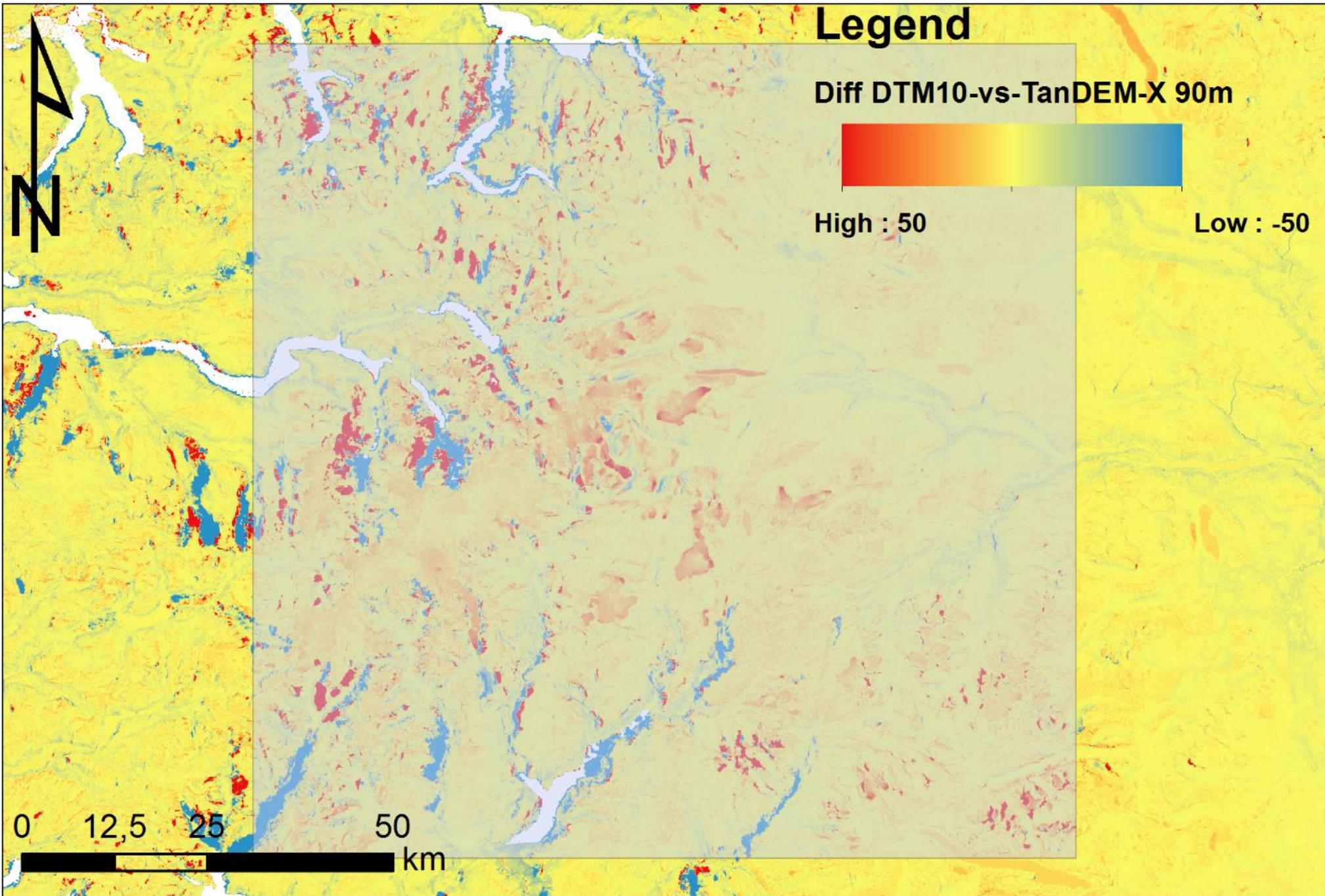
Produced Nordic DEM in collaboration with the Nordic countries

- ESA has requested data to test implementation in the new Copernicus DEM

Test of the height model TanDEM-X **90m** completed

- Not good enough for Nordic areas
- Copernicus DEM testing will be performed when available





WP-6 Further develop the "Norway in Images" viewing service for Sentinel-2

- It is now possible to see different individual layers in 3D in Norway in Pictures
- Sample areas are available at different times. Possibility of change analysis

3D



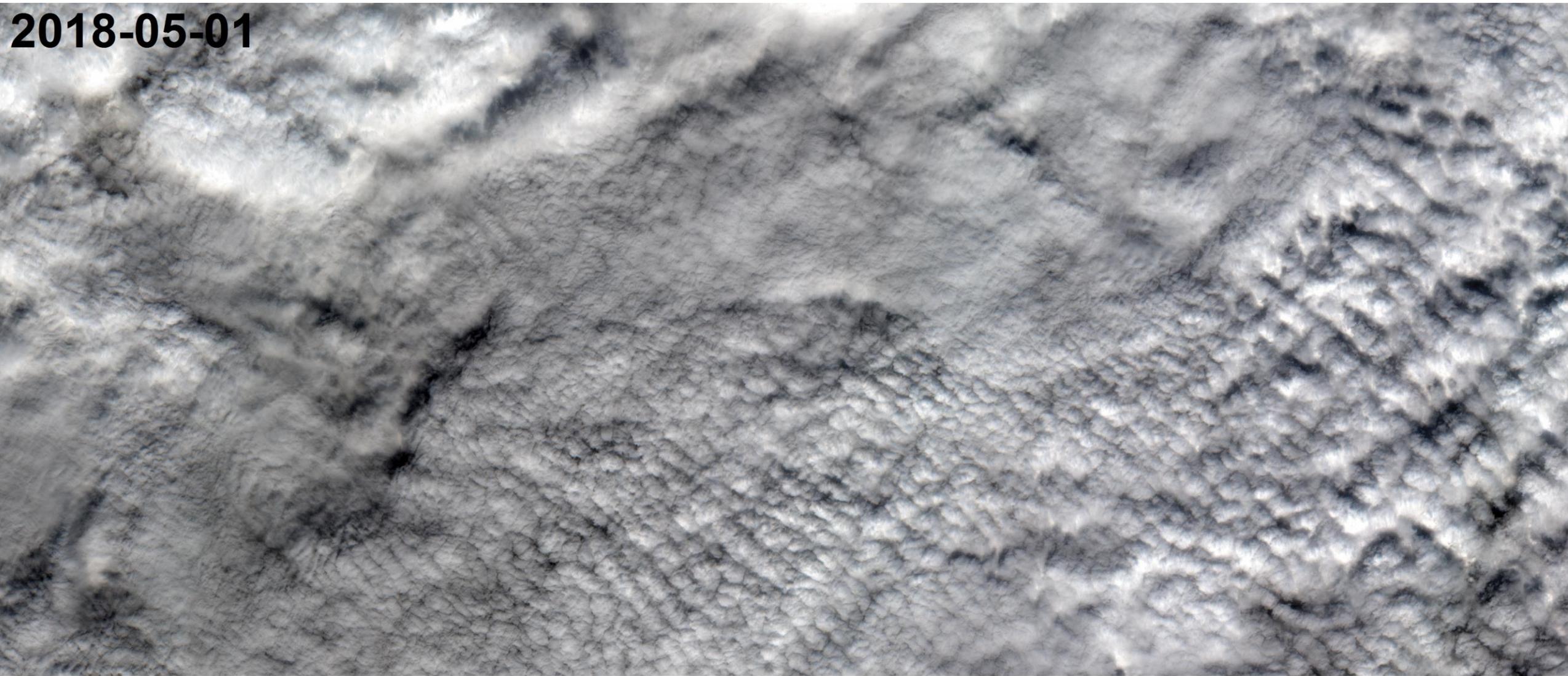
[Live demo: www.norgeibilder.no](http://www.norgeibilder.no)

WP-5 Cloudless mosaic over Norway

Cloudless mosaic 2019 (RGB and several bands) is in production

- Manually produced, but a semi-automatic method is tested
- Involves several different software:
 - PCI Geomatica
 - Mosaic Tool
 - Focus
 - Python libs
 - ArcMap
 - Anaconda Python
 - Sentinelsat
 - GDAL
 - Packages for U-Net CNN ~ CUDA
 - Erdas Imagine
 - Spatialbuilder

2018-05-01



Sentinel-2A

Sentinel-2B



1) Find the data

The screenshot displays the Sentinel Hub EO Browser interface. The browser's address bar shows the URL: `apps.sentinel-hub.com/eo-browser/?lat=61.656&lng=10.272&zoom=7&time=2018-07-26&preset=1_TRUE_COLOR&datasource=Sentinel-2%20L1C`. The interface includes a sidebar on the left with the following elements:

- EO Browser** header with a **Login** button.
- Search**, **Results**, **Visualization**, and **Pins** tabs.
- Dataset:** SENTINEL-2 L1C with a **SHOW L2A** button.
- Date:** 2018-07-26.
- Custom** rendering options:
 - True color (Based on bands 4,3,2)
 - False color (Based on bands 8,4,3)
 - NDVI (Based on the combination of bands (B8 - B4)/(B8 + B4))
 - False color (urban) (Based on bands 12,11,4)
 - Moisture index (Based on combination of bands (B8A - B11)/(B8A + B11))
 - SWIR (Based on bands 12,8A,4)
 - NDWI (Based on combination of bands (B3 - B8)/(B3 + B8))
 - NDSI (Based on combination of bands (B3 - B11)/(B3 + B11); values above 0.42 are regarded as snowy)
- Footer: **Free sign up** for all features, powered by **Sinerise** with contributions from the European Space Agency v2.20.6.
- SENTINEL HUB CUSTOM SCRIPT CONTEST** banner: THE DEADLINE IS EXTENDED TO JANUARY 31ST 2020!

The main map area shows a satellite image of Norway and Sweden, with labels for cities like **TRONDHEIM**, **OSLO**, **NORWAY**, **BERGEN**, **SWEDEN**, **STOCKHOLM**, **SUNDSVALL**, **Örebro**, **Norrköping**, **Linköping**, **Kristiansand**, **Trondheim**, **Oslo**, **Karlstad**, **Uppsala**, **Tampere**, and **Turku**. The map includes a search bar at the top right, a **Go to Place** button, and a vertical toolbar on the right with icons for home, location, layers, and other map functions. A scale bar at the bottom right indicates **50 km** and shows coordinates **Lat: 64.028, Lng: 0.703**. The bottom left corner features the **Ka** logo and the text **Carto © CC BY 3.0, OpenStreetMap © ODM**. The bottom center has links for **About EO Browser**, **Contact us**, and **Get data**.

2) Download the data

The image shows two browser windows side-by-side. The left window is the EO Browser interface, displaying a satellite image of a coastal region with a white polygon overlay. The right window is geojson.io, showing the same polygon converted into a JSON file.

EO Browser Interface:

- Dataset: SENTINEL-2 L1C
- Date: 2018-07-28
- Rendering options:
 - True color (Based on bands 4,3,2)
 - False color (Based on bands 8,4,3)
 - NDVI (Based on the combination of bands (B8 - B4)/(B8 + B4))
 - False color (urban) (Based on bands 12,11,4)
 - Moisture index (Based on combination of bands (B8A - B11)/(B8A + B11))
 - SWIR (Based on bands 12,8A,4)
 - NDWI (Based on combination of bands (B3 - B8)/(B3 + B8))
 - NDSI (Based on combination of bands (B3 - B11)/(B3 + B11); values above 0.42 are regarded as snowy)

geojson.io Interface:

- JSON view showing a FeatureCollection with a single Feature.
- The Feature has a Polygon geometry with the following coordinates (simplified):

```
[ [ [ [ 12.216796875, 62.91523303947614 ], [ 12.216796875, 63.203925767041305 ], [ 12.41455078125, 63.568120480921074 ], [ 13.38134765625, 63.89873081524394 ], [ 14.589843749999998, 64.95146502589559 ], [ 15.534667968749998, 66.05371622067922 ], [ 16.14990234375, 66.73990169639414 ], [ 16.58935546875, 67.42436394630637 ], [ 16.962890625, 67.73443510366032 ], [ 18.017578125, 67.94990041419247 ], [ 18.25927734375, 68.23682270936281 ], [ 18.43505859375, 68.51214331858073 ] ] ] ] ]
```

2) Download the data

```
Sentinel-hub EO-Browser
apps.sentinel-hub.com/eo-browser/?lat=68.29&lng=15.36&zoom=6&time=2018-07-28&preset=1_TRUE_COLOR&datasource=Sentinel-2%20L1C

geojson.io
Not secure | geojson.io/#map=5/68.982/21.775

Anaconda Prompt
(base) C:\Users\DFABRUKER>I:
(base) I:\>cd 2018_DATA\FOLDERS\GeoJson
(base) I:\2018_DATA\FOLDERS\GeoJson>activate sentinelSAT
(sentinelSAT) I:\2018_DATA\FOLDERS\GeoJson>pip show sentinelSAT
Name: sentinelSAT
Version: 0.12.2
Summary: Utility to search and download Copernicus Sentinel satellite images
Home-page: https://github.com/sentinelSAT/sentinelSAT
Author: Kersten Clauss
Author-email: kersten.clauss@gmail.com
License: GPLv3+
Location: c:\users\dfabruker\appdata\local\continuum\anaconda3\envs\sentinelSAT\lib\site-packages
Requires: geojson, requests, six, tqdm, geomet, html2text, click
Required-by:

(sentinelSAT) I:\2018_DATA\FOLDERS\GeoJson>sentinelSAT -u User -p Password -g 20180728.geojson -s 20180728 -e 20180729 --sentinel 2 --producttype S2MSI1C --path I:\2018_DATA --url https://colhub.met.no/search?q=* --download
Found 74 products
Will download 74 products
Downloading 1fb40445-d5d3-4509-b13f-90b7a0d4cfe8 to I:\2018_DATA\S2A_MSI1C_20180728T101031_N0202_R022_T35WNU_20180828T112302_DTERRENGDATA.zip
Downloading: 100%
MD5 checksumming: 100%
1/74 products downloaded
Downloading f5a83942-e4ba-4f8a-8f06-bd9274a64c91 to I:\2018_DATA\S2A_MSI1C_20180728T101031_N0202_R022_T35WPT_20180828T112302_DTERRENGDATA.zip
748M/748M [00:12<00:00, 59.7MB/s]
748M/748M [00:01<00:00, 488MB/s]
```

The screenshot shows the SentinelSAT documentation page for the 'stable' version. The page title is 'Docs » SentinelSAT' and it includes an 'Edit on GitHub' link. The main heading is 'SentinelSAT', followed by the text: 'SentinelSAT makes searching, downloading and retrieving the metadata of Sentinel satellite images from the Copernicus Open Access Hub easy. It offers an easy-to-use command line interface'. A code block shows the command: `sentinelSAT -u <user> -p <password> -g </path/to/search_polygon.geojson> --sentinel 2 --cloud 30`. Below this, it says 'and a powerful Python API.' and shows a code snippet: `from sentinelSAT import SentinelAPI, read_geojson, geojson_to_wkt`. On the right side, there is a map of a region in Russia with a search results list showing coordinates and product IDs. The map includes labels for 'Mурманск', 'Архангелъградская область', 'Петрозаводск', and 'С.-Петербург'. The search results list includes coordinates like [14.589843, 64.951465] and product IDs like '15.534667968749998'.

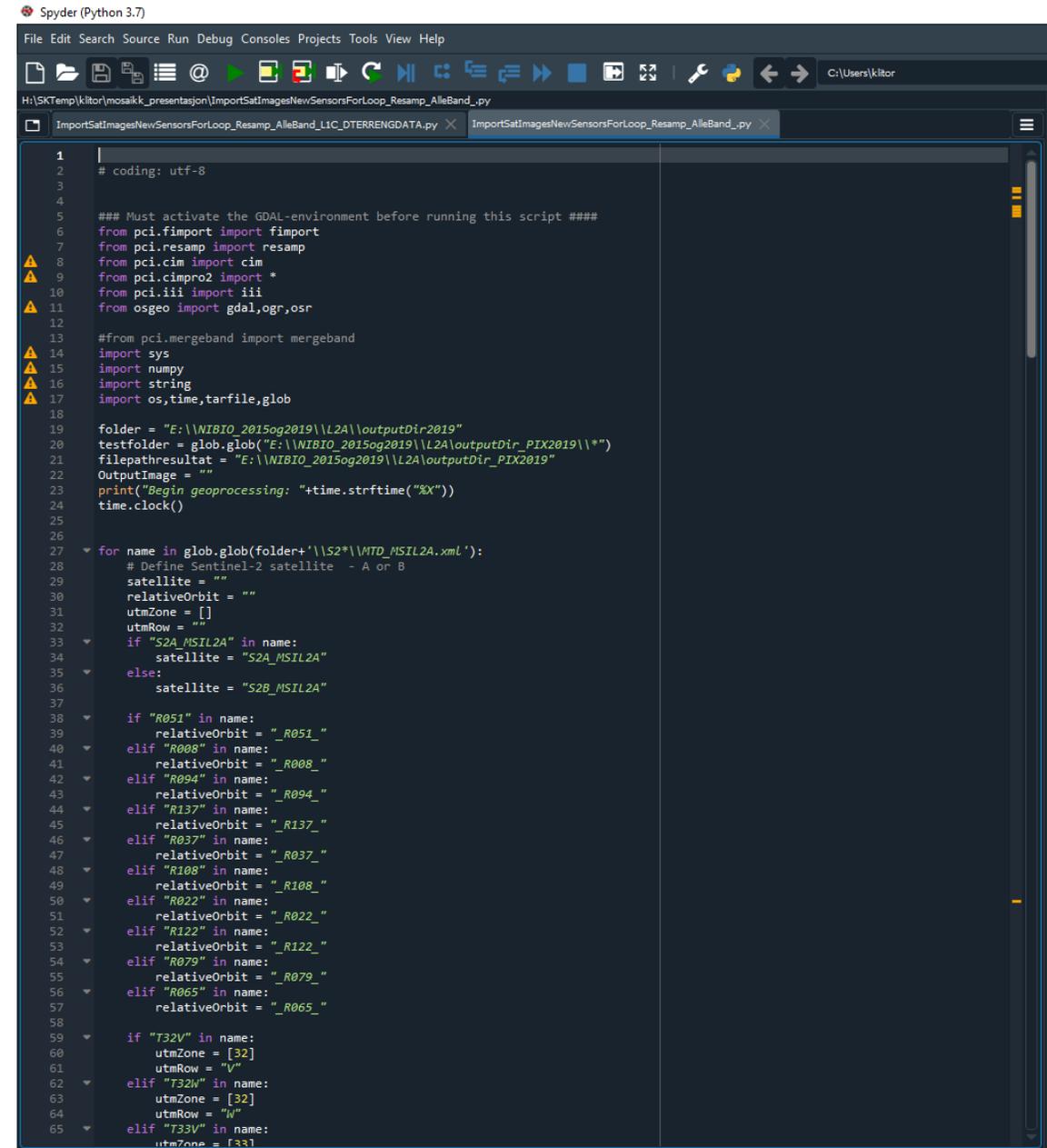
3) Prepare the data

Python libs:

- pci.fimport
 - From auxiliary files to .pix files
- pci.resamp
 - Resample 20 m bands to 10 m
 - Nearest neighbour
- pci.cimpro2
 - Prepare for merged file
- pci.iii
 - Merge bands

Output:

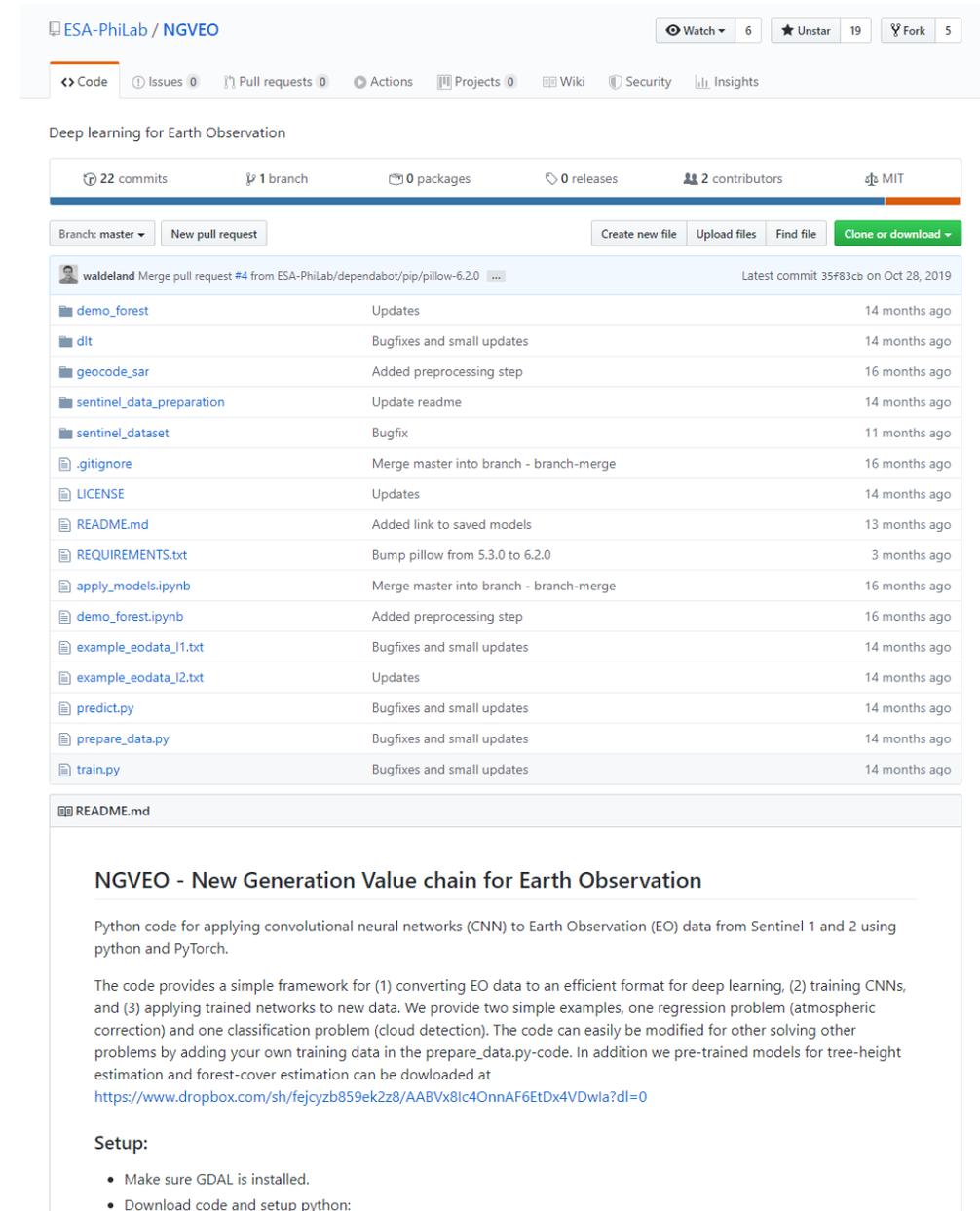
10 bands at 10 m resolution



```
1 | # coding: utf-8
2 |
3 |
4 |
5 | ### Must activate the GDAL-environment before running this script ###
6 | from pci.fimport import fimport
7 | from pci.resamp import resamp
8 | from pci.cim import cim
9 | from pci.cimpro2 import *
10 | from pci.iii import iii
11 | from osgeo import gdal,ogr,osr
12 |
13 | #from pci.mergerband import mergerband
14 | import sys
15 | import numpy
16 | import string
17 | import os,time,tarfile,glob
18 |
19 | folder = "E:\\WIBIO_2015og2019\\L2A\\outputDir2019"
20 | testfolder = glob.glob("E:\\WIBIO_2015og2019\\L2A\\outputDir_PIX2019\\*")
21 | filepathresultat = "E:\\WIBIO_2015og2019\\L2A\\outputDir_PIX2019"
22 | OutputImage = ""
23 | print("Begin geoprocessing: "+time.strftime("%X"))
24 | time.clock()
25 |
26 |
27 | for name in glob.glob(folder+'\\S2*\\MTD_MSIL2A.xml'):
28 |     # Define Sentinel-2 satellite - A or B
29 |     satellite = ""
30 |     relativeOrbit = ""
31 |     utmZone = []
32 |     utmRow = ""
33 |     if "S2A_MSIL2A" in name:
34 |         satellite = "S2A_MSIL2A"
35 |     else:
36 |         satellite = "S2B_MSIL2A"
37 |
38 |     if "R051" in name:
39 |         relativeOrbit = "_R051_"
40 |     elif "R008" in name:
41 |         relativeOrbit = "_R008_"
42 |     elif "R094" in name:
43 |         relativeOrbit = "_R094_"
44 |     elif "R137" in name:
45 |         relativeOrbit = "_R137_"
46 |     elif "R037" in name:
47 |         relativeOrbit = "_R037_"
48 |     elif "R108" in name:
49 |         relativeOrbit = "_R108_"
50 |     elif "R022" in name:
51 |         relativeOrbit = "_R022_"
52 |     elif "R122" in name:
53 |         relativeOrbit = "_R122_"
54 |     elif "R079" in name:
55 |         relativeOrbit = "_R079_"
56 |     elif "R065" in name:
57 |         relativeOrbit = "_R065_"
58 |
59 |     if "T32V" in name:
60 |         utmZone = [32]
61 |         utmRow = "V"
62 |     elif "T32W" in name:
63 |         utmZone = [32]
64 |         utmRow = "W"
65 |     elif "T33V" in name:
66 |         utmZone = [33]
```

4) Automatic cloud detection

- NGVEO - New Generation Value chain for Earth Observation
- Link: <https://github.com/ESA-PhiLab/NGVEO>
- License: MIT
- Main steps:
 - Prepare data
 - Train data
 - Predict data ~ clouds
 - Threshold
 - Vectorize
- **Authors:**
- Anders U. Waldeland* (anders@nr.no), Arnt-Børre Salberg*, Alessandro Marin** and Øyvind Due Trier*
- * Norwegian Computing Center (<https://www.nr.no/>)
- ** Phi Lab, European Space Agency (<http://blogs.esa.int/philab/>)



ESA-PhiLab / NGVEO

Watch 6 Unstar 19 Fork 5

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights

Deep learning for Earth Observation

22 commits 1 branch 0 packages 0 releases 2 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Commit Message	Time Ago
demo_forest	Updates	14 months ago
dlt	Bugfixes and small updates	14 months ago
geocode_sar	Added preprocessing step	16 months ago
sentinel_data_preparation	Update readme	14 months ago
sentinel_dataset	Bugfix	11 months ago
.gitignore	Merge master into branch - branch-merge	16 months ago
LICENSE	Updates	14 months ago
README.md	Added link to saved models	13 months ago
REQUIREMENTS.txt	Bump pillow from 5.3.0 to 6.2.0	3 months ago
apply_models.ipynb	Merge master into branch - branch-merge	16 months ago
demo_forest.ipynb	Added preprocessing step	16 months ago
example_eodata_11.txt	Bugfixes and small updates	14 months ago
example_eodata_12.txt	Updates	14 months ago
predict.py	Bugfixes and small updates	14 months ago
prepare_data.py	Bugfixes and small updates	14 months ago
train.py	Bugfixes and small updates	14 months ago

README.md

NGVEO - New Generation Value chain for Earth Observation

Python code for applying convolutional neural networks (CNN) to Earth Observation (EO) data from Sentinel 1 and 2 using python and PyTorch.

The code provides a simple framework for (1) converting EO data to an efficient format for deep learning, (2) training CNNs, and (3) applying trained networks to new data. We provide two simple examples, one regression problem (atmospheric correction) and one classification problem (cloud detection). The code can easily be modified for other solving other problems by adding your own training data in the prepare_data.py-code. In addition we pre-trained models for tree-height estimation and forest-cover estimation can be downloaded at <https://www.dropbox.com/sh/fejcyzb859ek2z8/AABVx8Ic4OnnAF6EtDx4VDwIa?dl=0>

Setup:

- Make sure GDAL is installed.
- Download code and setup python:

4-1) Automatic cloud detection

- Prepare data:
 - Converts to numpy memmap files

```
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\DFABRUKER
F:\2019_L1C_MOSADK\NGVEO_ForMosakk\prepare_data.py
s2_cloud_thresholds_GDAL_and_numpy.py - F:\... Kode Vectorize_clouds_S2.py example_eodata_I2.txt example_eodata_I1.txt Fungierende_GPT_tekst.txt prepare_data.py predict.py* Brute_Force_Matching_with_SIFT.py sft

1 import json
2 import os
3 import shutil
4 from sentinel_dataset_utils import parse_eodata_folder_name
5 from sentinel_data_preparation.data_preparation import DataPreparation
6
7 #Input data are level-1
8 config = {
9     "sensor": "sentinel-2",
10    "eocloud_path_list": "example_eodata_I1.txt",
11    "rad_cor": "taa",
12    "resolution": "10m",
13    "bands": ["b02", "b03", "b04", "b08", # 10m
14            "b05", "b06", "b07", "b8a", "b11", "b12", # 20m
15            "b01", "b09", "b10"], # 60m
16    "process_clouds": "yes",
17    "process_target_data": "no",
18    "outdir": 'data/',
19    # 'tile_ids': ["T29S0B", "T29S0C", "T30S7J", "T29TPE", "T30SUJ", "T32THP", "T32TNS", "T32THP", "T32TPR", "T32THL", "T32THK", "T32TNT", "T32UPU", "T32TPT", "T32QUU", "T32UNU", "T32THW", "T32THM"],
20    'tile_ids': ["T32VKK", "T32VKL", "T32VKM", "T32VKN", "T32VKP", "T32VKQ", "T32VLJ", "T32VLK", "T32VLL", "T32VLM", "T32VLN", "T32VLP", "T32VLQ", "T32VLR", "T32VLU",
21              "T32VMM", "T32VML", "T32VMP", "T32VMN", "T32VMP", "T32VMQ", "T32VMR", "T32VWK", "T32VWL", "T32VWM",
22              "T32VWN", "T32VWP", "T32VWQ", "T32VWR", "T32VPL", "T32VPM", "T32VPP", "T32VPR", "T32VPR", "T32VPR", "T32VPR",
23              "T32WMS", "T32WMA", "T32WMS", "T32WMT", "T32WNU", "T32WNV", "T32WNS", "T32WPT", "T32WPA", "T32WPB",
24              "T32WPC", "T32WPU", "T32WPV", "T32WVP", "T32WVQ", "T32WVM", "T32WVJ", "T32WVK", "T32WVL", "T32WVU",
25              "T33WMA", "T33WMA", "T33WVP", "T33WVQ", "T33WVR", "T33WVP", "T33WVS", "T33WVT", "T33WVU", "T33WVR",
26              "T33WMS", "T33WMT", "T33WNU", "T33WVR", "T33WVS", "T33WVT", "T33WVU", "T33WVU", "T33WVU", "T33WVU", "T33WVU",
27              "T34NDB", "T34NDC", "T34NDU", "T34NEB", "T34NEC", "T34NED", "T34NEE", "T34NFA", "T34NFB", "T34NFC",
28              "T34NFD", "T34NFE", "T34NLF", "T34NLR", "T34NLS", "T34NLT", "T34NLU", "T34NLU", "T34NLU", "T34NLU", "T34NLU",
29              "T35WNU", "T35WNV", "T35WNS", "T35WNT", "T35WPU", "T35WVU", "T35WNS", "T35WNT", "T35WNT", "T35WNT", "T35WNT"],
30    }
31
32 s2_dp = DataPreparation(config)
33 s2_dp.run_all()
34 #
35
36 #For atmospheric corection we will use Level-2 data as labels
37 config = {
38     "sensor": "sentinel-2",
39     "eocloud_path_list": "example_eodata_I2.txt",
40     "rad_cor": "taa",
41     "resolution": "10m",
42     "bands": ["b02", "b03", "b04", "b08", # 10m
43             "b05", "b06", "b07", "b8a", "b11", "b12", # 20m
44             "b01", "b09"], # 60m
45     "process_clouds": "no",
46     "process_target_data": "no",
47     "outdir": 'data_atm_corr/',
48     'tile_ids': ["T32THP", "T32TNS", "T32TPR", "T32THL", "T32THK", "T32TNT", "T32UPU", "T32TPT", "T32QUU", "T32UNU", "T32THW", "T32THM"],
49    }
50
51 #s2_dp = DataPreparation(config)
52 #s2_dp.run_all()
53
54
55 #Move atmospheric corrected data into the data-folder to act as labels
56 def use_atm_corr_data_as_labels(atm_corr_dir, out_dir, delete_atm_corr_dir = False):
57
58     for dir, dirnames, filenames in os.walk(atm_corr_dir):
59         for filename in filenames:
60             if 'data_b' in filename:
61                 src = os.path.join(dir, filename)
62
63                 atm_corr_properties = parse_eodata_folder_name(dir.split('\\')[-1])
64                 date_atm_corr = str(atm_corr_properties['datetime']).replace('-', '').replace(':', '')
65
66                 #find corresponding output-folder
67                 for outdir, _, _ in os.walk(out_dir):
68                     if date_atm_corr in outdir:
69                         out_properties = parse_eodata_folder_name(outdir.split('\\')[-1])
70                         if atm_corr_properties['tile_id'] == out_properties['tile_id']:
71                             dest = os.path.join(outdir, filename.replace(atm_corr_dir, '').replace('\\data_b', '\\lbl_b')) # Rename data as labels
72                             print('Moving', src, 'to', dest)
73                             shutil.move(src, dest)
74
75                 if delete_atm_corr_dir:
76                     shutil.rmtree(atm_corr_dir)
77
78     use_atm_corr_data_as_labels('data_atm_corr\\', 'data\\')
79
80
81 ##
82
```

4-2) Automatic cloud detection

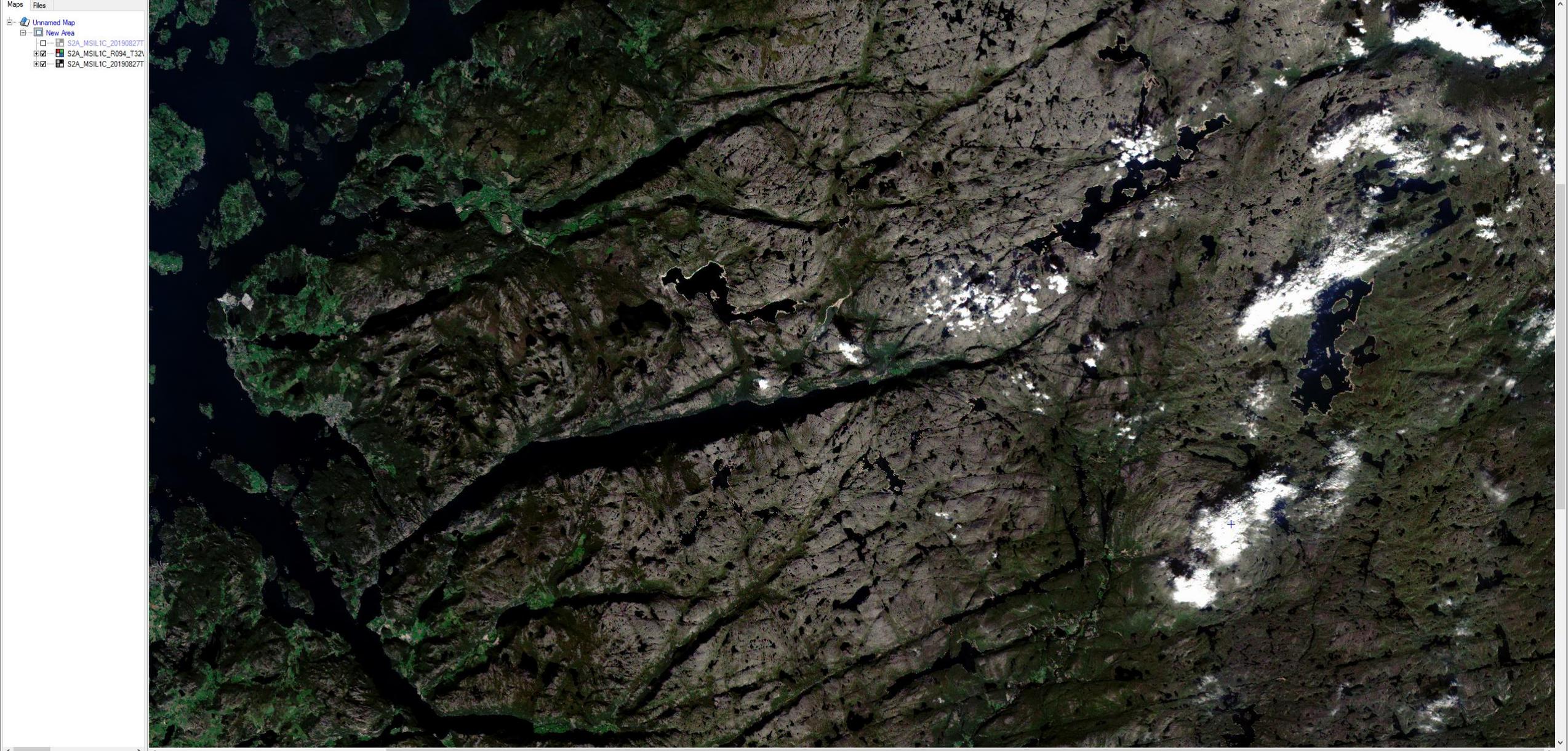
- Train data:
 - Train data using GPU/CUDA
 - Left to work for two days

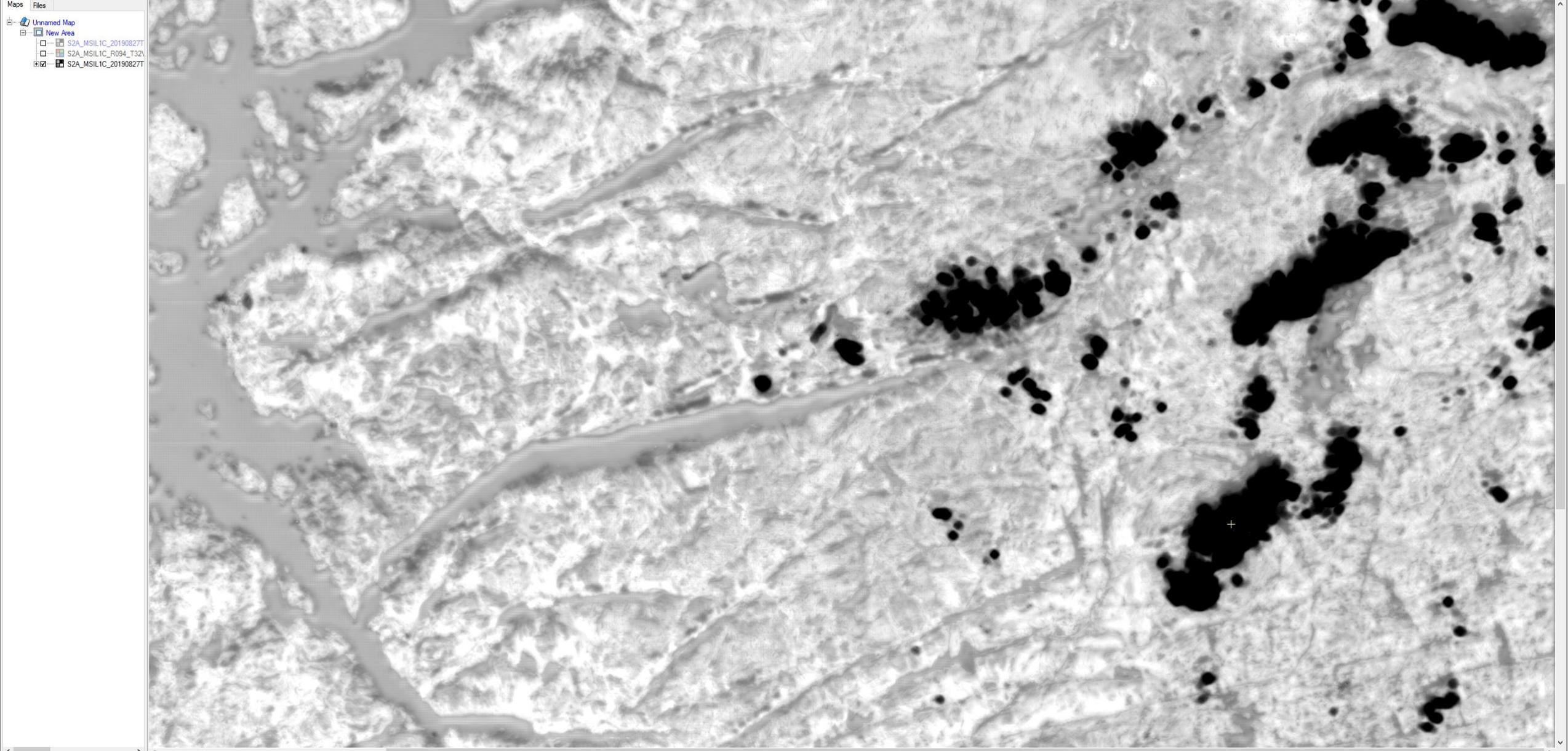
```
Spdyer (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\DFABRIJER
F:\2019_L1C-MOSAIK\NGVED_ForMosaik\train.py
_reclassfy.py s2a_angle_bands_mod.py ImportSatImagesNewSensorsForLoop_Resamp_AleBand_L1C_DITERRENGDATA.py setMetadataPasFiler.py Point1.XYZ.txt Temp_info_FID_14.txt Scale_Loop.py ImportSatImagesNewSensorsForLoop_Resamp_AleBand... train.py
1 import sys
2 import numpy as np
3 import torch
4 from torch.optim import Adam
5
6 from dlt.basic.batch import make_batch
7 from dlt.basic.cross_entropy import CrossEntropyLoss
8 from dlt.basic.mse_loss import MSELoss
9 from dlt.basic.pytorch_utils import torch_to_np, np_to_torch
10 from dlt.basic.summary import regression_summary, classification_summary
11 from dlt.basic.unet import UNet
12 from sentinel_dataset import Dataset
13 import os
14
15 data_bands = ['B02', 'B03', 'B04', 'B08', 'B05', 'B06', 'B07', 'B84', 'B11', 'B12']
16
17
18 GPU_NO = 0
19 batch_size = 8
20 win_size = [256, 256]
21 n_iteration = 30000
22 lr = 0.00004
23
24 # We showcase to simple examples; cloud detection and atmospheric correction. These tasks can easily be replaced by
25 # other tasks if required by replacing the labels. Labels may be provided as GeoTiffs and converted to the np-memmap-
26 # structure with the data-preparation tools.
27
28 #Make output folder
29 if not os.path.isdir('saved_models'):
30     os.mkdir('saved_models')
31
32 #Select which mode
33 if len(sys.argv) < 2:
34     sys.argv.append(2)
35
36 #Cloud detection (classification problem)
37 if int(sys.argv[1])==1:
38     label_bands = ['cloud_mask']
39     output_path = 'saved_models/cloud_detection.pt'
40
41     criterion = CrossEntropyLoss()
42     summary = classification_summary
43     n_outputs = 2
44     mask_clouds=False
45
46 # training_tiles = ["T29SQB", "T29SQC", "T30ST3"]
47 # validation_tiles = ["T29TPE"]
48 training_tiles = ["T32VNW", "T32WPS", "T32WPT", "T33VWK", "T32VPL", "T32VPM", "T32VWN", "T32VPP", "T32VPR", "T32VWQ", "T32VNR", "T32VPR", "T34WFC", "T35WHT"]
49 validation_tiles = ["T35WNT", "T35WNU", "T35WNV"]
50
51 #Atmospheric correction (regression problem)
52 else:
53     label_bands = ['B02'] #It is possible to add more bands here...
54     output_path = 'saved_models/atmospheric_correction_b02.pt'
55
56     criterion = MSELoss()
57     summary = regression_summary
58     n_outputs = len(label_bands)
59     mask_clouds = True
60
61     training_tiles = ["T32TWP", "T32TNS", "T32TNP", "T32TPR", "T32THL", "T32THK", "T32TNT", "T32UPU", "T32TPT", "T32UQU"]
62     validation_tiles = ["T32UNU", "T32TWW"]
63
64
65 #Model and optimizer
66 model = UNet(num_classes=n_outputs, in_channels=len(data_bands)).cuda(GPU_NO)
67 optimizer = Adam(model.parameters(), lr=lr)
68
69 # Datasets (T32TWN is reserved for test for the atmospheric correction setup)
70 train_dataset = Dataset([os.path.join('data', p) for p in training_tiles],
71     band_identifiers=data_bands,
72     label_identifiers=label_bands,
73 )
74
75 val_dataset = Dataset([os.path.join('data', p) for p in validation_tiles],
76     band_identifiers=data_bands,
77     label_identifiers=label_bands,
78 )
79
80 #Training steps
81 for iteration in range(n_iteration+1):
82     model.train()
```

4-3) Automatic cloud detection

- Predict data:
 - Tile-by-tile
 - Output: GeoTIFF

```
SpYder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\FABRUKER
F:\2019_1_IC-MOSAIK\INGIEO_ForMosaik\predict.py
S2_cloud_thresholds_GDAL_and_numpy.py -Fi...\Kode < Vectorize_clouds_S2.py < example_eodata_J2.txt < example_eodata_J1.txt < Fungerende_GPT_tekst.txt <
1 import torch
2 import numpy as np
3 import os
4 |
5 from dit.basic.predict_on_large_tile import apply_net_to_large_data
6 from dit.basic.unet import UNet
7 from sentinel_dataset import Dataset
8
9 #Configuration
10 network_path = 'saved_models\atmospheric_correction_b02.pt'
11 network_path = 'saved_models\cloud_detection.pt'
12 target_is_classes = False #Put to false for regression problems
13 n_output_channels = 2
14 model_name = network_path.split('\\')[1].replace('.pt','')
15
16 data_bands = ['B02', 'B03', 'B04', 'B08', 'B05', 'B06', 'B07', 'B04', 'B11', 'B12']
17 pred_win_size = [1024, 1024]
18 window_overlap = [50, 50]
19
20 #Load model with weights
21 net = UNet(n_output_channels, len(data_bands))
22 weights = torch.load(network_path, map_location=lambda storage, loc: storage)
23 net.load_state_dict(weights)
24
25 #Move model to GPU to enable GPU-computing
26 net.cuda()
27
28 #Put model in evaluation mode
29 net.eval()
30
31 #Load test-tile
32 #dataset = Dataset('data\T35WMT\...', band_identifiers=data_bands )
33 dataset = Dataset('data\T32WVK\...', band_identifiers=data_bands )
34
35 # Loop through tiles
36 for tile in dataset.tiles:
37     print('Predicting for tile', tile.tile_id, tile.file_name)
38
39     #Get data
40     data = tile.get_data(data_bands)
41     data = [np.expand_dims(d,-1) for d in data]
42     data = np.concatenate(data,-1)
43
44     #Run through network
45     output = apply_net_to_large_data(data, net, pred_win_size, window_overlap, apply_classifier=target_is_classes)
46
47     #Save output as GeoTiff
48     tile.export_prediction_to_tif( tile.file_name + '_' + model_name + '.tif', output)
49
50
51 #Load test-tile
52 #dataset = Dataset('data\T35WMT\...', band_identifiers=data_bands )
53 dataset = Dataset('data\T32WVK\...', band_identifiers=data_bands )
54
55 # Loop through tiles
56 for tile in dataset.tiles:
57     print('Predicting for tile', tile.tile_id, tile.file_name)
58
59     #Get data
60     data = tile.get_data(data_bands)
61     data = [np.expand_dims(d,-1) for d in data]
62     data = np.concatenate(data,-1)
63
64     #Run through network
65     output = apply_net_to_large_data(data, net, pred_win_size, window_overlap, apply_classifier=target_is_classes)
66
67     #Save output as GeoTiff
68     tile.export_prediction_to_tif( tile.file_name + '_' + model_name + '.tif', output)
69
70
71 #Load test-tile
72 #dataset = Dataset('data\T35WMT\...', band_identifiers=data_bands )
73 dataset = Dataset('data\T32WVK\...', band_identifiers=data_bands )
74
75 # Loop through tiles
76 for tile in dataset.tiles:
77     print('Predicting for tile', tile.tile_id, tile.file_name)
78
79     #Get data
80     data = tile.get_data(data_bands)
81     data = [np.expand_dims(d,-1) for d in data]
82     data = np.concatenate(data,-1)
```

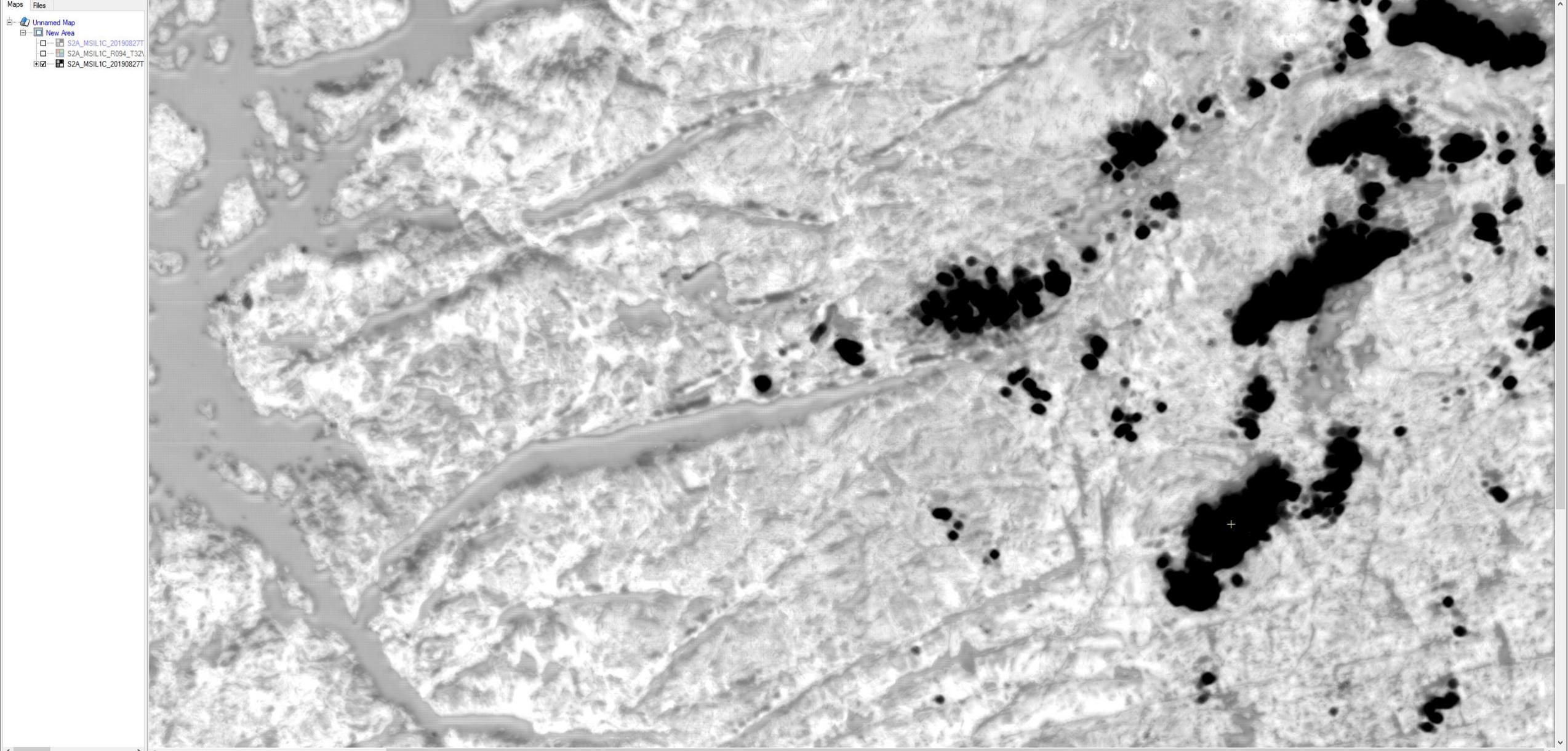




4-4) Automatic cloud detection

- **Threshold predicted data:**
 - Creates a new GeoTiff file with values for predicted clouds

```
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\DFABRUKER
F:\Vectorize_cloud\code\S2_cloud_thresholds_GDAL_and_numpy.py
S2_cloud_thresholds_GDAL_and_numpy.py - F:\..._code Vectorize_clouds_S2.py example_eodata_I2.txt example_eodata_I1.txt Fungerende_GPT_tekst.txt prepare_data.py predict.py
17 import numpy as np
18 import gdal
19 import glob, os
20 from gdalcopyproj_modified import gdalcopyproj # Make sure you have the gdalcopyproj_modified.py file in the same folder with this pythonscript
21 |
22 |
23 # (CHANGE ?)
24 thresholds1 = [0]
25 # (CHANGE ?)
26 # thresholds2 = [0.60, 0.70]
27
28
29 # Folder containing different Sentinel-2 folders
30 # CHANGE
31 # dataFolder = r"C:\Users\Klitor\Arbeidsmappe\PythonKode\WVE\Files"
32 # dataFolder = r"F:\Vectorize_cloud\Files\"
33 dataFolder = r"F:\2019_L1C-NDSATXX\WGVEO_ForMosaikk\VECTORIZE\WGVEO_CloudDetectionFiles\"
34
35 # Folder for results of different calculated thresholds
36 # CHANGE
37 # thresholdsFolder = r"C:\Users\Klitor\Arbeidsmappe\PythonKode\WVE\Files\thresholds\"
38 # thresholdsFolder = r"F:\Vectorize_cloud\thresholds\"
39 thresholdsFolder = r"F:\2019_L1C-NDSATXX\WGVEO_ForMosaikk\VECTORIZE\thresholds\"
40
41 # outputFolder
42 # tilFocalCalc = r"F:\Vectorize_cloud\focalMedian_in\"
43 # tilFocalCalc = r"F:\Vectorize_cloud\focalMedian_in\"
44 tilFocalCalc = r"F:\2019_L1C-NDSATXX\WGVEO_ForMosaikk\VECTORIZE\Result_1_Threshold\"
45
46 # Folder of already calculated NDMI files
47 # CHANGE
48 # NDMI_folder = "C:\Users\Klitor\Arbeidsmappe\PythonKode\WVE\Files\Scratch"
49 # NDMI_folder = "F:\Vectorize_cloud\Scratch\"
50 NDMI_folder = r"F:\2019_L1C-NDSATXX\WGVEO_ForMosaikk\VECTORIZE\scratch\"
51
52 # Single for loop in the data folder containing different *.SAFE folders
53 for folder in glob.glob(dataFolder):
54     print('Working with cloudfile in this folder: ' + folder)
55
56     # Open the cloudfile image and get its only band.
57     cloudf = glob.glob(folder + '*_cloud_detection.tif')
58     cloud = cloudf[0]
59     print(cloud)
60     cloud_filename = ''.join(cloud)
61     cloud_dataset = gdal.Open(cloud_filename)
62
63     # Display the dataset dimensions, number of bands, driver, and geotransform
64     cols = cloud_dataset.RasterXSize; print('# of columns: ',cols)
65     rows = cloud_dataset.RasterYSize; print('# of rows: ',rows)
66     print('# of Bands: ',cloud_dataset.RasterCount)
67     print('# driver: ',cloud_dataset.GetDriver().LongName)
68
69     print('projection: ',cloud_dataset.GetProjection())
70
71     print('geotransform: ',cloud_dataset.GetGeoTransform())
72
73     cloud_mapinfo = cloud_dataset.GetGeoTransform()
74     xmin = cloud_mapinfo[0]
75     ymax = cloud_mapinfo[3]
76
77     xmax = xmin + cloud_dataset.RasterXSize/cloud_mapinfo[1] #divide by pixel width
78     ymin = ymax + cloud_dataset.RasterYSize/cloud_mapinfo[5] #divide by pixel height (note sign +/-)
79     cloud_ext = (xmin,xmax,ymin,ymax)
80     print('cloud raster extent: ',cloud_ext)
81
82     cloud_raster = cloud_dataset.GetRasterBand(1)
83     noDataVal = cloud_raster.GetNoDataValue(); print('no data value: ',noDataVal)
84     scaleFactor = cloud_raster.GetScale(); print('scale factor: ',scaleFactor)
85     cloud_stats = cloud_raster.GetStatistics(True,True)
86     print('SERC cloud statistics: Minimum=%3f, Maximum=%3f, Mean=%3f, StDev=%3f' %
87           (cloud_stats[0], cloud_stats[1], cloud_stats[2], cloud_stats[3]))
88
89     cloud_array = cloud_dataset.GetRasterBand(1).ReadAsArray(0,0,cols,rows).astype(np.float)
90     # cloud_array[cloud_array==int(noDataVal)]=np.nan #Assign Cloud No Data Values to NaN
91     cloud_array=cloud_array/scaleFactor
92     print('SERC Cloud Array:\n',cloud_array) #display array values
93
94     cloud_array.shape
95
96     # Calculate the % of pixels that are NaN and non-zero:
97     pct_nan = np.count_nonzero(np.isnan(cloud_array))/(rows*cols)
98     print('% NaN: ',round(pct_nan*100,2))
```



Maps Files

- Unnamed Map
 - New Area
 - S2A_MSILIC_20190827T
 - S2A_MSILIC_R094_T32L
 - S2A_MSILIC_20190827T

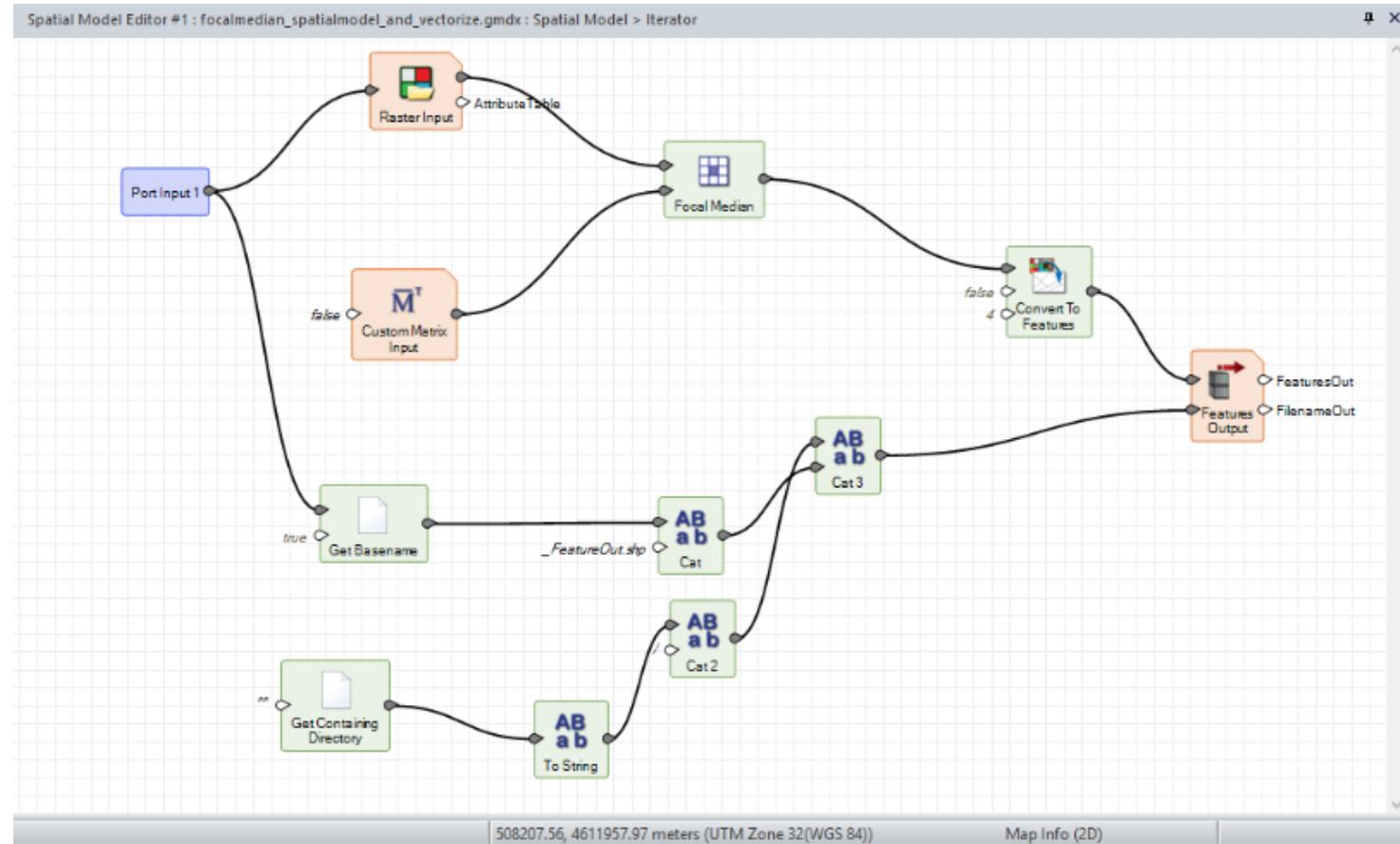


- Maps
- Files
- Unnamed Map
 - New Area
 - S2A_MSIL1C_20190827T
 - S2A_MSIL1C_R094_T32V
 - S2A_MSIL1C_20190827T



4-5) Automatic cloud detection

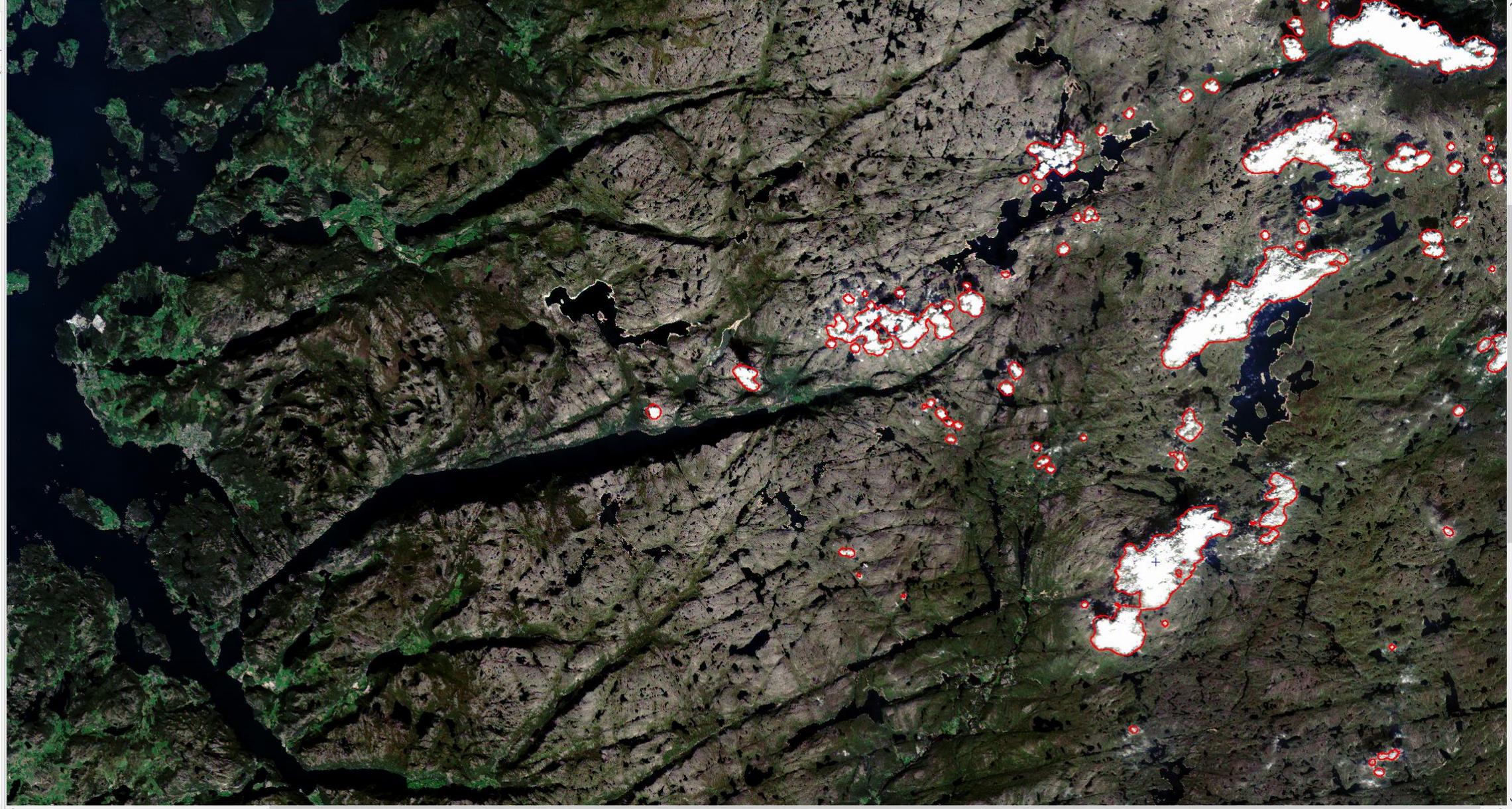
- **Vectorize predicted data:**
 - Creates a new shapefile of predicted clouds





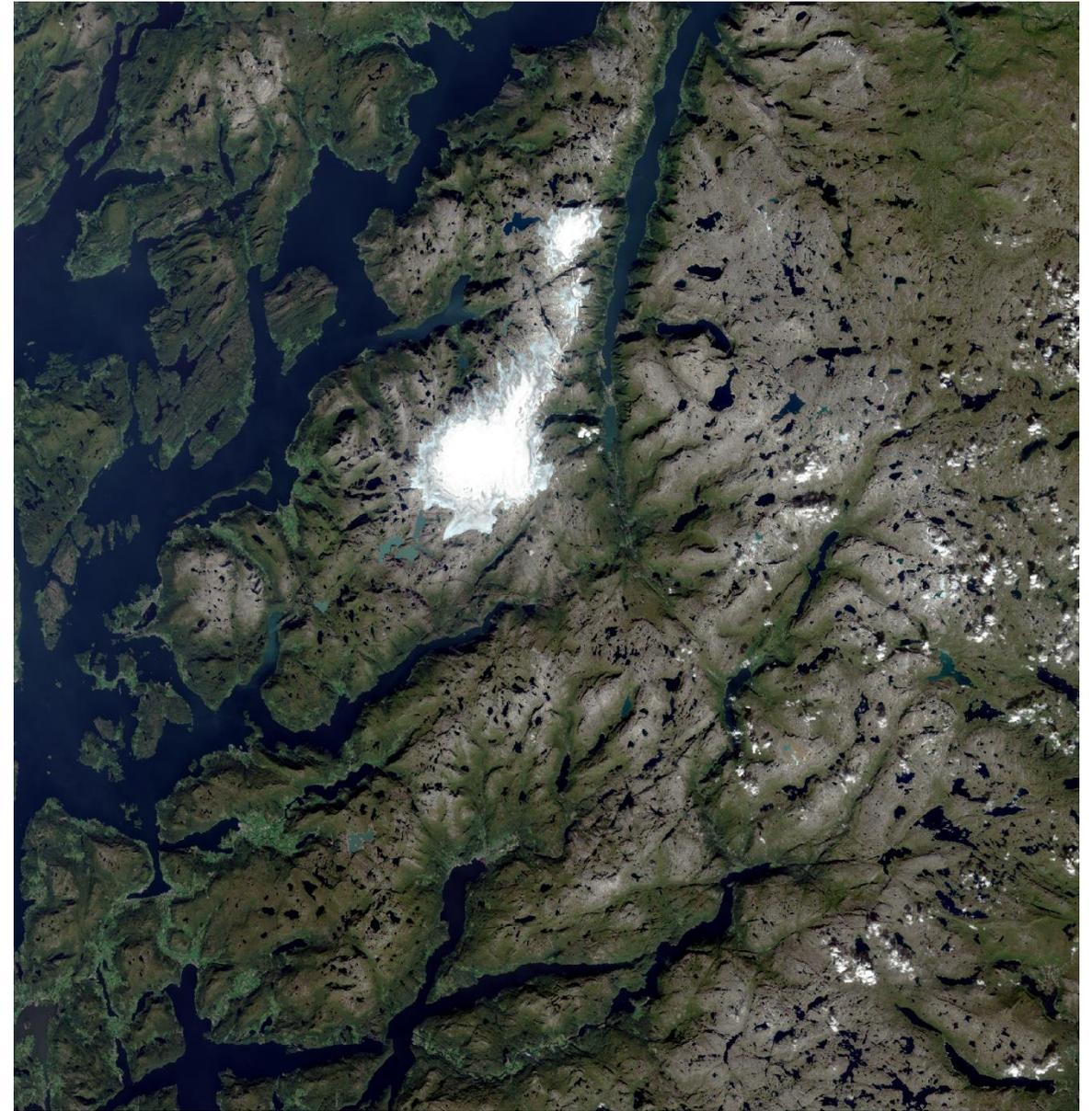
Maps Files

- Unnamed Map
 - New Area
 - S21+2a_mall1c_20190827105621
 - Default Polygon
 - S2A_MSIL1C_R094_T32VLL_



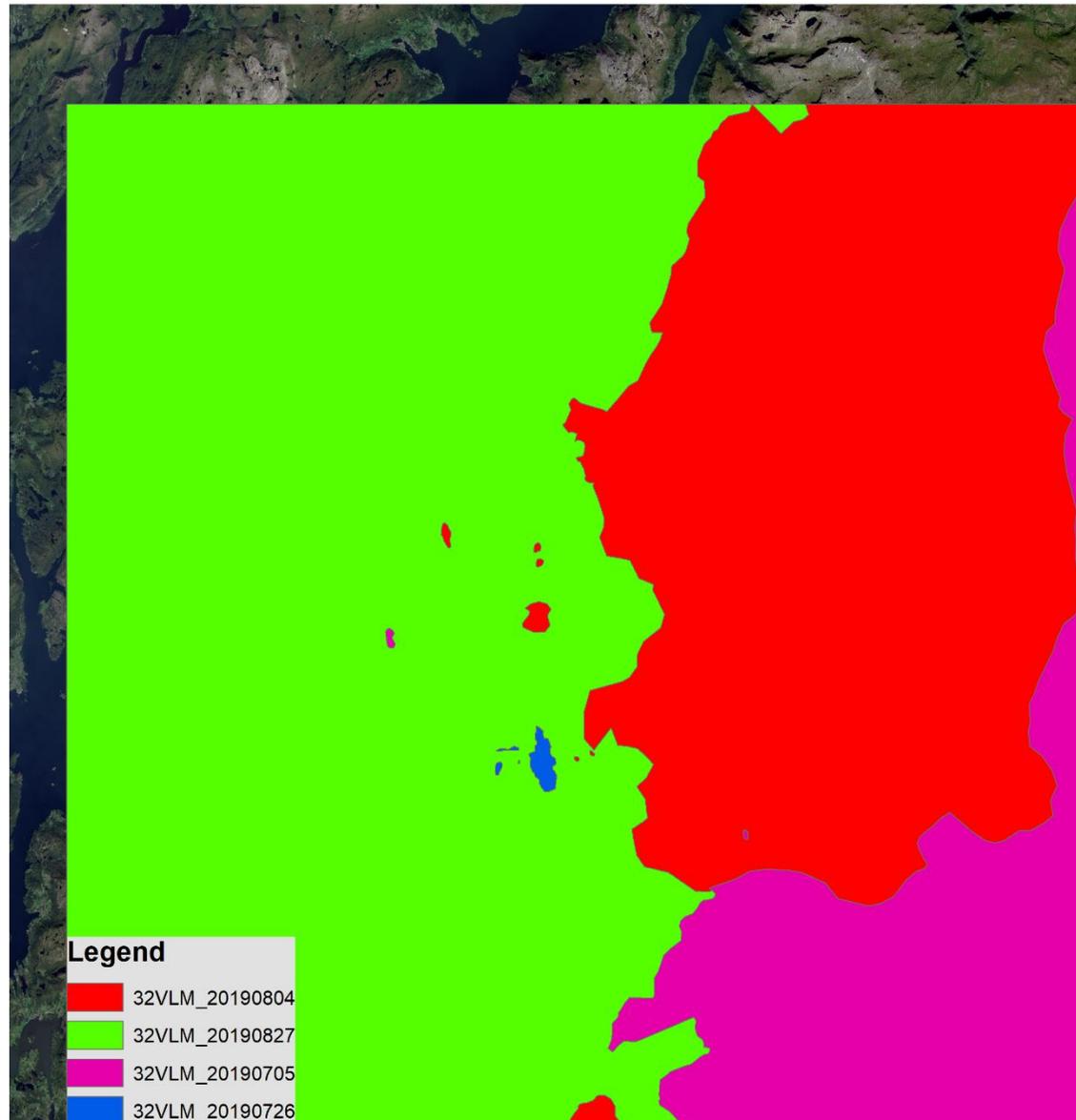
5) Vectorizing clouds

- Draw cloud and cloud shadow by hand
 - Use automatic cloud detection as guideline
- Vectorize tile-by-tile, with respect of neighbouring tile
 - Easier colorblend ~ no large time difference



Copernicus Sentinel-2 data(27-08-2019)

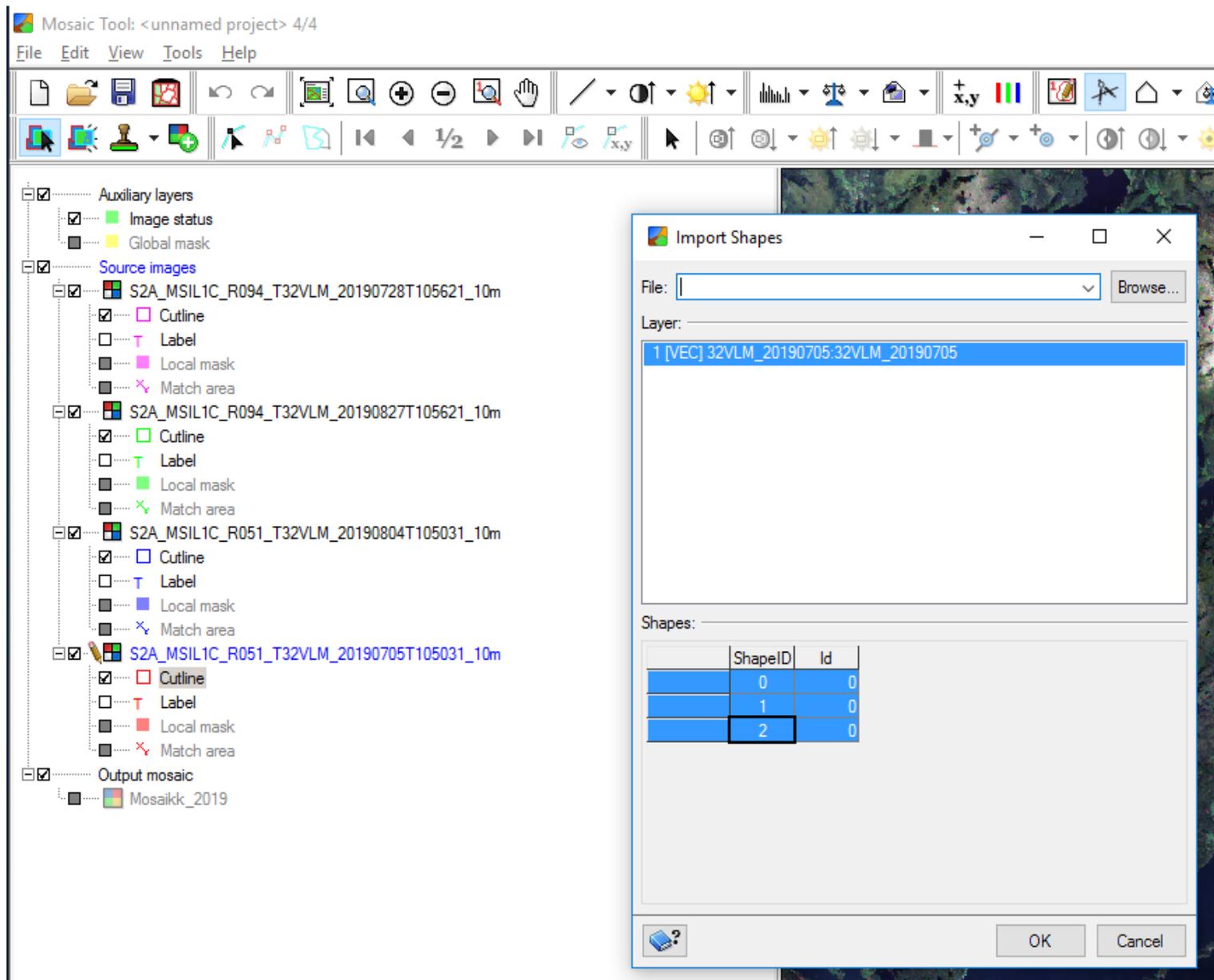
5) Vectorizing clouds



6) Mosaicing

PCI Geomatica: Mosaic Tool

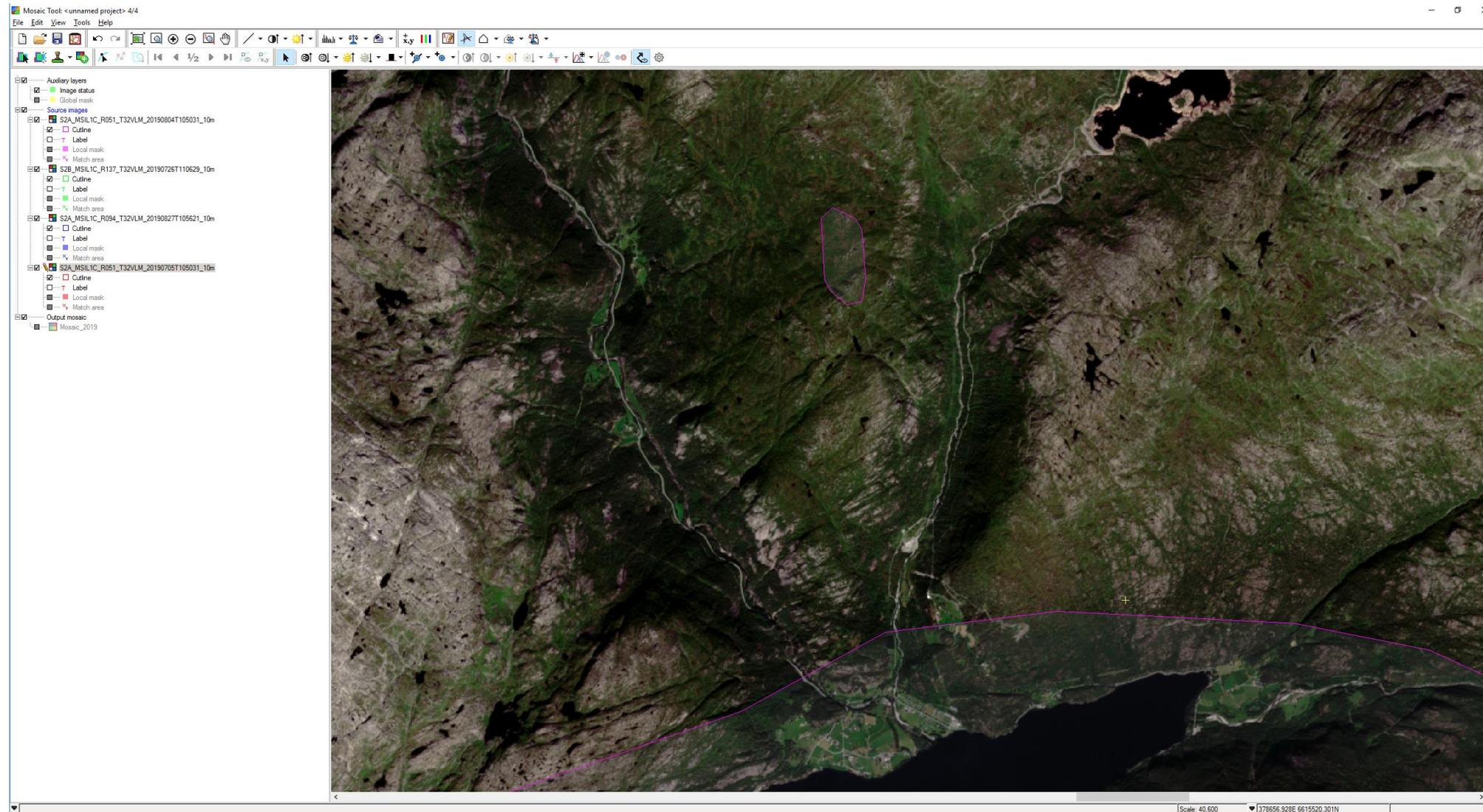
- Import cloud-free vectors for each file



6) Mosaicing

PCI Geomatica: Mosaic Tool

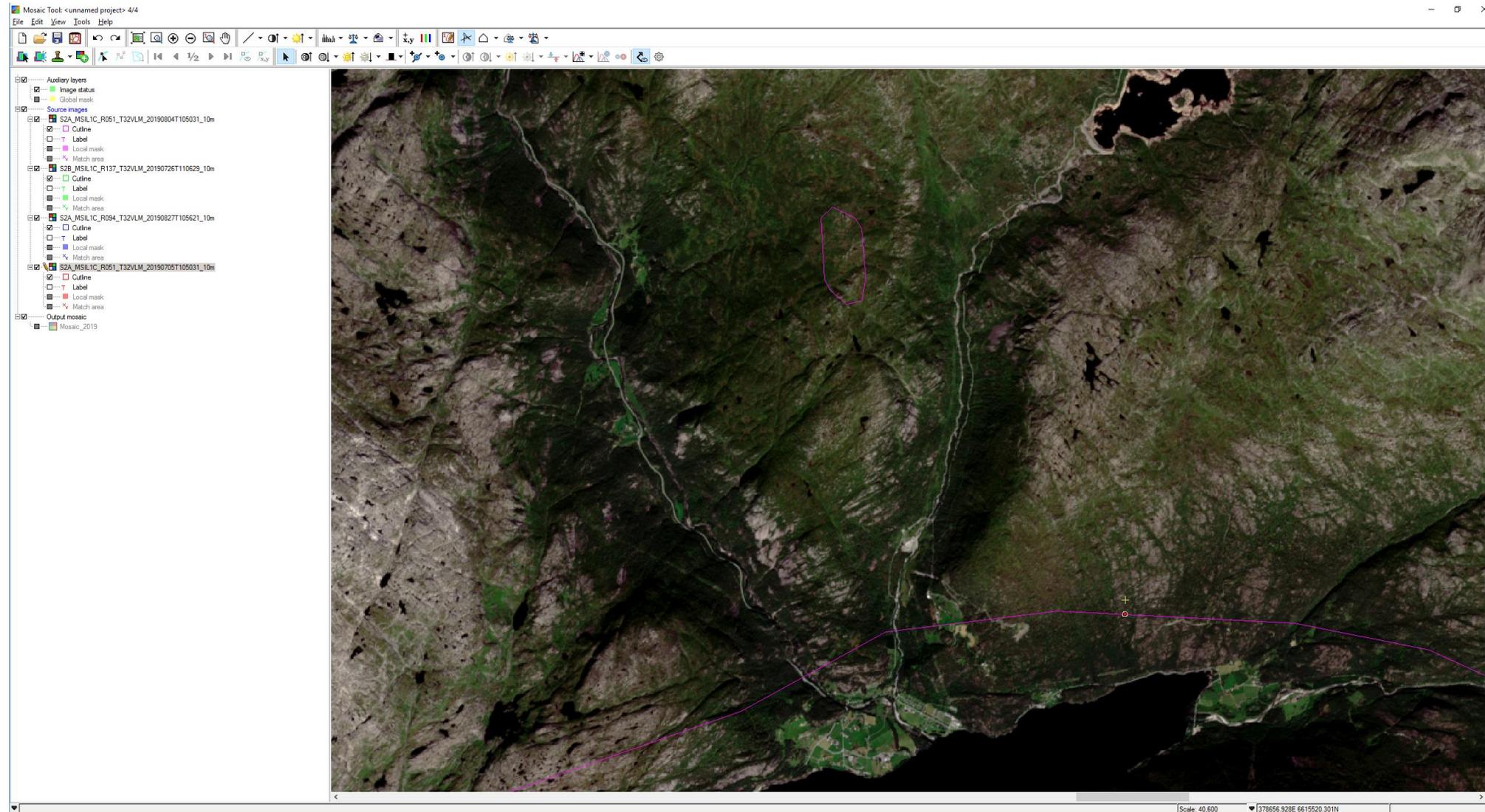
- Dogding



6) Mosaicing

PCI Geomatica: Mosaic Tool

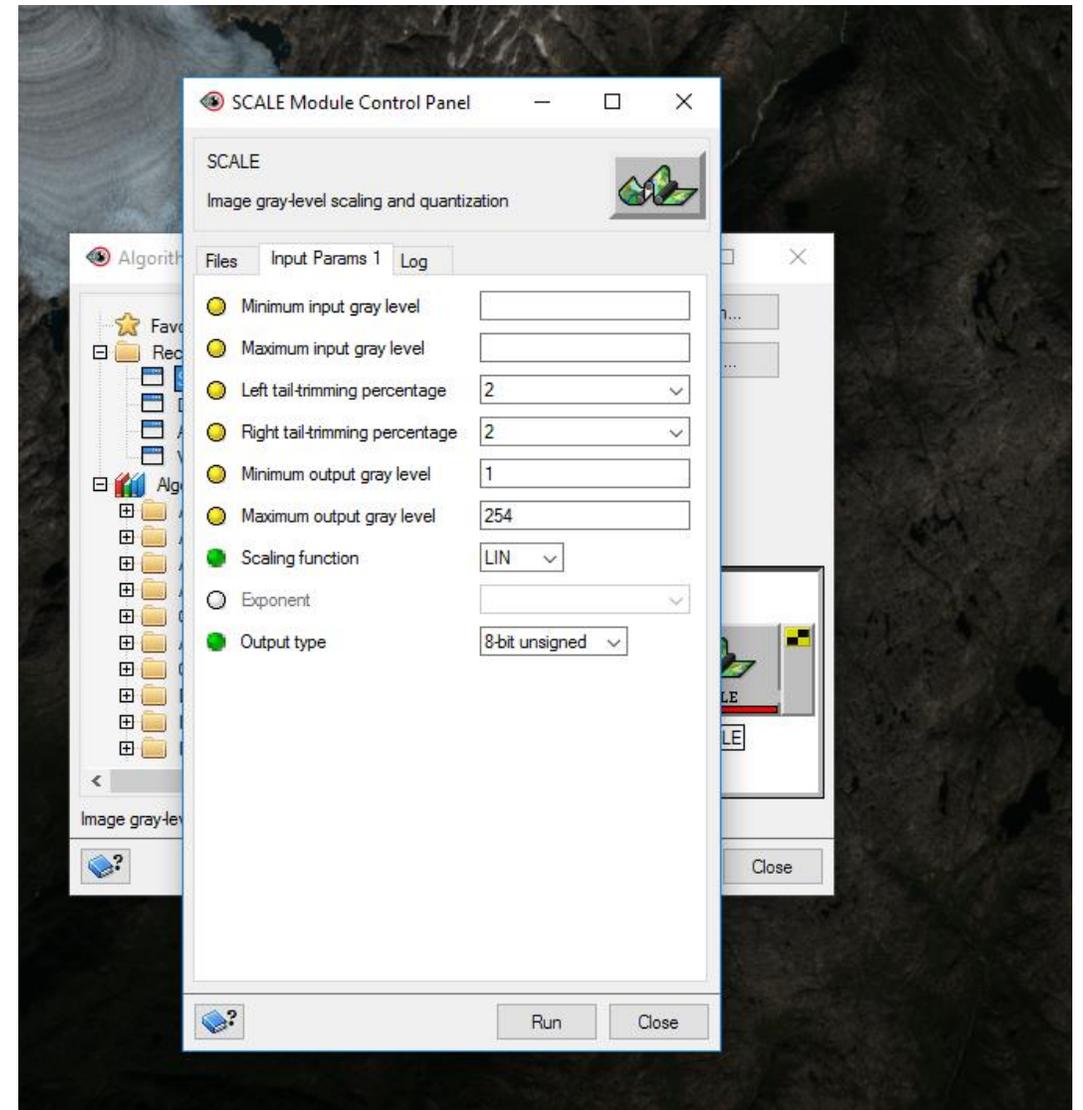
- Dogding

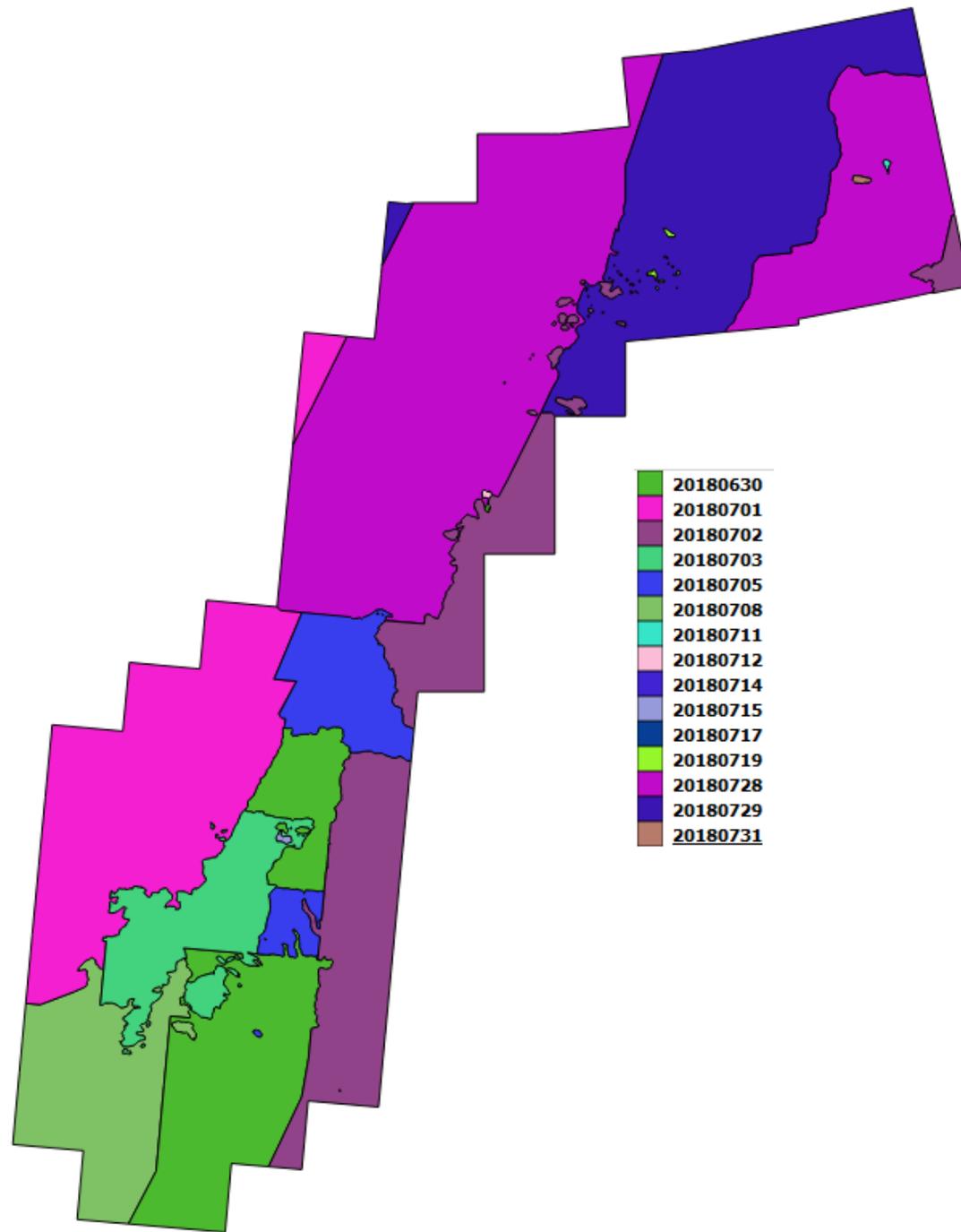


7) Scale

PCI Geomatica: Focus

- Due to 8-bit limitation in “Norway in Images” we scale the mosaic from Uint 16 to Uint 8
- Min value 1 and max value 254
 - Then we don't risk transparency pixels (0,0,0 and 255,255,255)

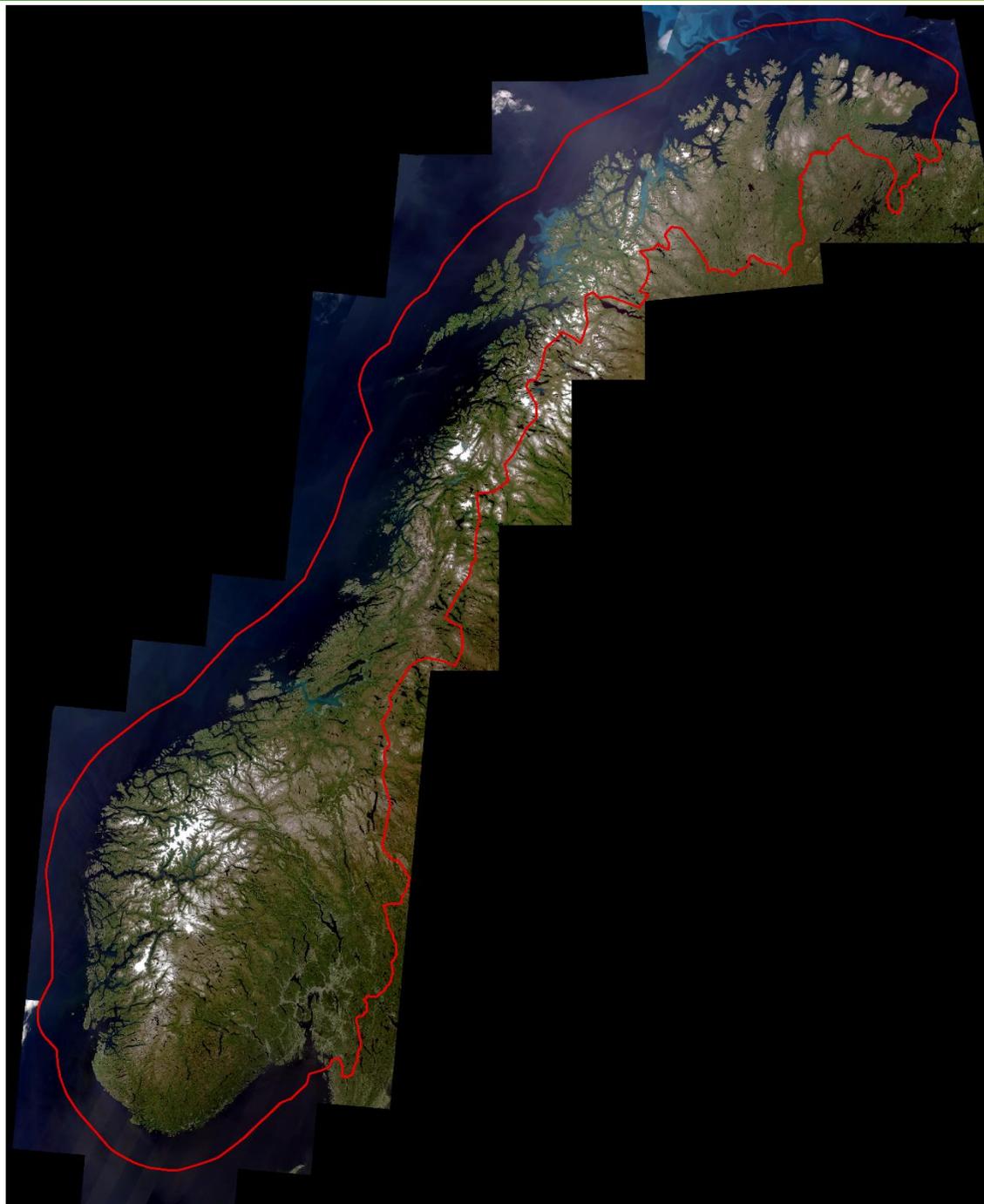




- Download at Geonorge.no

- [RGB 8-bit](#)

- [10 bands 16-bit](#)



2018 version

Thank you for your attention

Torgeir.Ferdinand.Klingenberg@kartverket.no

www.kartverket.no

