SINERGISE sentinelhub TOMTOM meteoblue Technical University of Munich SatCen

Global Earth Monitor

Deliverable 4.2

Cloud Masking

The project has received funding from European Union's Horizon 2020 Research and Innovation Programme" under the Grant Agreement 101004112

PREPARATION SLIP			
	Name	**Organisation**	**Date**
Prepared by	Mariza Pertovt	Sinergise	12.04.2021
Reviewed by	Matej Batič	Sinergise	26.04.2021
Approved for submission by	Matej Batič	Sinergise	29.04.2021

EXECUTIVE SUMMARY

As part of the Machine Learning Package (WP4), this deliverable (D4.2: Cloud masking) demonstrates the development and implementation of cloud masking algorithms that were included in eo-learn to enable pseudo-probability cloud masking of Sentinel-2 data. The cloud masking technique is integrated into the Sentinel Hub's pre-processing chains and made readily available for the complete Sentinel-2 archive.

We have upgraded the eo-learn library with two cloud masking algorithms: s2cloudless for single-observation cloud masking and InterSSIM for multi-temporal cloud masking. The deliverable also shows how the s2cloudless model is used to produce cloud masks and accompanying cloud mask pseudo-probabilities accessible directly through Sentinel Hub service. This gives users a more streamlined access to the data, directly suitable for value added services, since they do not have to deal with cloudy pixels anymore.

For a quick demonstration of the cloud masking capabilities, reader can go to EO-browser application using this link http://bit.ly/eobrowser.

Contractual Delivery Date	30.04.2021
Actual Delivery Date	30.04.2021
Type of delivery	Demonstrator
Dissemination level:	Public

Table of Contents

1 Introduction 3
2 Implementation of the project proposal 4
2.1 Broadening Sentinel Hub's capabilities 4
3 s2cloudless 5
3.1 Training and validation 5
3.2 Model selection and hyper-parameter tuning 6
3.3 s2cloudless algorithm 7
3.4 InterSSIM algorithm 7
3.5 Evaluation of s2cloudless at CMIX 8
4 Using s2cloudless 9
4.1 s2cloudless in eo-learn 9
4.1.1 Cloud masking with Sentinel Hub and s2cloudless 9
4.1.2 Use Sentinel Hub precomputed cloud masks and cloud probabilities 9
4.1.3 Computing cloud maps yourself 10
4.1.4 Multi-temporal cloud masks 11
4.1.5 Summary of cloud mask usage 12
4.2 s2cloudless data via Sentinel-Hub services 12
4.2.1 Cloud Masks and Cloud Probabilities 12
4.2.2 Alignment with s2cloudless 13
4.2.3 Demonstration of cloud masking capabilities in EO browser 13
4.3 s2cloudless external uses 14
5 Conclusion 15

List of Figures

Figure 1: True color, cloud probabilities and cloud masks from Sentinel Hub services. 10

Figure 2: True color, cloud probabilities from running s2cloudless locally. 11

Figure 3: True color, cloud probabilities from running multi-temporal s2cloudless masking locally. 12

Figure 4: A screenshot from EO Browser with a simple custom script for masking out the clouds using the cloud mask information from the service. 13

List of Abbreviations

aDC	Adjustable Data Cube
BOA	Bottom of the Atmosphere
CLM	Cloud Mask
CLP	Cloud Probability
CMIX	Cloud Masking Inter-Comparison Exercise
GEM	Global Earth Monitor
EO	Earth Observation
ML	Machine learning
EO data	Earth Observation data
ESA	European Space Agency
TOA	Top of the Atmosphere

1 Introduction

The Grant Agreement for the Global Earth Monitor (GEM) project (num. 101004112 – hereinafter referred to as the "Grant Agreement") defined the fourth work package (WP4) as Machine learning which encompasses:

- development and merging of new machine learning (ML) technologies and their introduction into eo-learn[1].
- creation of ML algorithms/models and their respective enveloping use-cases.

This document describes the second deliverable of WP4 that focuses on the Cloud masking algorithms, and the development of source code components required to enable seamless integration with Sentinel Hub adjustable Data Cube (aDC).

On Earth, satellite images used for remote sensing invariably contain visuals of clouds that impair the quality of data in subsequent analysis (e.g., crop monitoring in agriculture, land-use classification and change detection). Therefore, cloud detection is the most crucial step during the pre-processing of optical satellite images. Multiple algorithms are currently in use to identify pixels contaminated by clouds. Unfortunately, the existing cloud masking algorithms that work with Sentinel-2 images are either too complicated, expensive to run, or simply don't produce satisfactory results. They miss to identify clouds too often and like to identify clear sky over land as clouds. That is why we developed a fast ML-based algorithm that detects clouds in real-time and delivers state-of-the-art results in the class single-scene algorithms.

This document contains the following elements of information regarding GEM WP4.

Section 2 presents the implementation of the project proposal with the cloud masking algorithms that are included in eo-learn enabling pseudo-probability cloud masking of Sentinel data.

Section 3 describes the s2cloudless, the single and multi-temporal cloud masking algorithms and presents description of the s2cloudless model.

Section 4 demonstrates how to perform cloud masking using eo-learn and how to obtain pre-computed cloud masks through Sentinel-Hub. The section also provides information on external uses of s2cloudless.

[1] https://eo-learn.readthedocs.io/en/latest/

2 Implementation of the project proposal

eo-learn is a collection of open-source Python packages that we have devised to access seamlessly and automatically, and process spatial-temporal image sequences captured by any satellite fleet. The library is written in Python and uses NumPy arrays to store and handle remote sensing data.

With this GEM deliverable we have upgraded and extended the cloud masking EO-value-extraction workflow to eo-learn. eo-learn and its workflows are open source, free to use and open to community sharing and improvement. Adding new tasks to eo-learn is rather trivial, as the eo-learn collection has a modular design and is collaboration- friendly regarding sharing and reusing specific tasks in its workflows. The cloud masking algorithms that are now included in eo-learn enable pseudo-probability cloud masking of Sentinel data, using either a single Sentinel-2 image or a temporal sequence for multi-temporal cloud masking.

The eo-learn cloud masking tasks rely on the updated s2cloudless2 algorithm. s2cloudless is a Python based open-source tool and a part of sentinelhub-py3 and developed by Sinergise. With it, we have achieved superb accuracy (99,4%) when classifying clouds in Sentinel data. The tool ranks among the top three in a Cloud Masking Inter-Comparison Exercise (CMIX) conducted by European Space Agency (ESA).

2.1 Broadening Sentinel Hub's capabilities

Sentinel-2 data, accessed through Sentinel Hub is now further expanded with cloud masking capabilities using the s2cloudless algorithm. The cloud masks can be retrieved for both Level-1C (Top of the Atmosphere (TOA) reflectance data) and Level-2A (Bottom of the Atmosphere (BOA) reflectance, processed from Level-1C data with Sen2Cor[4] processor).

Along the nominal Sentinel-2 reflectance data and observation metadata, Sentinel Hub provides following per-pixel values:

Name	Description	Resolution
CLP	Cloud probability	160m
CLM	Cloud masks (more)	160m

In addition to sections 4.1.2 and 4.2, the link[5] provides more info about cloud masks available through Sentinel Hub and how to use them.

With this we have scaled up the Sentinel Hub service capabilities to be able to deliver a global "meta-data layer" about clouds. The layer has pseudo-probability on a pixel level so that this can be used in ML processes not only as a cloud mask but as also as "de-facto cloud data". Pseudo-probabilities empower users to also

[2] https://github.com/sentinel-hub/sentinel2-cloud-detector
[3] https://sentinelhub-py.readthedocs.io/en/latest/
[4] https://step.esa.int/main/snap-supported-plugins/sen2cor/
[5] https://docs.sentinel-hub.com/api/latest/user-guides/cloud-masks/

model data in "hazy" areas or use them to generate TRUE/FALSE cloud masks based on user-defined thresholds. The cloud (probability) layer can also be used as a meta-layer describing data quality, which can be further employed in uncertainty modelling.

3 s2cloudless

We have developed a single-scene, pixel-based cloud detection algorithm that works on a global scale and heavily relies on machine learning techniques. We decided to take the simplest approach, using the so-called pixel-based classification. We assigned each image pixel a probability that it is being covered with a cloud solely based on satellite's spectral response for that pixel.

Calculation of a pixel's cloud probability therefore doesn't depend on its neighborhood. We only take the broader context (pixel's neighborhood) into account when we construct the cloud mask of a given scene from its cloud probability map by performing so called morphological operations, such as convolution.

We believe that machine learning algorithms which can consider the context such as semantic segmentation using convolutional neural networks will ultimately achieve the best results among single-scene cloud detection algorithms. However, due to the high computational complexity of such models and related costs they are not yet used in large scale production. We decided to develop first the best possible pixel-based cloud detector and see how it compares to other existing algorithms.

Following section will provide additional information about the training, validation, and optimization of the s2cloudless model. The two algorithms, s2cloudless for single scene cloud masking and InterSSIM for multi-temporal cloud masking are also presented. Further resources regarding the cloud detector could be found at the s2cloudless repository: https://github.com/sentinel-hub/sentinel2-cloud-detector.

3.1 Training and validation

The success of any machine learning application depends heavily not only on the quality and size of the training data but also on the usage of an appropriate validation set. For the latter, we chose the publicly available data set of manually labelled pixels from Sentinel-2 scenes from Hollstein et al.[6] sampled roughly evenly from around the globe and throughout the year.

Regarding our training data, we had to be inventive. Not having an extensive high quality-labelled data set that we could use as our training-data sample, we turned to a second-best option - to use the best currently available cloud detector and its cloud masks as a proxy for ground truth. We chose the MAJA[7] multi-temporal processor since it has, based on our experiences, a high cloud detection rate and a very low misclassification rate of no cloudy pixels.

Usage of a machine-labelled data set unavoidably introduces noise in labels which is not ideal, but on the other hand, a machine curated data set can be produced very fast with no or very little costs. Our training sample

[6] https://bit.ly/3aL4FB7
[7] https://bit.ly/3eze1RI

extracted from MAJA at the end consisted of around 14 million pixels, 47% of them being classified by MAJA as cloudy pixels.

3.2 Model selection and hyper-parameter tuning

Once we had the training and validation sets in place, we experimented with input features, models, hyper-parameters, and other things to see what works best.

To summarize, we have checked the following:

- Feature space:
    - raw band values (Bi),
    - pairwise band differences (Bi -Bj),
    - pairwise band ratios (Bi/Bj), and
    - pairwise band differences over their sums (Bi-Bj/Bi+Bj).

We didn't observe significant improvement when using derived features instead of raw band values. Our final classifier uses the following 10 bands as input: B01, B02, B04, B05, B08, B8A, B09, B10, B11, B12.

- Models:
    - decision trees,
    - support vector machines,
    - Bayes tree,
    - gradient boosting of tree-based learning algorithms with XGBoost and LightGBM, and
    - neural networks with fastai/PyTorch.

Neural net gives the best accuracy but is considerably slower at inference time. The second-best results were achieved with XGBoost and LightGBM. We opted for the latter since it is faster during training and inference.

- Hyper-parameters:
  We did some hyper parameter tuning for all different models. The tuned hyper-parameters of selected model (LightGBM) are determined to be:
    - min_data_in_leaf=1500,
    - n_estimators=170, and num_leaves=770.
    - All others are set to their default values.

3.3 s2cloudless algorithm

The s2cloudless algorithm is mono temporal (uses single Sentinel-2 observation to classify clouds), does not take any spatial context into account (it is pixel based), and can be executed at any resolution. s2cloudless can, unlike many other algorithms, be also executed on averaged Sentinel-2 reflectance values over arbitrary user-defined geometries and still provide meaningful results.

The input features are Sentinel-2 reflectance values of the following ten bands: *B01, B02, B04, B05, B08, B8A, B09, B10, B11, B12*.

The output of the algorithm is a cloud probability map. Users of the algorithm can convert the cloud probability map to a cloud mask by setting a threshold on the cloud probability map. The recommended value for the threshold is 0.4.

Users can optionally apply additional morphological operations during the conversion of the cloud probability map to the cloud mask. These operations are convolution of the probability map with a disk and dilation of the binary cloud mask with a disk.

We recommend:

- to convolve cloud probability maps at 10 m (160 m) resolution,
- with a disk with a radius of 22 (2) and
- dilate cloud masks with a disk with radius 11 (1).

3.4 InterSSIM algorithm

The InterSSIM cloud detection algorithm is a *multi-temporal extension of the s2cloudless algorithm*[8] which is also based on the gradient boosting algorithm LightGBM. The latter was as well trained on a training dataset with global coverage.

Unlike s2cloudless, the InterSSIM algorithm takes temporal and spatial context into account. The input features are:

- Sentinel-2 reflectance values of the same 10 bands from the target time frame;
- spatially averaged reflectance values for the target frame using a Gaussian filter;
- minimum and mean reflectance values of all available time frames at a given spatial coordinate;
- maximum and mean differences of reflectance values between the target frame and any other time frame;
- maximum, mean, and standard deviation of structural similarity indices computed between the target and every other time frame.

[8] https://bit.ly/3vrCdMm

Additionally, if not explicitly provided, s2cloudless probabilities for each time frame are computed from reflectance values and used as inputs as well. Both algorithms have been integrated into the eo-learn Python library, published under the MIT License on GitHub[9].

The output of the algorithm is a cloud probability map for the target time frame, which can be converted into a cloud mask with the same procedure as in the case of s2cloudless algorithm.

3.5 Evaluation of s2cloudless at CMIX

For the purpose of CMIX, ESA used the traditional approach of defining cloud masking: cloud mask is an absolute indication on cloudiness with a binary mask for different levels of cloudy/clear. Cloud Masking is then a support task to identify pixels at TOA signal level and which are suitable for subsequent processing (e.g., cloud, atmosphere, land, or marine applications). The objective of the CMIX exercise was to inter-compare a set of cloud-detection algorithms for space-borne high-spatial-resolution (10 m-30 m) optical sensors, focusing on Landsat-8 and Sentinel-2 missions. There were 10 cloud masking processors in the CMIX altogether (8 other processors beside the s2cloudless and InterSSIM).

s2cloudless and InterSSIM algorithms scored among the top three cloud masking algorithms, confirming that our cloud masking provides state-of-the-art results.

[9] https://bit.ly/32URZn4

4 Using s2cloudless

The s2cloudless repository comes with installation instructions as well as with a Jupyter Notebook on how to use the cloud detector to produce a cloud mask or cloud probability.

In following chapters, we show how (s2cloudless) cloud masks and cloud probabilities can be used within eo-learn and how they can be retrieved or accessed directly through Sentinel Hub.

4.1 s2cloudless in eo-learn

4.1.1 Cloud masking with Sentinel Hub and s2cloudless

This Notebook illustrates how to get cloud masks from Sentinel Hub services, or compute them locally using the CloudMaskTask task from the eolearn.mask module.

```
%matplotlib inline

import datetime as datetime
import numpy as np
import matplotlib.pyplot as plt

from sentinelhub import BBox, CRS, DataCollection

from eolearn.core import FeatureType, LinearWorkflow
from eolearn.io import SentinelHubInputTask
from eolearn.mask import CloudMaskTask
```

We must also define ROI BBOX and time interval, in this case a bounding box around Ljubljana, February - April 2020:

```
roi_bbox = BBox([14.218369, 46.007455, 14.71344, 46.271275], crs=CRS.WGS84)

time_interval = ('2020-02-01', '2020-05-01')
```

4.1.2 Use Sentinel Hub precomputed cloud masks and cloud probabilities

As an exercise, we can download RGB Sentinel-2 bands together with cloud mask (CLM) and cloud probabilities (CLP) bands.

```
input_task = SentinelHubInputTask(
    data_collection=DataCollection.SENTINEL2_L1C,
    bands=['B02','B03','B04'],
    bands_feature=(FeatureType.DATA, 'trueColorBands'),
    additional_data=[(FeatureType.MASK, 'dataMask'),
                     (FeatureType.MASK, 'CLM'),
                     (FeatureType.DATA, 'CLP')],
    resolution=(160, 160),
    time_difference=datetime.timedelta(hours=2),
    max_threads=3
)
```

```
%%time
eopatch = input_task.execute(bbox=roi_bbox, time_interval=time_interval)
CPU times: user 527 ms, sys: 96.4 ms, total: 623 ms
Wall time: 5.29 s
```

```
timestamp_idx = 11

fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(20,6))
ax[0].imshow(eopatch.data['trueColorBands'][timestamp_idx][...,[2,1,0]])
ax[1].imshow(np.squeeze(eopatch.data['CLP'][timestamp_idx]), cmap='gray')
ax[2].imshow(np.squeeze(eopatch.mask['CLM'][timestamp_idx]), cmap='gray')

ax[1].set_title(f'True color, cloud probabilities and cloud masks for {eopatch.timestamp[
timestamp_idx].date().isoformat()} from Sentinel Hub service');
```

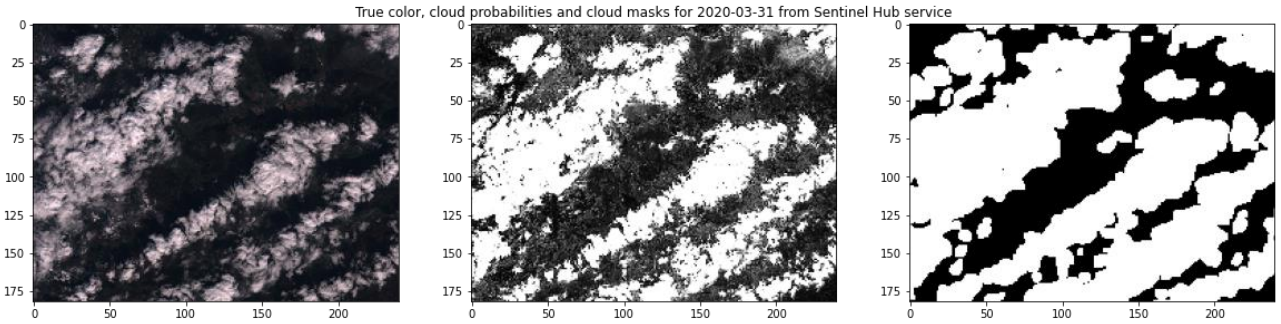

Figure 1: True color, cloud probabilities and cloud masks from Sentinel Hub services.

4.1.3 Computing cloud maps yourself

Sometimes, we want to compute cloud maps locally. In this workflow example, we do so by requesting appropriate bands from service and running s2cloudless.

```
bands_task = SentinelHubInputTask(
    data_collection=DataCollection.SENTINEL2_L1C,
    bands=['B01','B02','B04','B05','B08','B8A','B09','B10','B11','B12'],
    bands_feature=(FeatureType.DATA, 'bands'),
    additional_data=[(FeatureType.MASK, 'dataMask')],
    resolution=(160, 160),
    time_difference=datetime.timedelta(hours=2),
    max_threads=3)

add_clm = CloudMaskTask(
    data_feature='bands',
    all_bands=False,
    is_data_feature='dataMask',
    mono_features=('CLP', 'CLM'),
    mask_feature=None,
    processing_resolution='160m')
```

To continue, run the whole workflow; download bands and execute CloudMaskTask:

```
%%time
workflow = LinearWorkflow((bands_task, 'add data'), (add_clm, 'compute cloud mask'))
result = workflow.execute({bands_task: {'bbox': roi_bbox, 'time_interval': time_interval}
})
eopatch_clm = result[add_clm]
```

```
CPU times: user 51.6 s, sys: 387 ms, total: 52 s
Wall time: 11.6 s
```

```
timestamp_idx = 11

fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(20,6))
ax[0].imshow(eopatch_clm.data['bands'][timestamp_idx][...,[2,1,0]])
ax[1].imshow(np.squeeze(eopatch_clm.data['CLP'][timestamp_idx]), cmap='gray')
ax[2].imshow(np.squeeze(eopatch_clm.mask['CLM'][timestamp_idx]), cmap='gray')

ax[1].set_title(f'True color, cloud probabilities and cloud masks for {eopatch.timestamp[
timestamp_idx].date().isoformat()} from running s2cloudless locally');
```

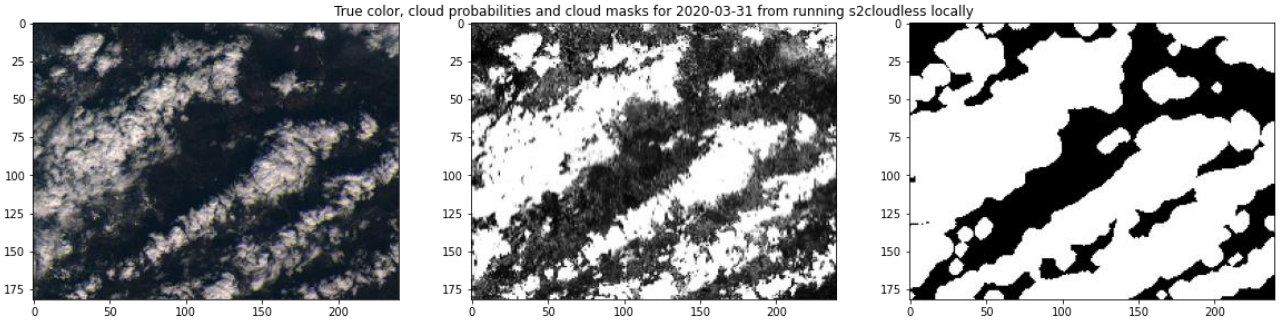

Figure 2: True color, cloud probabilities from running s2cloudless locally.

4.1.4 Multi-temporal cloud masks

The CloudMaskTask also supports multi-temporal cloud masking, taking several observations into account when calculating masks for any given observation. As we have already all the data available, we'll just create the task and execute it on the eopatch_clm. For possible comparison with mono-temporal masks, we'll append the multi-temporal cloud masks and pseudoprobabilities with _multi.

```
add_multitemporal_clm = CloudMaskTask(
    data_feature='bands',
    all_bands=False,
    is_data_feature='dataMask',
    multi_features=('CLP_multi', 'CLM_multi'),
    mask_feature=None,
    processing_resolution='160m')
```

```
%%time
eopatch_multi = add_multitemporal_clm.execute(eopatch_clm)
```

```
CPU times: user 2min 31s, sys: 3.44 s, total: 2min 34s
Wall time: 14 s
```

```
timestamp_idx = 11

fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(20,6))
ax[0].imshow(eopatch_clm.data['bands'][timestamp_idx][...,[2,1,0]])
ax[1].imshow(np.squeeze(eopatch_clm.data['CLP_multi'][timestamp_idx]), cmap='gray')
ax[2].imshow(np.squeeze(eopatch_clm.mask['CLM_multi'][timestamp_idx]), cmap='gray')
```

```
ax[1].set_title(f'True color, cloud probabilities and cloud masks for {eopatch.timestamp[
timestamp_idx].date().isoformat()} from running multi-temporal s2cloudless masking locall
y');
```

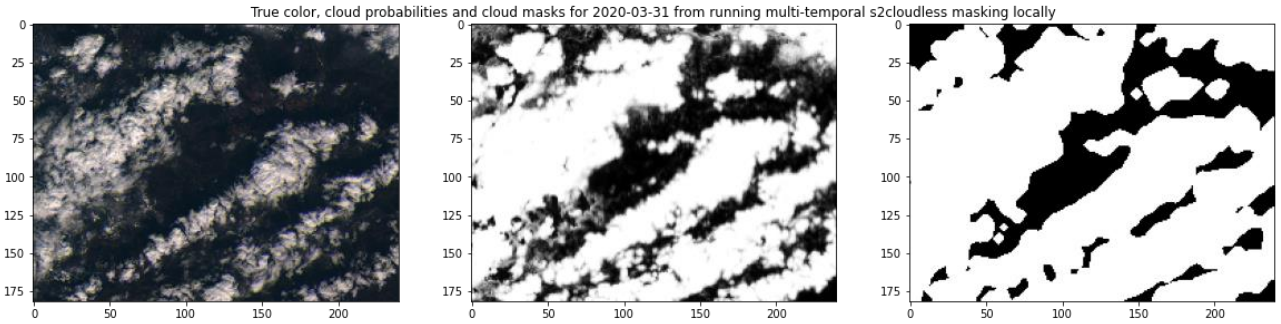

Figure 3: True color, cloud probabilities from running multi-temporal s2cloudless masking locally.

4.1.5 Summary of cloud mask usage

Using cloud masks provided by the service suffices for most use cases, driving down the cost of your processing while being faster. You might have noticed the `%%time` magic command used throughout the Notebook; in our case, the comparison of the three approaches shows:

- Sentinel Hub services:
  `CPU times: user 527 ms, sys: 96.4 ms, total: 623 ms`
- mono-temporal cloud mask
  `CPU times: user 51.6 s, sys: 387 ms, total: 52 s`
- multi-temporal cloud masking
  `CPU times: user 2min 31s, sys: 3.44 s, total: 2min 34s`

Unless Sentinel Hub provided cloud masks are not sufficient in your use case, we strongly encourage to use them. Here you can find the blog post for details.

4.2 s2cloudless data via Sentinel-Hub services

4.2.1 Cloud Masks and Cloud Probabilities

Cloud masks and probabilities computed using s2cloudless are available at a fixed resolution of 160m per pixel. Sentinel Hub implements the 10-band version. These are meant as convenience bands with the purpose of speeding up processing. Cloud masks are generated in a very slightly different way than the implementation above but for most applications this should not matter.

They are available as Sentinel-2 bands named CLP (cloud probabilities) and CLM (cloud masks) and have the following return values:

- CLM: 0 (no clouds), 1 (clouds), 255 (no data);
- CLP: 0–255 (divide by 255 to get to the [0-1] range);

The CLM no data value of 255 is also returned if a tile has missing CLM and CLP bands, for example due to errors. This ensures that values of 0 and 1 can be used with confidence for each pixel. CLP will in such a case return 0. Consider using CLM alongside CLP in your evalscript if this is an issue.

4.2.2 Alignment with s2cloudless

CLP is generated per tile using the s2cloudless product at 160m resolution. Due to the 60m Sentinel-2 bands this means that a perfect match between CLP and s2cloudless is not possible for all requests. In case you require identical data, there are a few constraints that must be met. These are:

- requesting data in the native UTM zone of each tile
- nearest neighbor interpolation
- 160m resolution or slightly less (more zoomed out)
- the request origin is a multiple of 480m away from the tile origin (the top-left point of the source tile)
- requesting a single tile only; no mosaicking (mosaicking violates the previous point)

If any of these are not met, you can expect slight differences. For exact values the s2cloudless product may be used without these constraints, at the cost of requiring more processing time and processing units; for most applications, however, we do not expect this to be necessary.

4.2.3 Demonstration of cloud masking capabilities in EO browser

A concrete example of demonstrating capabilities is shown here: http://bit.ly/eobrowser, where users can see the cloud masking in action browsing the whole Sentinel-2 archive.

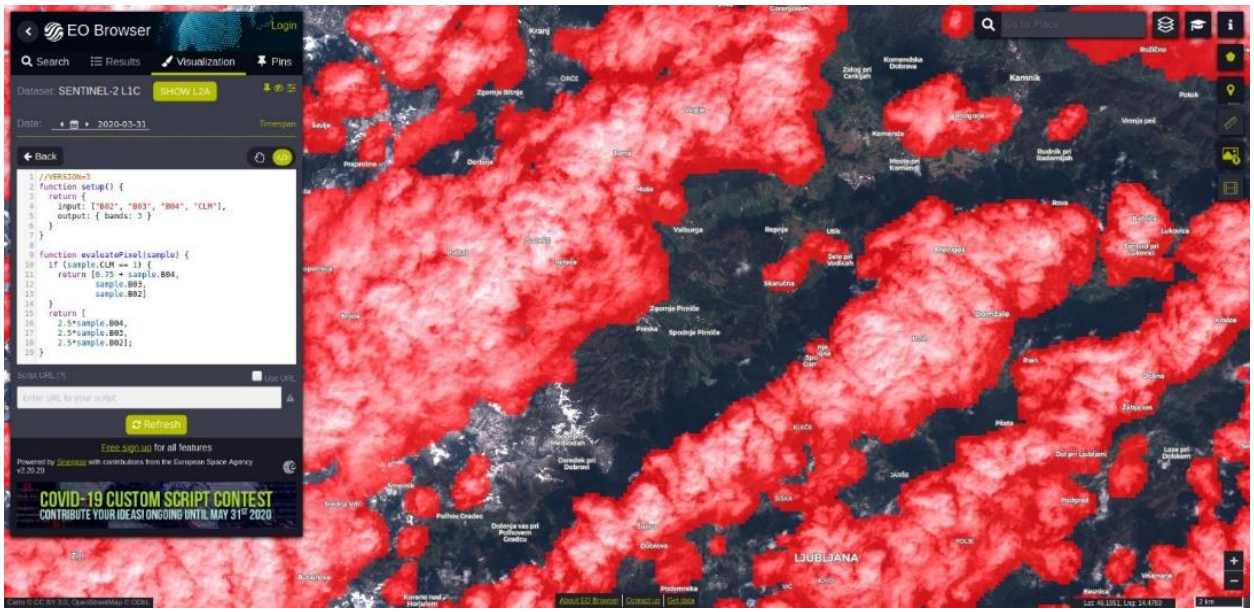

Figure 4: A screenshot from EO Browser with a simple custom script for masking out the clouds using the cloud mask information from the service.

Sentinel-Hub provides documentation on cloud masks and cloud probabilities. Additionally, reader might be interested in the blog posts found here: Cloud Masks at Your Service.

4.3 s2cloudless external uses

Sentinel-2 Cloud Masking with s2cloudless is being used also by Google Earth Engine[10]. The blog post[11] also describes how by uniting our algorithm with Google's computing resources, Google has calculated per-pixel cloud probability for the entire Sentinel-2 archive at 10 m scale; each new image added to the Earth Engine catalog is also accompanied by an s2cloudless image. The authors of the blog post write about the algorithm:

The s2cloudless dataset provides a flexible method to accurately mask cloudy pixels in Level 1C (TOA) and 2A (SR) imagery for generating cloud-free composites and running classification procedures.

[10] https://developers.google.com/earth-engine/tutorials/community/sentinel-2-s2cloudless
[11] https://medium.com/google-earth/more-accurate-and-flexible-cloud-masking-for-sentinel-2-images-766897a9ba5f

5 Conclusion

A machine learning approach to cloud masking can give state-of-the-art results if the training and validation sets are both of good enough quality and representative of the unseen data. Procurement of labelled samples in remote sensing suitable for the development of models that perform well on a global scale was a particular challenge for us. We're fortunate to be living on a planet with such versatile landscapes. On the other hand, this versatility is a curse for machine learning. Models based on machine learning have, however, the ability to constantly evolve and improve.

Our s2cloudless demonstrator that we developed is a single-scene cloud detection algorithm predominantly based on machine learning techniques. We decided to on the simple solution - implementing specifically pixel-based classification. We assigned each image pixel a probability of it being covered with a cloud solely based on the satellite's spectral response for that pixel.

In the ESA comparison exercise, our cloud detector yielded a higher cloud detection rate while at the same time having a much lower misclassification rate of land and snow as clouds. This was also the first comparison of the most popular cloud masking algorithms on a large human-labeled data set of Sentinel-2 imagery to the best of our knowledge. The s2cloudless is now used within Sentinel Hub to provide pre-computed cloud masks, and similarly in Google Earth Engine.