ESA Sen4Stat

Sentinels for Agricultural Statistics

D7.0 – DDF - ATBD
"Algorithm Theoretical Basis Document: Crop Mapping"

Issue/Rev : 1.0

Date Issued : 30-08-21

Milestone: 2

Submitted to European Space Agency

Consortium Partners

Participant Organisation Name	Acronym	City, Country
Université catholique de Louvain	UCLouvain	Louvain-la-Neuve, Belgium
CS GROUP - Romania	CSG RO	Craiova, Romania
Systèmes d'Information à Référence Spatiale - Filiale CLS	SIRS-CLS	Villeneuve d'Ascq, France
Universidad Polytecnica de Madrid	UPM	Madrid, Spain

Contact

Université catholique de Louvain – Earth and Life Institute
Place de l'Université, 1 – B-1348 Louvain-la-Neuve – Belgium
Email : Sophie.Bontemps@uclouvain.be
Internet : https://uclouvain.be/en/research-institutes/eli/elie

Document sheet

Authors and Distribution

Authors	Nicolas DEFFENSE, Cosmin CARA, Laurentiu NICOLA, Cosmin UDROIU, Sophie BONTEMPS
Distribution	ESA - Benjamin KOETZ, Zoltan SZANTOI

Document Status

Issue/Rev.	Date	Reason
0.1	18/05/2021	Initial version of the deliverable
0.2	16/06/2021	Internal review of the deliverable
1.0	30/08/2021	1st version related to the Acceptance Review

Detailed record sheet

From version 1.0 to 1.x

Comment	Section	Problem description	Change

Table of contents

List of figures

List of tables

List of codes

Acronyms

Acronym	Definition
AD	Applicable Document
AOI	Area Of Interest
ATBD	Algorithm Theoretical Basis Document
BOA	Bottom-Of-Atmosphere
DDF	Design Definition File
EO	Earth Observation
ESA	European Space Agency
fAPAR	fraction of Absorbed Photosynthetically Active Radiation
fCOVER	fraction of Vegetation Cover
IW	Interferometric Wide Swath
L1, L2, L2A	Level 1, Level 2, Level 2A
L8	Landsat 8
LAI	Leaf Area Index
LUT	Look-Up Table
NDVI	Normalized Difference Vegetation Index
NDWI	Normalized Difference Water Index
NIR	Near InfraRed
NSO	National Statistical Office
RF	Random Forest
S1, S2	Sentinel-1, Sentinel-2
SAR	Synthetic Aperture Radar
Sen2-Agri	Sentinel-2 for Agriculture
Sen4CAP	Sentinels for Common Agricultural Policy
Sen4Stat	Sentinels for Agricultural Statistics
SLC	Single Look Complex
SMOTE	Synthetic Minority Over-Sampling Technique
SWIR	ShortWave InfraRed

1 Logical model and processor overview

The developed classification processor has been designed to generate various crop mapping products (i.e. crop map with different legends), following a per-pixel approach and relying on machine learning and/or deep learning algorithms. Two types of data are used to feed the classification algorithms: *in situ* data provided by the National Statistical Offices (NSOs) (and pre-processed by the "*In situ* Data Preparation" processor) and Earth Observation (EO) data.

Figure 1-1 presents the general workflow of the mapping products processing chain. There are 5 main components detailed in the following sections of this document:

1. *In situ* data sample selection;
2. Optical EO data temporal resampling & gap filling;
3. EO features computation & extraction:
    a. Optical data;
    b. SAR data;
    c. Biophysical indicators;
    d. Spectral indices;
    e. Composite;
4. Pixel-based classification;
5. Validation.

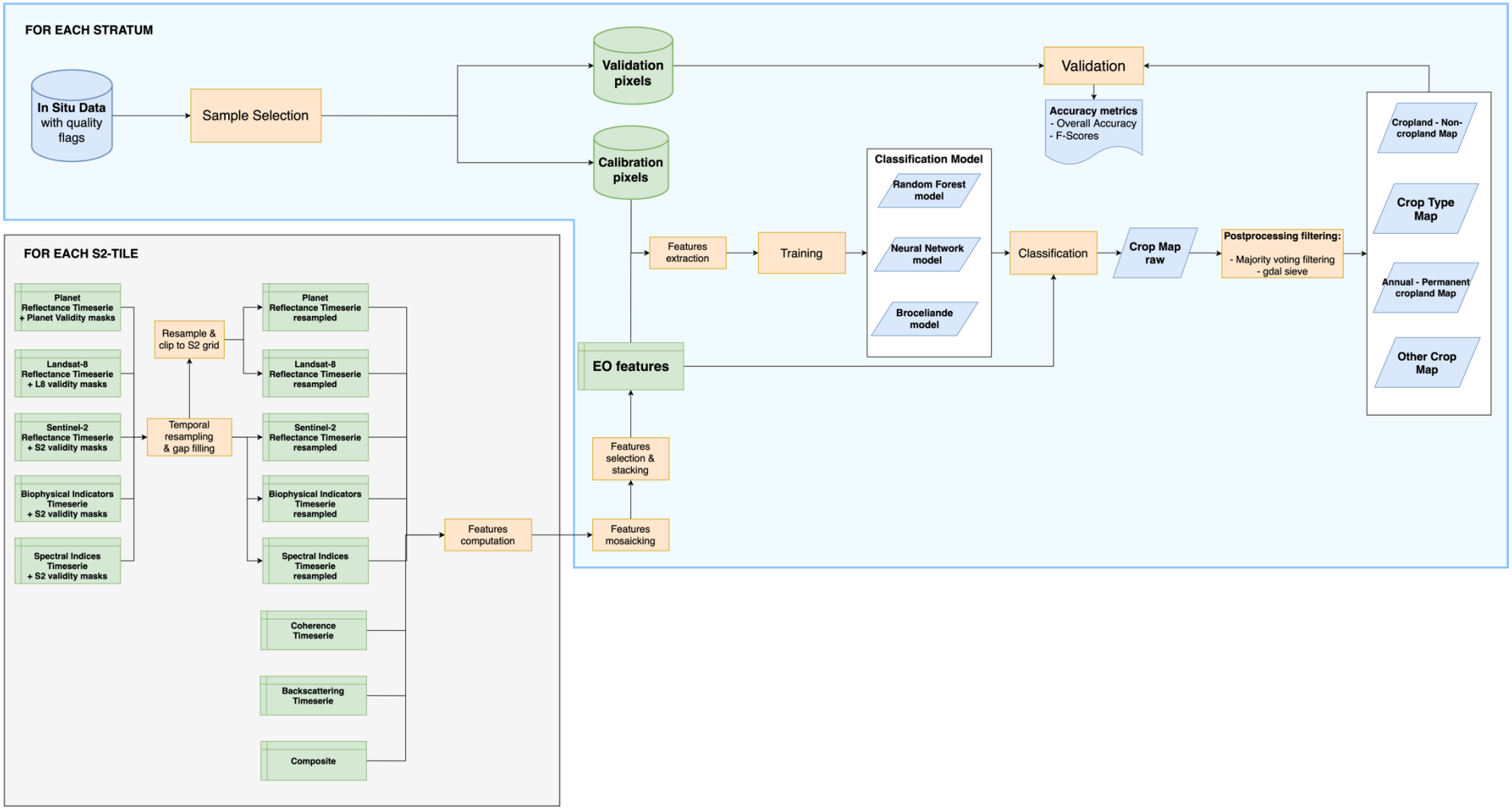


Figure 1-1. Workflow of the Crop Mapping processor

2 Input data

2.1 Standardized Sen4Stat *in situ* PostGIS tables

To ensure a certain level of consistency between the different Sen4Stat high-level processors (crop mapping, yield estimation and segmentation), the preparation of the NSO *in situ* data is performed prior to their execution, by a specific processor named "*In situ* Data Preparation" which is fully described in a dedicated ATBD.

The "Crop Mapping" processor uses the main output of this "*In situ* Data Preparation" processor, which are three standardized Sen4Stat PostGIS tables (Figure 2-1):

- "**in_situ_data**" table with all the statistical information about each crop;
- "**in_situ_polygons**" table with parcels geometry;
- **"polygon_attributes"** table with geometry flags, S2 pixels numbers, quality control and stratum flags of each parcel.

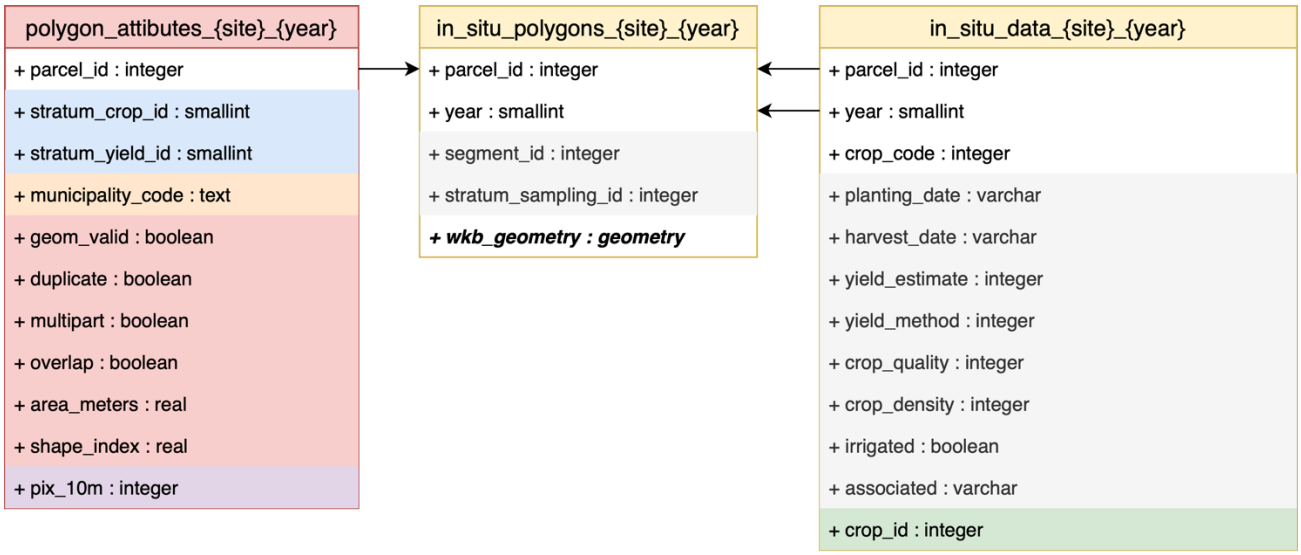


Figure 2-1. Output data of the "*In situ* Data Preparation" processor

2.2 Optical data

Optical data can be provided by several sensors: Sentinel-2 (S2), Landsat-8 (L8) and Planet. The inclusion of Planet data is currently only an option for the latter versions of the Sen4Stat system and this sensor will not be mentioned anymore in the following parts of the document.

Before being use by the "Crop Mapping" processor, these data have been pre-processed by the "EO Data Pre-Processing" processor (Table 2-1):

- an atmospheric correction has been applied to get the Bottom-Of-Atmosphere (BOA) reflectance - this is not needed for S2 if the user decides to rely on the S2 Level 2A (L2A) products proposed by

the European Space Agency SciHub. If the user wants to start from the S2 Level 1C (L1C) product and/or use L8 data, this step is mandatory;
- a validity mask has been generated to identify invalid pixels (clouds, cloud shadows, …).

Table 2-1. Output optical data of the "EO Data Pre-Processing" processor

Data	Description	Default value [format]
s2_b{xx}_ts_{tile}	Original S2 L2A surface reflectance time series; {xx} = band number; {tile} = S2 tile name	[GeoTIFF]
l8_b{xx}_ts_{tile}	Original L8 L2A surface reflectance time series; {xx} = band number; {tile} = L8 tile name	[GeoTIFF]
s2_validityMask_{tile}_FM	Validity mask for each S2 acquisition date obtained with Fmask; {tile} = S2 tile name	[GeoTIFF]
l8_validityMask_{tile}_FM	Validity mask for each L8 acquisition date obtained with Fmask; {tile} = L8 tile name	[GeoTIFF]

2.3 SAR data

Synthetic Aperture Radar (SAR) data are provided by Sentinel-1 (S1) satellite. Only Interferometric Wide Swath (IW) Single Look Complex (SLC) products are used. Two pre-processing chains have been applied on these products to get the sigma naught (or gamma naught) backscattering every 6 (or 12) days and the coherence at 6 (or 12) days (Table 2-2).

Table 2-2. Output SAR data of the "EO Data Pre-Processing" processor

Data	Description	Default value [format]
bck_ts_{tile}	Backscattering time series; {tile} = S2 tile name	[GeoTIFF]
cohe_ts_{tile}	Coherence time series; {tile} = S2 tile name	[GeoTIFF]

2.4 Biophysical indicators

Three biophysical indicators can be optionally used in the "Crop Mapping" processor:

- the Leaf Area Index (LAI);
- the fraction of Absorbed Photosynthetically Active Radiation (FAPAR);
- the fraction of Vegetation Cover (FCOVER).

The time series of these biophysical indicators are generated from the S2 BOA reflectance time series by the "Spectral Indices & Biophysical Indicators" processor (Table 2-3).

Table 2-3. Output biophysical indicators of the "Spectral Indices & Biophysical Indicators" processor

Data	Description	Default value [format]
{bv_name}_ts_{tile}	Original biophysical indicators time series; {bv_name} = LAI, FAPAR, FCOVER; {tile} = S2 tile name	[GeoTIFF]

2.5 Spectral indices

Three spectral indices can be optionally used in the "Crop Mapping" processor:

- the Normalized Difference Vegetation Index (NDVI);
- the Normalized Difference Water Index (NDWI);
- the brightness.

The time series of these spectral indices are generated from the S2 BOA reflectance time series by the "Spectral Indices & Biophysical Indicators" processor (Table 2-4).

Table 2-4. Output spectral indices of the "Spectral Indices & Biophysical Indicators" processor

Data	Description	Default value [format]
{spectral_name}_ts_{tile}	Original spectral indices time series; {spectral_name} = NDVI, NDWI, BRIGHT; {tile} = S2 tile name	[GeoTIFF]

2.6 S2 (or L8) cloud-free composites

Coming in the next version

3 Detailed workflow

3.1 *In situ* data sample selection

The 1st component of the "Crop Mapping" processor involves selecting the *in situ* data that will be used to calibrate the classifier and validate the final product. First, the polygons that can be used for the classification are selected. Then, among the polygons that fulfil all the criteria to be used for classification, the polygons that will be used to train the classification model and the ones that will be used to validate the produced crop mapping products are identified. Finally, the last step will move from the polygon-level to the pixel-level.

The output of this selection step is presented as a single table which has the same number of rows as the input standardized *in situ* statistical dataset with quality flags and with two additional columns specifying the *trajectory* (classified or not) and the *purpose* (calibration or validation) of each polygon.

The input and output data are presented in **Error! Reference source not found.**. The variables are presented in Table 3-2.

Table 3-1. Input and output data of *in situ* data sample selection

Input data	Description	Default value [format]
in_situ_data	Standardized Sen4Stat table with crop infromation	[PostGIS table]
in_situ_polygons	Standardized Sen4Stat table with polygon geographical data (parcel geometry)	[PostGIS table]
polygon_attributes	Standardized Sen4Stat table with polygon attributes	[PostGIS table]
Output data	**Description**	**Default value [format]**
{country}_{year}_CalPix	Pixels used to train the classification model	[PostGIS/SHP]
selected_polygons	Sen4Stat table with all selected polygons and sample selection strategy	[PostGIS table]

Table 3-2. Specific parameters for the *in situ* data sample selection

Parameters	Role	Default value [format]
monitored_land_cover	List of land cover categories that should be monitored	1,2,3,4,5,6,7,8,9 [integer]
monitored_crop	List of crops (sub-classes) that should be monitored	All sub-classes [integer]
pix_min	Minimum number of 10m pixels required for a polygon to be used for the classification	3 [integer]
pix_ratio_min	Minimum pixel_ratio by sub-class for the sub-class to be used for the classification; where pixel_ratio is the number of 10m pixels belonging to the sub-class divided by the total number of 10m pixels (see below)	0.0002 [float]
poly_min	Minimum number of polygons by sub-class for the sub-class to be used for the classification	10 [integer]

pix_best	Minimum number of 10m pixels in the polygon to be used for the calibration	10 [integer]
pix_ratio_hi	Threshold used to determine whether a sub-class belongs to strategy 1 or strategy 2	0.05 [float]
pix_ratio_lo	Threshold used to determine whether a sub-class belongs to strategy 2 or strategy 3	0.01 [float]
smote_ratio	Ratio used to know how many synthetic extra pixels should be generated with SMOTE	0.0075 [float]
sample_ratio_hi	Ratio of the remaining polygons by sub-class to be used for the calibration (if strategy 1)	0.25 [float]
sample_ratio_lo	Ratio of the remaining polygons by sub-class to be used for the calibration (if strategy 2 or 3)	0.75 [float]
Computed variables	**Role**	**Default value [format]**
polygon_num	Polygons count by crop type	- [integer]
crop_pixels	Total 10m pixels by crop type	- [integer]
total_pixels	Total 10m pixels of all crop types (trajectory = 0 and 1)	- [integer]
pixel_ratio	Number of 10m pixels belonging to specific crop type divided by the total number of 10m pixels $$pixel\ ratio = \frac{crop\ pixels}{total\ pixels}$$	- [float]
$strategy_i$	Strategy chosen for crop type i	- [integer] (1,2 or 3)
smote_pixels	Number of synthetic extra pixels that should be generated with SMOTE by crop type	- [integer]
$S2pixCAL_i$	Number of original pixels to select from the sub-class i to train the model	- [integer]

3.1.1 Polygons used for the classification

The first step consists in selecting the polygons that can be used for the classification (= eligible polygons), and conversely, in filtering out the polygons non-usable. For each crop type, the *pixel_ratio* is computed, which is defined as the number of 10m pixels belonging to a given crop type divided by the total number of 10m pixels, *total_pixels* (see Table 3-2), and the number of polygons by crop type, *polygon_num*. The decision about the eligibility of each polygon is taken looking at different aspects:

1) the polygon must have a valid geometry (*geom_valid* = True);
2) the polygon cannot have a multipart geometry (*multipart* = False);
3) the polygon cannot overlap with other polygons with more than 10% of its area (overlap = False);
4) the polygon must be of good quality (e.g. homogeneous) (*quality_control* = True) - **This step will be included in the next version**;

5) the polygon must contain at least '*pix_min'* 10m pixels (by default, *'pix_min'* = 3) given the fact that too small polygons could badly impact the classification performance;
6) the land cover (= code_n1) must be part of the land cover types listed in *monitored_land_cover*. By default, all the land cover categories are used for the classification but for some specific products it might be necessary to exclude some of them;
7) the crop code must be part of the crop codes listed in *monitored_crop*. By default, all the crop codes are used for the classification but for some specific products/countries we might want to exclude some of them;
8) the crop codes that have too few pixels show a bad accuracy in the classification results. It is needed a minimum number of pixels by crop code to get enough information for the classification. All crop codes with a *pixel_ratio* higher than *pix_ratio_min* (by default, 0.0002) are considered as having enough pixels to be used in the classification;
9) the crop codes that have too few polygons show a bad accuracy in the classification results. It is needed a minimum number of polygons, *poly_min* (by default, 10) by crop code to get enough information for the classification.

3.1.2 Polygons used for calibration and validation

The polygons that are selected for the classification are used either for the calibration of the classifier model or for the validation of the results (Figure 3-1). Even though the classification is run per-pixel, this split needs to be done at the polygon-level to ensure a proper independent validation following international standards.

This split at the polygon-level is done into 3 steps:

- the smallest polygons are automatically NOT considered for the calibration and thus assigned to the validation pool;
- for each class, the number of training pixels *S2pixCAL* is defined considering the size of the class;
- calibration polygons are randomly selected to reach this *S2pixCAL* value and the remaining polygons are allocated to the validation.

These 3 steps are illustrated in Figure 3-1 and detailed in the following sub-sections.

3.1.2.1 Selecting the best polygons

Only the polygons with a certain number of pixels are selected for the training to use consistent and stable statistic values at polygon-level to train the classification model. This minimum number of pixels, *pix_best*, is a parameter fully configurable. The by-default value is set to **10** pixels.

The polygons with a number of pixels below this minimum number threshold will be automatically excluded from the calibration pool and will be part of the validation pool.

3.1.2.2 Defining the number of training pixel by class

This second step will define the number of training pixels *S2pixCAL* for each class. This *S2pixCAL* value will depend on the representativeness of each class in the area to classify: three strategies are designed to distinguish between the majority classes (strategy 1), the intermediate classes (strategy 2) and the minor classes (strategy 3).

For the minor class (strategy 3), the Synthetic Minority Over-Sampling Technique (SMOTE) algorithm will later be applied (see Section 3.4.2) to artificially populate the calibration dataset by creating synthetic samples.

These strategies are defined using the minimum level of the crop Look-Up Table (LUT) defined in the "*In situ* Data Preparation" ATBD, represented by the *crop_code*. They are illustrated in Figure 3-1 and detailed hereafter and in Code 3-1.

- **Strategy 1 — majority classes**

By default, the majority classes are defined as those counting a number of 10m pixels corresponding to more than 5% (*pix_ratio_hi*) of the total number of pixels considering all classes to map. Crop type belongs to strategy 1 if :

$$pixel\_ratio \geq pix\_ratio\_hi$$

In this case, for each majority class i, the number of training pixels S2pixCAL$_i$ will correspond to 25% of the number of pixels of the class i (*crop_pixels*), with the extra constraint that the number of pixels used for calibration, S2pixCALi, cannot exceed 5% of the total number of pixels. This extra constraint aims at avoiding an unbalanced training dataset:

$$S2pixCAL_i = \min\left(sample\_ratio\_hi \times crop\_pixels\ ;\ pix\_ratio\_hi \times total\_pixels\right)$$

The by-default 5% and 25% values assigned to the parameters *"pix_ratio_hi"* and *",sample_ratio_hi"* can be configured.

- **Strategy 2 — intermediate classes**

The intermediate classes are defined as those having a number of pixels between 1% and 5% of the total number of pixels. Crop type belongs to strategy 2 if:

$$pix\_ratio\_lo \leq pixel\_ratio < pix\_ratio\_hi$$

In this case, for each intermediate class i, the number of training pixels S2pixCAL$_i$ will correspond to 75% of the number of pixels of the class i:

$$S2pixCAL_i = sample\_ratio\_lo \times crop\_pixels$$

Here also, the by-default 1%, 5% and 75% values assigned to the parameters *"pix_ratio_lo", "pix_ratio_hi"* and *"sample_ratio_lo"* can be configured.

- **Strategy 3 — minority classes**

The minor classes are defined as those counting a number of S2 pixels corresponding to less than 1% of the total number of pixels considering all classes to map. Crop type belongs to strategy 3 if:

$$pixel\_ratio < pix\_ratio\_lo$$

In this case, like in Strategy 2, for each minor class i, the number of training pixels S2pixCAL$_i$ will correspond to 75% of the number of pixels of the class i:

$$S2pixCAL_i = sample\_ratio\_lo \times crop\_pixels$$

In addition, the SMOTE algorithm is applied to create a certain number of extra training samples, *smote_pixels*, given the following equation:

$$smote\_pixels = (smote\_ratio \times total\_pixels) - (sample\_ratio\_lo \times crop\_pixels)$$

where the by-default values of *"smote_ratio"* and of *"sample_ratio_lo"* are 0.75% and 75%. All by-default values can be adjusted.

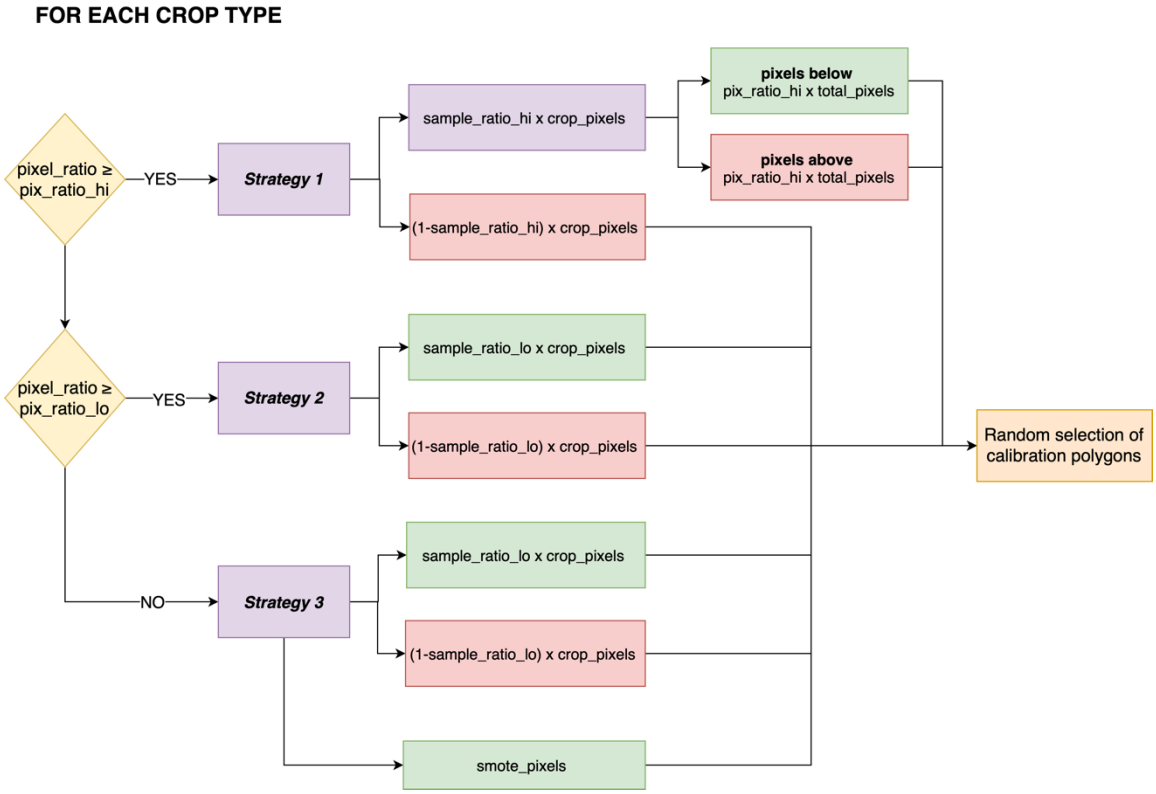


Figure 3-1. In-situ calibration and validation sampling design workflow

```
WITH eligible_polygons AS (
         SELECT in_situ_polygons.*
              , polygon_attributes.pix_10m
              , in_situ_data.crop_code
              , sum(pix_10m) over (partition by crop_code) as crop_pixels
-- compute total pixels by crop type
              , SUM(pix_10m) over () as total_pixels
-- compute total pixels
              , COUNT(*) over (partition by crop_code) as polygon_num
-- compute polygon counts by crop type
         FROM config,
              in_situ_polygons
                  INNER JOIN polygon_attributes using (parcel_id)
                  INNER JOIN in_situ_data using (parcel_id)
                  INNER JOIN crop_list_n4 on crop_list_n4.code_n4 =
in_situ_data.crop_code
                  INNER JOIN crop_list_n3 using (code_n3)
                  INNER JOIN crop_list_n2 using (code_n2)
         WHERE geom_valid
           AND NOT multipart
           AND NOT overlap
```

```
            --  AND quality_control
            AND pix_10m >= pix_min
            AND (monitored_land_cover IS NULL
              OR code_n1 = ANY (monitored_land_cover))
            AND (monitored_crop IS NULL
              OR crop_code = ANY (monitored_crop))
    ),
    eligible_polygons_with_attr AS (
        SELECT *,
               crop_pixels :: float / total_pixels AS pixel_ratio
-- compute pixel ratio
        FROM eligible_polygons
    ),
    selected_polygons AS (
        SELECT eligible_polygons_with_attr.*,
               CASE
                   WHEN pixel_ratio >= pix_ratio_hi THEN 1
                   WHEN pixel_ratio >= pix_ratio_lo THEN 2
                   WHEN pix_10m >= pix_best THEN 3
                   ELSE 4 -- validation only
               END AS strategy
        FROM config,
             eligible_polygons_with_attr
        WHERE pixel_ratio >= pix_ratio_min
          AND polygon_num >= poly_min
    )
SELECT selected_polygons.*
FROM selected_polygons;
```

Code 3-1. SQL snippet for *in situ* data sample selection

3.1.2.3 Selecting randomly calibration polygons

Once the number of calibration pixels for each class ($S2pixCAL_i$) is defined, the calibration polygons are randomly selected such that the sum of their pixels is less than or equal to $S2pixCAL_i$. The remaining polygons are assigned to the validation pool (Code 3-2).

```
for i in crop_code_list:


    # Select polygons with S2pix ≥ S2pixBEST and with SUBnum = SUBnumi

    polys_by_class = polys_best.loc[polys_best['SUBnum'] == i]


    # Reorder randomly all the polygons selected

    polys_by_class_reordered = polys_by_class.sample(len(polys_by_class), random_state=10)


    # Compute cumulative sum and select polygons where cumsum ≤ S2pixCAL à purpose = 1
(calibration)

    polys_cal = polys_by_class_reordered[polys_by_class_reordered['S2pix'].cumsum() <= S2pixCALi]


    # Compute cumulative sum and select polygons where cumsum > S2pixCAL à purpose = 2
(validation)
```

```
polys_val = polys_by_class_reordered[polys_by_class_reordered['S2pix'].cumsum() > S2pixCAL_i]
```

Code 3-2. Random selection of calibration / validation polygons to meet the number of calibration pixels a priori defined for each class

3.1.3 Extracting pixels from calibration polygons

Once the purpose of each polygon is known, the pixels inside the polygons selected for calibration (*purpose* = 1) need to be extracted and stored into a new database. A new *pixID* is added to each calibration pixel (Figure 3-2).

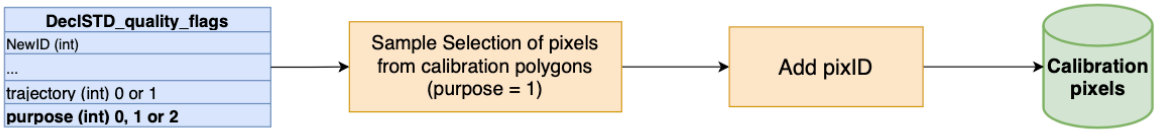


Figure 3-2. Workflow for moving from calibration polygons to calibration pixels

The variables used for this pixels extraction step are listed in Table 3-3 and the implementation is detailed in sub-sections 3.1.3.1 and 3.1.3.2.

Table 3-3. Variables for the sample selection

Input variable	Role	Default value [format]
support_image	Support image that will be classified. For instance, we could use {feat_name}_{stratumID} (Table 3-27)	[GeoTIFF / VRT]
code_field	Name of the field where the class code is stored	crop_code [string]
table_name	Name of the SQLite table, typically the name of the file.	[string]
{country}_{year} _xml_polygonStat	Statistics of a training polygon set: - number of samples per class - number of samples per geometry *This file must be generated because it is mandatory in otbcli_SampleSelection*	[XML]

3.1.3.1 Sample selection

The OTB's application "*PolygonClassStatistics*" generates an XML file with the statistics of the calibration polygons which is a mandatory input for the "*SampleSelection*" application. In practice, actually, this file will not be used by the function because it is chosen to extract all the samples (*strategy* = *all*). Indeed, we the separation between the calibration and validation polygons has already been done in the previous step.

```
otbcli_PolygonClassStatistics
   -in support_image
   -vec {country}_{year}_DeclSTD_classif_flags (where purpose = 1)
   -field {code_field}
   -out {country}_{year}_xml_polygonStat
```

Code 3-3. OTB's application — Polygon Class Statistics

```
otbcli_SampleSelection
   -in support_image
   -vec {country}_{year}_DeclSTD_classif_flags (where purpose = 1)
   -out {country}_{year}_CalPix
   -instats {country}_{year}_xml_polygonStat
   -sampler periodic
   -strategy all
   -field code_field
```

Code 3-4. OTB's application — Sample Selection

3.1.3.2 Add pixel ID

A new "*pixId*" is added to each new generated sample (= points = centroids of each calibration pixel) to facilitate the next steps.

```
ogrinfo
   -dialect SQLite
   -sql "ALTER TABLE table_name ADD COLUMN pixId integer"
   {country}_{year}_CalPix

ogrinfo
   -dialect SQLite
   -sql "UPDATE table_name SET pixId = rowid"
   {country}_{year}_CalPix
```

Code 3-5. Adding new ID to each calibration pixel

3.2 Optical EO data temporal resampling & gap filling

The goal of this second step of temporal resampling and gap filling is to produce an image time series which is

1. gap-filled with respect to missing data;
2. temporally sampled on a regular 10-day grid.

These two elements are key to allow performing classifications over large areas in a consistent way.

The workflow of these temporal resampling and gap filling is illustrated in Figure 3-3.

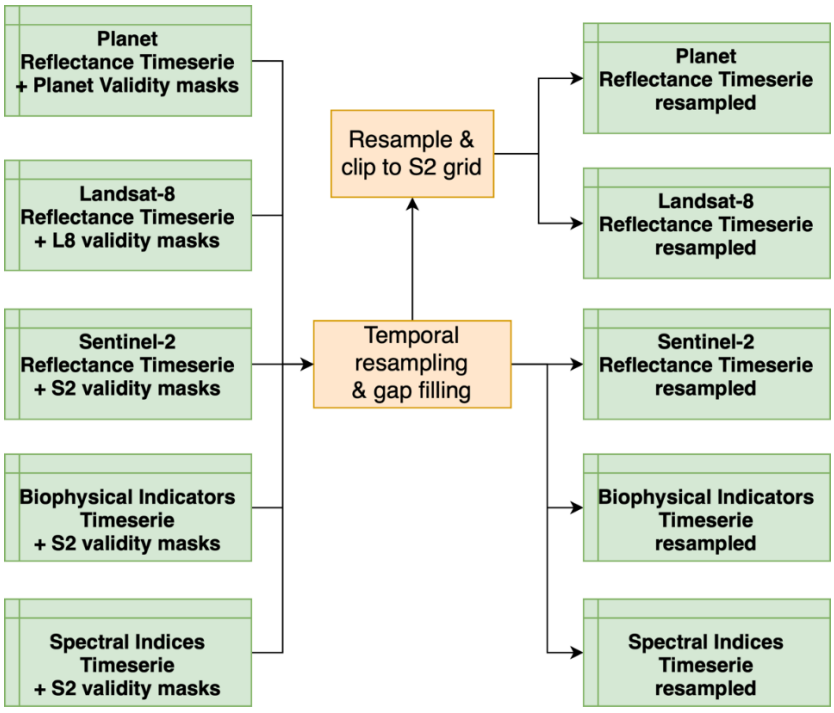


Figure 3-3. Temporal resampling & gap filling workflow

Missing data are referred to as the data masked as cloud, cloud shadow and saturated pixels. Gap filling is performed using a linear interpolation, with a constraint on the length of the temporal window.

The 10-day period of the resampling was chosen instead of the 5-day (which is the S2 native temporal resolution) to avoid an over-fitting of the classifier that can be easily caused by using very correlated data. The first date for each interpolated time series is the median of the first available dates of all tiles.

This step should be applied independently to all sensors used in the classification process (S2, L8 and later Planet). It is also applied independently to biophysical indicators and spectral indices time series.

Input and output data of this step are presented in Table 3-4 and the related parameters are provided in

Table 3-5. The implementation is detailed in Code 3-6.

Table 3-4. Input and output data for the temporal resampling and gap filling step

Input data	Description	Default value [format]
s2_b{xx}_ts_{tile}	Original S2 L2A surface reflectance time series; {xx} = band number; {tile} = S2 tile name	[GeoTIFF]
{spectral_name}_ts_{tile}	Original spectral indices time series; {spectral_name} = NDVI, NDWI, BRIGHT; {tile} = S2 tile name	[GeoTIFF]

{bv_name}_ts_{tile}	Original biophysical indicators time series; {bv_name} = LAI, fAPAR, fCOVER; {tile} = S2 tile name	[GeoTIFF]
s2_validityMask	Validity mask for each acquisition date	[GeoTIFF]
input_dates	Dates of each image acquisition	[Julian dates]
t_0	First date (starting sampling date)	Median of first date [Julian date]
t_end	Last date	Median of last date [Julian date]
Output data	**Description**	**Default value [format]**
s2_b{xx}_ts_{tile}_res	Temporally resampled S2 L2A surface reflectance time series; {xx} = band number; {tile} = S2 tile name	[GeoTIFF]
{spectral_name}_ts_{tile}_res	Temporally resampled spectral indices time series; {spectral_name} = NDVI, NDWI, BRIGHT; {tile} = S2 tile name	[GeoTIFF]
{bv_name}_ts_{tile}_res	Temporally resampled biophysical indicators time series; {bv_name} = LAI, fAPAR, fCOVER; {tile} = S2 tile name	[GeoTIFF]
dates_resampled	List of dates after the temporal resampling	[Julian dates]

Table 3-5. Parameters for the temporal resampling and gap filling algorithm

Parameters	Description	Default value [format]
samplingPeriod	Temporal sampling rate	10 [days]
radiusWindow	Radius of the temporal window	15 [days]
maxTempDist	Maximum distance allowed between 2 dates to perform interpolation	30 [days]
noData	No data value	-10000 [integer]

The research of valid dates is performed inside a temporal window of size "2 × radius + 1". The function *find_previous_valid_date* (resp. *find_next_valid_date*) searches backward (resp. forward) the first date for which a valid value is available.

```
begin
  for pixel in tocr and mpixel in mask:
    odc    = 0
    od     = t_0
    outpix = no_data

    while od ≤ t_end:
        pvd = find_previous_valid_date(od, mask, input_dates)
        nvd = find_next_valid_date(od, mask, input_dates)
        dvd = nvd − pvd

        if dvd < max_temporal_dist:
```

```
          pweight     = od - pvd
          nweight     = nvd - od
          outpix[od] = (pixel[pvd]×pweight+pixel[nvd]×nweight)/(pweight+nweight)

        else :
          outpix[od] = no_data

        odc = odc + 1
        od  = t_0 + sampling period × odc

    end
    rtocr[positon(pixel)] = outpix
  end
end
```

Code 3-6. Optical data temporal resampling & gap filling

3.3 EO features computation & extraction

3.3.1 Features computation

Many features, inherited from the Sentinel-2 for Agriculture (Sen2-Agri) and Sentinels for Common Agricultural Policy (Sen4CAP) experience, were tested in the benchmarking. These classification features can be separated in different groups. These groups and their associated output features are listed in the sub-sections below while the input data for the features' computation in itself are provided in Table 3-6.

Table 3-6. Input data for features computation step

Input data	Description	Default value [format]
s2_b{band}_ts_{tile}_res	Temporally resampled and gap-filled S2 L2A surface reflectance time series; {band} = band number – 03, 04, 05, 06, 07, 08, 11, 12 ; {tile} = S2 tile name	[GeoTIFF]
{spectral_name}_ts_{tile}_res	Temporally resampled and gap-filled spectral indices time series; {spectral_name} = NDVI, NDWI, BRIGHT; {tile} = S2 tile name	[GeoTIFF]
{bv_name}_ts_{tile}_res	Temporally resampled and gap-filled biophysical indicators time series; {bv_name} = LAI, fAPAR, fCOVER; {tile} = S2 tile name	[GeoTIFF]
bck_ts_{tile}	Backscattering time series; {tile} = S2 tile name	[GeoTIFF]
cohe_ts_{tile}	Coherence time series; {tile} = S2 tile name	[GeoTIFF]
Output data	**Description**	**Default value [format]**
{feat_name}_{tile}	Computed features; {feat_name} = feature name; {tile} = S2 tile name	[VRT/GeoTIFF]

3.3.1.1 Group A: Sentinel-2 surface reflectance for each resampled date

Surface reflectance values for each resampled date come from the temporally resampled and gap-filled time series of S2 (see workflow in Figure 3-3). For each date of the time series, the reflectance value are extracted for the bands of interest: green (B03), red (B04), NIR (B08), red-edge (B05-07) and SWIR (B11-12).

Table 3-7. S2 surface reflectance features

ID	Feature name	Description
A{band}	S2_ts_{band}_{tile}	Sentinel-2 surface reflectance for each resampled date ; {band} = band number – B03, B04, B05, B06, B07, B08, B11, B12; {tile} = S2 tile name

Examples

A03 → green reflectance time series for each resampled date during the period of interest

A11 → SWIR reflectance time series for each resampled date during the period of interest

3.3.1.2 Groups B-C-D: NDVI, NDWI, Brightness spectral indices for each resampled date

Spectral indices values for each resampled date come from the temporally resampled and gap-filled spectral indices time series. These spectral indices will also be used to compute statistic and temporal features.

Table 3-8. Spectral indices features

ID	Feature name	Description
B	NDVI_ts_{tile}	NDVI for each resampled date
C	NDWI_ts_{tile}	NDWI for each resampled date
D	BRIGHT_ts_{tile}	Brightness for each resampled date

3.3.1.3 Groups E-F-G: LAI, fAPAR, fCOVER biophysical indicators for each resampled date

Biophysical indicators values for each resampled date come from the temporally resampled and gap-filled biophysical indicators time series.

Table 3-9. Biophysical indicators features

ID	Feature name	Description
E	LAI_ts_{tile}	Leaf Area Index for each resampled date
F	fAPAR_ts_{tile}	fAPAR for each resampled date
G	fCOVER_ts_{tile}	fCOVER for each resampled date

3.3.1.4 Groups H-I-J-K: S2 red-edge features for each resampled date

Four red-edge features, illustrated in Table 3-10, can be computed from Sentinel-2 data for each resampled date.

Table 3-10. S2 red-edge features

ID	Feature name	Description
H	NDVIredge_{tile}	Red edge NDVI for each resampled date (B08 -B06) / (B08+B06)
I	Redge_pos_{tile}	Sentinel-2 Red Edge Position for each resampled date 705 + 35 *(0.5*(B07+B04)-B05) /(B06-B05)
J	PSRI_{tile}	Plant Senescence Reflectance Index for each resampled date (B04-B02)/B05
K	Chl_Redge_{tile}	Chlorophyll Red-Edge for each resampled date B05/B08

3.3.1.5 Spectral indices statistic features (from groups B-C-D)

Five different statistic features are computed for each pixel from the temporally resampled and gap-filled time series of the spectral indices during all the monitoring period:

- Maximum;
- Minimum;
- Mean;
- Median;
- Standard deviation.

The aim of these features is to better discriminate the land cover classes not included in cropland area. The maximum (or minimum) of each index is calculated as well as the average of the 3 maximum (or minimum) values to avoid the risk of taking an outlier as maximum (or minimum) value.

Table 3-11. Spectral indices statistic features

ID	Feature name	Description
1	NDVImax_{tile}	Average of the 3 maximum NDVI; {tile} = S2 tile name
2	NDVImin_{tile}	Average of the 3 minimum NDVI; {tile} = S2 tile name
3	NDVImean_{tile}	Mean NDVI; {tile} = S2 tile name
4	NDVImedian_{tile}	Median NDVI; {tile} = S2 tile name
5	NDVIstd_{tile}	Standard deviation NDVI; {tile} = S2 tile name
6	NDWImax_{tile}	Average of the 3 maximum NDWI; {tile} = S2 tile name
7	NDWImin_{tile}	Average of the 3 minimum NDWI; {tile} = S2 tile name
8	NDWImean_{tile}	Mean NDWI; {tile} = S2 tile name

9	NDWImedian_{tile}	Median NDWI; {tile} = S2 tile name
10	NDWIstd_{tile}	Standard deviation NDWI; {tile} = S2 tile name
11	BRIGHTmax_{tile}	Average of the 3 maximum Brightness; {tile} = S2 tile name
12	BRIGHTmin_{tile}	Average of the 3 minimum Brightness; {tile} = S2 tile name
13	BRIGHTmean_{tile}	Mean Brightness; {tile} = S2 tile name
14	BRIGHTmedian_{tile}	Median Brightness
15	BRIGHTstd_{tile}	Standard deviation Brightness

Parameters for the spectral indices statistic features computation step are given in Table 3-12, and the implementation is explained by Code 3-7.

Table 3-12. Parameters for spectral indices statistic features computation step

Parameters	Description	Default value [format]
dates_resampled	List of dates after the temporal resampling	[Julian dates]

```
otbcli_FeaturesWithInsituMeanmax
   -ndvi NDVI_ts_{tile}_res
   -ndwi NDWI_ts_{tile}_res
   -brightness BRIGHT_ts_{tile}_res
   -dates date_resampled
   -out stat_temp_feat int16
```

Code 3-7. OTB's application — Spectral indices statistic features computation

3.3.1.6 NDVI (group B) temporal features

Crop NDVI signatures allow describing the three most important crop stages:

- the onset of greenness;
- a maturity period;
- the onset of senescence.

NDVI temporal features are computed to characterize the evolution of a crop profile from the temporally resampled and gap-filled NDVI time series during all the monitoring period.

It includes:

- Features detecting largest local NDVI transitions corresponding to the greenness and the senescence onset periods (Table 3-13);

Table 3-13. Features detecting largest local NDVI transitions

ID	Feature name	Description
16	NDVIdifMax_{tile}	Maximum NDVI difference found in a slicing temporal neighbourhood having a size "win_size"

17	NDVIdifMin_{tile}	Minimum NDVI difference found in a slicing temporal neighbourhood having a size "win_size"
18	NDVIdifDif_{tile}	Difference between the two before-mentioned features, estimating the transition jump

- Features associated to the maximum NDVI value, and more precisely to the area containing the maximum NDVI value (Table 3-14);

Table 3-14. Features associated to the maximum NDVI value

ID	Feature name	Description
19	NDVImaxm_{tile}	Maximum mean NDVI value found in a slicing neighbourhood having a size "win_size"
20	NDVImaxmLg_{tile}	Length of the flat zone containing the peak area associated to the maximum mean NDVI value
21	NDVImaxmSr_{tile}	Surface of the flat zone containing the peak area associated to the maximum mean NDVI value

- Features associated to the greenness onset, i.e. to the associated to the largest increasing period of the NDVI function (Table 3-15);

Table 3-15. Features associated to the greenness onset

ID	Feature name	Description
22	NDVIposSr_{tile}	Maximum surface of the first positive derivative period
23	NDVIposLg_{tile}	Length of the first positive derivative period associated to the maximum surface
24	NDVIposRt_{tile}	Rate of the first positive derivative period associated to the maximum surface

- Features associated to the senescence onset, i.e. to the largest decreasing period of the NDVI function (Table 3-16);

Table 3-16. Features associated to the senescence onset

ID	Feature name	Description
25	NDVInegSr_{tile}	Maximum surface of the first negative derivative period
26	NDVInegLg_{tile}	Length of the first negative derivative period associated to the maximum surface
27	NDVInegRt_{tile}	Rate of the first negative derivative period associated to the maximum surface

- Features characterizing bare soil transitions, i.e. detecting if there is a transition between the bare soil before the greenness onset or after the senescence onset (Table 3-17). There are two features, one for the greenness and one for the senescence, being both binary (value 0 or 1).

Table 3-17. Features characterizing bare soil transitions

ID	Feature name	Description
28	NDVIposTr_{tile}	Transition between the bare soil before the greenness onset
29	NDVInegTr_{tile}	Transition between the bare soil after the senescence onset

Parameters for the statistic and temporal features computation step are given in Table 3-18, and the implementation is explained by Code 3-8.

Table 3-18. Parameters for NDVI temporal features computation step

Parameters	Description	Default value [format]
dates_resampled	List of dates after the temporal resampling	[Julian dates]
win_size	Slicing temporal neighbourhood window	2 [integer]
T_soil	Threshold defining non cultivated areas	0.2 [float]

```
otbcli_FeaturesWithInsituMeanmax
   -ndvi NDVI_ts_{tile}_res
   -dates date_resampled
   -soil T_soil
   -window win_size
   -out stat_temp_feat int16
```

Code 3-8. OTB's application — NDVI temporal features computation

3.3.1.7 Crop-specific spectral-temporal features

The features of this category are defined according to generic characteristics of crop growing cycle (Table 3-19). They were successfully used in Sen2-Agri to discriminate between cropland and non-cropland areas, but they might also be used to distinguish between crop groups. First, 5 distinct remote sensing stages in the crop cycle are identified for each pixel:

1. the maximum value of NDVI;
2. the minimum value of NDVI;
3. the maximum positive slope of the NDVI time series;
4. the maximum negative slope of the NDVI time series;
5. the maximum value of red.

Second, the spectral information corresponding to these 5 key dates is extracted. These features are time independent, which allowed us to deal with the cropland diversity and the agro-climatic gradient across the landscape.

Table 3-19. Crop-specific spectral-temporal features

ID	Feature name	Description
1{band}	B{band}_maxNDVI_{tile}	Reflectance B{band} at **max**imum value of NDVI date
2{band}	B{band}_minNDVI_{tile}	Reflectance B{band} at **min**imum value of NDVI date
3{band}	B{band}_mpsNDVI_{tile}	Reflectance B{band} at **m**aximum **p**ositive **s**lope of the NDVI time series date
4{band}	B{band}_mnsNDVI_{tile}	Reflectance B{band} at the **m**aximum **n**egative **s**lope of the NDVI time series date
5{band}	B{band}_minRED_{tile}	Reflectance B{band} at **min**imum value of RED date

Examples

103 → Reflectance B03 (green) at **max**imum value of NDVI date

204 → Reflectance B04 (red) at **min**imum value of NDVI date

Input and output data for the computation of these crop specific spectral-temporal features are shown in Table 3-20, and the implementation is explained by Code 3-9.

Table 3-20. Input and output data for computation step

Input data	Description	Default value [format]
s2_b04_ts_{tile}_res	Temporally resampled S2 L2A red surface reflectance time series; {tile} = S2 tile name	[GeoTIFF]
s2_b08_ts_{tile}_res	Temporally resampled S2 L2A NIR surface reflectance time series; {tile} = S2 tile name	[GeoTIFF]
stage_idx	Remote sensing stage in the crop cycle	0-7 [integer]
Output data	**Description**	**Default value [format]**
feat_pos_{tile}	Positions (dates) starting at 0; {tile} = S2 tile name	[GeoTIFF]
feat_val_{tile}	Values; {tile} = S2 tile name	[GeoTIFF]

```
otbcli_NDVIFeatureExtraction /d2/s4s_ucl/race
   -in.red s2_b04_ts_{tile}_res
   -in.nir s2_b08_ts_{tile}_res
   -out.idx feat_pos_{tile}?&gdal:co:COMPRESS=DEFLATE
   -out.val feat_pos_{tile}?&gdal:co:COMPRESS=DEFLATE&gdal:co:BIGTIFF=YES
```

```
otbcli_BandByIndexExtraction /d2/s4s_ucl/race
   -in.img s2_b{xx}_ts_{tile}_res
   -in.idx feat_pos_{tile}
   -sel stage_idx
```

```
    -out {feat_name}_{tile}
```

Code 3-9. OTB's application — Crop-specific spectral-temporal features computation

3.3.1.8 Groups L to U: SAR features

SAR features are computed from coherence and backscatter time series generated by the "EO Data Pre-Processing" processor. Before the computation of SAR backscatter and coherence temporal features, two-weekly composites are generated and formatted to match the S2 tile format (resolution, extent and projection). The compositing step is performed independently for the two passes – ascending and descending – and for the two polarimetry – VV and VH – respectively for the coherence and the backscattering. The VV/VH ratio is also computed for each two-weekly backscattering composites.

Two-weekly backscattering and coherence composites are used as features and summarized in Table 3-21.

Table 3-21. Two-weekly composites SAR features

ID	Feature name	Description
L	BCK_VH_ASC_W{week}_{tile}	Backscattering – VH – ASC – mean over 2 weeks
M	BCK_VV_ASC_W{week}_{tile}	Backscattering – VV – ASC – mean over 2 weeks
N	BCK_RATIO_ASC_W{week}_{tile}	Backscattering – RATIO – ASC – mean over 2 weeks
O	BCK_VH_DESC_W{week}_{tile}	Backscattering – VH – DESC – mean over 2 weeks
P	BCK_VV_DESC_W{week}_{tile}	Backscattering – VV – DESC – mean over 2 weeks
Q	BCK_RATIO_DESC_W{week}_{tile}	Backscattering – RATIO – DESC – mean over 2 weeks
R	COHE_VH_ASC_W{week}_{tile}	Coherence – VH – ASC – mean over 2 weeks
S	COHE_VV_ASC_W{week}_{tile}	Coherence – VV – ASC – mean over 2 weeks
T	COHE_VH_DESC_W{week}_{tile}	Coherence – VH – DESC – mean over 2 weeks
U	COHE_VV_DESC_W{week}_{tile}	Coherence – VV – DESC – mean over 2 weeks

Once the two-weekly composites are computed, the backscattering and coherence values are synthetized for key periods of the growing cycle of crops. This is performed on a pixel-basis using four descriptive statistics:

- Mean;
- Standard deviation;
- Coefficient of variation;
- Value of the quantile 10, as a proxy of the minimum value.

Such temporal features allow to reduce typical noise of the SAR data while keeping a strong temporal dimension.

The following **backscattering** temporal features are computed for the VV, VH and Ratio (VV/VH and ASC/DESC separately) from the two-weekly composites:

- the **mean** over iterative 2-month periods (Table 3-22).

  If the whole year is considered for the classification, the key periods are: Jan-Feb / Mar-Apr / May-Jun / Jul-Aug / Sep-Oct / Nov-Dec.

Table 3-22. Backscattering mean over iterative 2-month periods

ID	Feature name	Description
Lm	BCK_VH_ASC_Mean_M{mth}_{tile}	Backscattering – VH – ASC – mean over 2 months
Mm	BCK_VV_ASC_ Mean_M{mth}_{tile}	Backscattering – VV – ASC – mean over 2 months
Nm	BCK_RATIO_ASC_ Mean_M{mth}_{tile}	Backscattering – RATIO – ASC – mean over 2 months
Om	BCK_VH_DESC_ Mean_M{mth}_{tile}	Backscattering – VH – DESC – mean over 2 months
Pm	BCK_VV_DESC_Mean_M{mth}_{tile}	Backscattering – VV – DESC – mean over 2 months
Qm	BCK_RATIO_DESC_Mean_M{mth}_{tile}	Backscattering – RATIO – DESC – mean over 2 months

- the **coefficient of variation** over iterative 2 months periods (Table 3-23).

  If the whole year is considered for the classification, the key periods are: Jan-Feb / Mar-Apr / May-Jun / Jul-Aug / Sep-Oct / Nov- Dec.

Table 3-23. Backscattering coefficient of variation over iterative 2-month periods

ID	Feature name	Description
Lc	BCK_VH_ASC_CoV_M{mth}_{tile}	Backscattering – VH – ASC – CoV over 2 months
Mc	BCK_VV_ASC_ CoV _M{mth}_{tile}	Backscattering – VV – ASC – CoV over 2 months
Nc	BCK_RATIO_ASC_ CoV _M{mth}_{tile}	Backscattering – RATIO – ASC – CoV over 2 months
Oc	BCK_VH_DESC_ CoV _M{mth}_{tile}	Backscattering – VH – DESC – CoV over 2 months
Pc	BCK_VV_DESC_ CoV _M{mth}_{tile}	Backscattering – VV – DESC – CoV over 2 months
Qc	BCK_RATIO_DESC_CoV _M{mth}_{tile}	Backscattering – RATIO – DESC – CoV over 2 months

The following **coherence** temporal features are computed (VV/VH separately but ASC/DESC merged) from the two-weekly composites:

- the **standard deviation** along the whole period of interest (Table 3-24).

  If the whole year is considered for the classification, this feature corresponds to the standard deviation of the coherence of each pixel through the entire year and one temporal feature is produced;

Table 3-24. Coherence standard deviation along the whole period

ID	Feature name	Description
RTs	COHE_VH_Std_{tile}	Coherence – VH – std along the whole period of interest
SUs	COHE_VV_Std_{tile}	Coherence – VV – std along the whole period of interest

- the value of the **quantile 10** of the coherence for each month of the period of interest, as a proxy of the minimum coherence value (Table 3-25);

Table 3-25. Coherence quantile 10 for each month

ID	Feature name	Description
RTq	COHE_VH_q10_M{mth}_{tile}	Coherence – VH – quantile10 over 1 month
SUq	COHE_VV_q10_M{mth}_{tile}	Coherence – VV – quantile10 over 1 month

- the **mean** value of the coherence of each month of the period of interest (Table 3-26).

  If the whole year is considered for the classification, 12 temporal features are produced.

Table 3-26. Coherence mean over 1-month period

ID	Feature name	Description
RTm	COHE_VH_Mean_M{mth}_{tile}	Coherence – VH – mean over 1 month
SUm	COHE_VV_Mean_M{mth}_{tile}	Coherence – VV – mean over 1 month

3.3.1.9 Custom features

Coming in the next version

Custom features may be added according to the specific needs of a user. For instance, in some cases, a soil map or a DEM could be use as valuable features. These custom features should be provided with the same projection, resolution and extent as the other features computed by the system.

3.3.2 Features mosaicking

The features computation step is performed on a tile basis. The next step consists in creating a virtual mosaic of all tiles covering the Area Of Interest (AOI), over the full area or over each stratum (Code 3-10). This mosaic is created for each date (resampled date or two-weekly) in the case of a time series or only once in the case of the temporal / statistic features. Input and output data of this step are provided in Table 3-27.

Table 3-27. Input and output data for features mosaicking step

Input data	Description	Default value [format]
{feat_name}_{tile}	Computed features; {feat_name} = feature name; {tile} = S2 tile names belonging to the same stratum	[VRT/GeoTIFF]
no_data	No data value	-10000 [integer]
Output data	**Description**	**Default value [format]**
{feat_name}_{stratumID}	Virtual mosaic of the feature over the stratum area; {feat_name} = feature name; {stratumID} = stratum ID	[VRT]

```
gdalbuildvrt
   -scrnodata no_data
   {feat}_{stratumID}
   {feat_name}_{tile} {feat_name}_{tile} {feat_name}_{tile}
```

Code 3-10. Features mosaicking

3.3.3 Features selection & stacking

The last step of this EO features component of the "Crop Mapping" processor consists in selecting the EO features relevant for the AOI (full area or stratum) among the whole set of computed and mosaicked features and in stacking them in a virtual step. Input and output data are presented in Table 3-28 and the associated code is shown in Code 3-11.

All available EO features for this selection step are organized in tables (Table 5-1 to Table 5-3 in Appendix 1) where each feature is associated with a name and a unique ID. Two modes are available: a default mode where a series of features are automatically selected or a custom mode where the user can select the features he wants to use. A text file is automatically generated using a Pyton code to list all the necessary features.

Table 3-28. Input and output data for features selection step

Input data	Description	Default value [format]
{feat_name}_{stratumID}	Virtual mosaic of the feature over the stratum area; {feat_name} = feature name; {stratumID} = stratum ID	[VRT]
feat_ID_list	Features ID to be selected	[integer, character]
Output data	**Description**	**Default value [format]**
feat_list_{stratumID}	List of the selected mosaicked features; {featGroup} = feature group name; {stratumID} = stratum ID	[TXT]
feat_stack_{stratumID}	Virtual stack of the selected mosaicked features; {featGroup} = feature group name; {stratumID} = stratum ID	[VRT]

The python function called to generate the list of features feat_list_{stratumID}.txt is in ***/Features/FeaturesSelection.py***.

```
gdalbuildvrt
   -separate
   -input_file_list feat_list_{stratumID}.txt
   feat_stack_{stratumID}.vrt
```

Code 3-11. Features selection & stacking

3.4 Pixel-based Classification

Three classifiers are included in the "Crop Mapping" processor: Random Forest (OpenCV and Ranger implementations), Neural Network and Broceliande.

A preliminary step, which is common for all classifiers consists in extracting the selected EO features over the calibration samples. For the minority classes, the EO features are artificially augmented using the SMOTE algorithm. These two steps are described in the sections 3.4.1 and 3.4.2, while the following sub-sections (3.4.3 to 3.4.6) present the functioning of the different classifiers.

3.4.1 Features extraction over calibration pixels

The selected EO features are extracted for each calibration pixel, using the OTB "SampleExtraction" function (Code 3-12). Input and output data are shown in Table 3-29.

Table 3-29. Input and output data for features extraction step over calibration samples

Input data	Description	Default value [format]
feat_stack_{stratumID}	Virtual stack of the selected mosaicked features; {featGroup} = feature group name; {stratumID} = stratum ID	[VRT]
{country}_{year}_CalPix	Pixels used to train the classification model	[PostGIS/SHP]
code_field	Name of the field where the class code is stored	SUBnum [string]
feat_list_name	Full list of features field names	[string1 string2 …]
Output data	**Description**	**Default value [format]**
{country}_{year}_feat _{stratumID}	Calibration pixels with features values	[SQLite]

```
otbcli_SampleExtraction
   -in feat_stack_{stratumID}.vrt
   -vec {country}_{year}_CalPix
   -out {country}_{year}_feat_{stratumID}.sqlite
   -field code_field
   -outfield list
   -outfield.list.names feat_list_name
```

Code 3-12. OTB's application — Sample Extraction

3.4.2 Features augmentation for minor classes

The SMOTE method was introduced by Chawla N. et al (2002)[1] to artificially create calibration samples and increase their number. It is a powerful over-sampling method that allows avoiding over-fitting effects encountered by using more simple methods. Input and output data are presented in Table 3-30The associated code is shown in Code 3-13, , with the parameters in Table 3-31.

Table 3-30. Input and output data for features augmentation

[1] Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, P. (2002). SMOTE: Synthetic Minority Over-sampling TEchnique

Input data	Description	Default value [format]
{country}_{year}_{featGroup} _{stratumID}	Calibration pixels with features values	[SQLite]
Output data	**Description**	**Default value [format]**
{country}_{year}_feat _{stratumID}_smote_{i}	Synthetic calibration pixels generated with SMOTE; {i} = class name	[SQLite]

Table 3-31. Parameters for features augmentation

Parameters	Description	Default value [format]
field_name	Name of the field carrying the class name in the input vectors	crop_code [string]
i	Label of the class of the input file for which new samples will be generated	[integer]
smote_pixels	Number of synthetic extra pixels to add using SMOTE; {i} = class name	[integer]
field_excluded	List of field names in the input vector data that will not be generated in the output file	[string]
smote_neighbors	Number of nearest neighbors to be used in the SMOTE algorithm	5 [integer]

```
otbcli_SampleAugmentation
   -in {country}_{year}_feat_{stratumID}.sqlite
   -field field_name
   -label i
   -samples smote_pixels
   -out {country}_{year}_feat_{stratumID}_smote_{i}.sqlite
   -exclude field_excluded
   -strategy smote
   -strategy.smote.neighbors smote_neighbors
```

Code 3-13. OTB's application — Sample Augmentation

3.4.3 OpenCV Random Forest

3.4.3.1 Build the Random Forest model

The variables are the input data (the time series, the training data), the seed for the random number generator (in order to be able to reproduce exactly the same results in different runs) and the ratio between training and validation samples.

It is worth noting that, even in the training step, a validation of the quality of the classification is performed. This allows to check if the training behaves correctly before producing the complete crop map for the final validation. This is the utility of the parameter giving the ratio between training and validation samples.

However, the sample splitting in this step does not guarantee that a training and a validation sample will not be drawn from the same polygon. These may lead to very similar training and validation samples and, therefore, give optimistic performances. Input and output data as well as the parameters associated with this RF model training are presented in Table 3-32 and Table 3-33.

Table 3-32. Input and output data for training RF model step

Input data	Description	Default value [format]
{country}_{year}_feat_{stratumID}	Calibration pixels with features values	[SQLite]
feat_list_name	Full list of features field names	[string1 string2 …]
Output data	**Description**	**Default value [format]**
{country}_{year}_{featGroup} _{stratumID}_RF_OpenCV_model	Output file containing the model estimated	[TXT]

Table 3-33. Parameters for the OpenCV Random Forest classification algorithm

Parameters	Description	Default value [format]
code_field	Field containing the class id for supervision. The values in this field shall be cast into integers.	crop_code [string]
rf_max	Maximum depth of the tree	25 [integer]
rf_min	Minimum number of samples in each node	25 [integer]
rf_ra	Termination Criteria for regression tree	0 [float]
rf_cat	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split	10 [integer]
rf_var	Size of the randomly selected subset of features at each tree node. If you set it to 0, then the size will be set to the square root of the total number of features.	0 [integer]
rf_nbtrees	Maximum number of trees in the forest	100 [integer]
rf_acc	Sufficient accuracy (OOB error)	0.01 [float]

The RF classifier is trained using the OTB "*Train Vector Classifier*" application, as shown in Code 3-14.

```
otbcli_TrainVectorClassifier
   -io.vd {country}_{year}_feat_{stratumID}
   -io.out { country}_{year}_{stratumID}_RF_OpenCV_model
   -feat feat_list_name
   -cfield code_field
```

```
  -classifier rf
  -classifier.rf.max rf_max
  -classifier.rf.min rf_min
  -classifier.rf.ra rf_ra
  -classifier.rf.cat rf_cat
  -classifier.rf.var rf_var
  -classifier.rf.nbtrees rf_nbtrees
  -classifier.rf.acc rf_acc
```

Code 3-14. OTB's application — Train Vector Classifier

3.4.3.2 Apply the Random Forest model

The RF model is finally applied to the whole area of interest to generate the final crop map, using the OTB "*Image Classifier*" application (Code 3-15). Input and output data are show in Table 3-34.

Table 3-34. Input and output data for training RF model step

Input data	Description	Default value [format]
{country}_{year}_{stratumID} _RF_OpenCV_model	File containing the model estimated	[txt]
feat_{stratumID}	Virtual stack of the selected mosaicked features; {featGroup} = feature group name; {stratumID} = stratum ID	[VRT]
Output data	**Description**	**Default value [format]**
{country}_{year}_{featGroup} _{stratumID}_RF_OpenCV_classif	Image containing class labels	[GeoTIFF]

```
otbcli_ImageClassifier
  -in {featGroup}_{stratumID}
  -model {country}_{year}_{featGroup}_{stratumID}_RF_OpenCV_model
  -out {country}_{year}_{featGroup}_{stratumID}_RF_OpenCV_classif
```

Code 3-15. OTB's application — Image Classifier

3.4.4 Ranger Random Forest

Coming in the next version

3.4.5 Broceliande

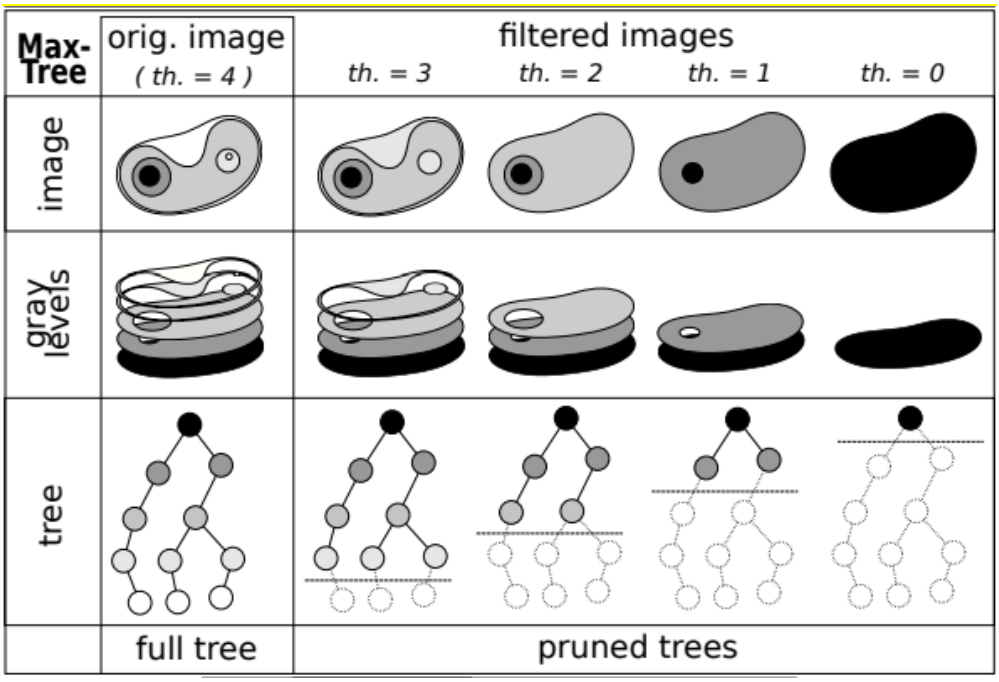
Broceliande is used to extract the permanent crop class from S2 imagery. The broceliande software (as well as broceliandeConfig software) is an executable file that allows to run the C++ broceliande library to process an image. Broceliande is a project from Obelix team in IRISA aimed to perform some classification for

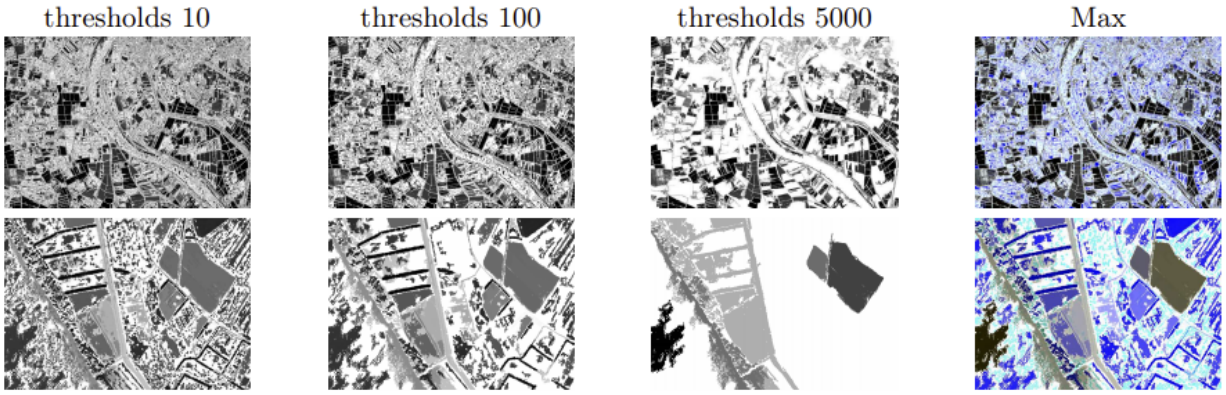
remote sensing images. It is mainly based on 2 other pieces of software, Triskele from Obelix team used to produce hierarchical representations of images, and Shark library used for Machine Learning.

A hierarchical representation of the content of the images can be made by using morphological trees, here most often inclusion trees. Those trees are constructed in such a way that the leaves are small regions of pixels with constant value, and the nodes are connected regions created by adding surrounding pixels with very close values. In that way all possible pixel values are represented up to the value of the root node, which can be either the minimum, the maximum or the median value, according to the type of tree chosen. Let's take an example using pixel values between 0 (black) and 255 (white). We will build the tree node by node, from the leaves to the root by browsing all this interval of pixels values. And the order that we choose to browse the interval will define the type of tree we build:

- If we order the pixel values from 0 (dark) to 255 (white), then the leaves will be local minima, and the tree will be called a "min-tree".
- If we order the pixel values from 255 (white) down to 0 (dark), then the leaves will be local maxima, and the tree will be called a "max-tree".
- If we order the pixel values using the distance to the median value, beginning from the farthest values to the median (either greater or lower) then getting closer and closer to the median, then the leaves will be local extrema (minima and/or maxima), and the tree will be called a "tree of shape". We can mention that the name "tree of shape" comes from the fact that it creates the different nodes by mixing in parallel "bright" and "dark" nodes from the median node, therefore focuses more of the shape of the nodes than on their gray-scale.

In this project, from the set of selected bands, we will build multiscale representations through the model of morphological trees from which we derive multiscale features called attribute profiles ("AP/DAP generator" stage). Such trees can also be seen as a stack of nested segmentations and thus as a generalization of the concept of mono-scale segmentation layer in GEOBIA tools. For each hierarchical representation, we will measure specific attributes (e.g., area, weight, compactness…) for all components, i.e., objects appearing at different scales for a given pixel. These features are thus assigned to each pixel.





The next step is the use of the random forest classifier. Due to its relatively simple parameterization, computation efficiency, and high accuracy, we will use as a basis the Random Forest (RF) classification algorithm based on its proven experience (Merciol et al., 2019). The advantage of using a supervised classifier over predefined rulesets is the ability to adapt to a wide range of landscapes without explicit definition of the properties of mapped objects. However, it also requires labelled samples that describe the requested classes. Given the low computation time of the classification process over the labelled samples, the user can then easily improve the quality of the training set by providing new samples in an active learning way. Once the model is accurate enough, the final prediction will be performed, and the final Land Cover map produced.

Before using Broceliande, two steps are computed to format input data to the software. The first step is to create a stack of all bands needed for the creation of the hierarchical representation and then for the classification. This step is realised by a python script using gdal tools. Input and output are shown in Table 3-35.

Table 3-35. Input and output data for creating a stack of all bands needed for the Broceliande classifier

Input data	Description	Default value [format]
s2_b{xx}_ts_{tile}_res	Temporally resampled and gap-filled S2 L2A surface reflectance time series; {xx} = band number – 02, 03, 04, 05, 06, 07, 08, 08A, 11, 12 ; {tile} = S2 tile name	[GeoTIFF]
intermediate_vrt	Text file of all vrt band created	TXT
Output data	**Description**	**Default value [format]**
output_{band}_{image}	temporary vrt image of each band of each temporally resampled and gap filled S2 L2 surface reflectance ; {band} = band number ; {image} = name of the image in the time serie	VRT
s2_{tile}_{date}_full_stack	Computed stack; {date} = date interval of the time series; {tile} = S2 tile name	VRT & [GeoTIFF]

Each input data is decomposed in one VRT for each band and recomposed in a full VRT stack that will be then converted into GeoTIFF format (Code 3-16).

```
gdalbuildvrt
   output_{band}_{image}
   S2_b{xx}_ts{tile}_res
```

Code 3-16. Broceliande -

A full stack is then created made of all VRT output (Code 3-17).

```
gdalbuildvrt
   -separate
   S2_{tile}_{date}_full_stack.vrt
   Intermediate_vrt.txt
```

Code 3-17. Broceliande - Creation of a full stack from all VRT outputs

This stack is then converted into GeoTIFF format (Code 3-18).

```
gdal_translate
   S2_{tile}_{date}_full_stack.vrt
   S2_{tile}_{date}_full_stack.tif
   -a_nodata -10000
   --config GDAL_MAX_DATASET_POOL_SIZE 1000
```

Code 3-18. Broceliande - Conversion of the stack into GeoTIFF

As a second step, the new training dataset converted in GeoTIFF format is selected.

Table 3-36. Input and output data for …

Input data	Description	Default value [format]
s2_{tile}_{date}_full_stack	Computed stack; {date} = date interval of the time series; {tile} = S2 tile name	[GeoTIFF]
{tile}_{year}_CalPix	Pixels used to train the classification model ; {tile} = tile name	[PostGIS/SHP]
code_field	Name of the field where the class code is stored	SUBnum [string]
Output data	**Description**	**Default value [format]**
{tile}_{year}_CalPix_samplestats	Statistics of sample set; {tile} = tile name	xml
s2_{tile}_{date}_full_stack	Computed stack; {date} = date interval of the time series; {tile} = S2 tile name	[GeoTIFF]

```
otbcli_PolygonClassStatistics
   -in s2_{tile}_{date}_full_stack
   -vec {tile}_{year}_CalPix
   -out {tile}_{year}_CalPix_samplestats
   -field code_field
```

Code 3-19. Broceliande -

Table 3-37. Input and output data for …

Input data	Description	Default value [format]
s2_{tile}_{date}_full_stack	Computed stack; {date} = date interval of the time series; {tile} = S2 tile name	[GeoTIFF]
{tile}_{year}_CalPix	Pixels used to train the classification model ; {tile} = tile name	[PostGIS/SHP]
code_field	Name of the field where the class code is stored	SUBnum [string]
{tile}_{year}_CalPix_samplestats	Statistics of sample set; {tile} = tile name	xml
Output data	**Description**	**Default value [format]**
{tile}_{year}_CalPix_select_updatesamples	selected new set	SHP
s2_{tile}_{date}_full_stack	Computed stack; {date} = date interval of the time series; {tile} = S2 tile name	VRT & [GeoTIFF]

```
otbcli_SampleSelection
   -in s2_{tile}_{date}_full_stack
   -vec {tile}_{year}_CalPix
   -instats {tile}_{year}_CalPix_samplestats
   -out {tile}_{year}_CalPix_select_updatesamples
   -field code_field
   -strategy constant
   -strategy.constant.nb 1000
   -sampler random
   -outrates
```

Code 3-20. Broceliande -

Table 3-38. Input and output data for …

Input data	Description	Default value [format]
s2_{tile}_{date}_full_stack	Computed stack; {date} = date interval of the time series; {tile} = S2 tile name	[GeoTIFF]
code_field	Name of the field where the class code is stored	SUBnum [string]
{tile}_{year}_CalPix_select_updatesamples	selected new set	SHP
Output data	**Description**	**Default value [format]**
{tile}_{year}_CalPix_final_updatesamples	Final sample set	SHP

```
otbcli_SampleExtraction
   -in s2_{tile}_{date}_full_stack
   -vec {tile}_{year}_CalPix_select_updatesamples
   -out {tile}_{year}_CalPix_final_updatesamples
   -field code_field
```

Code 3-21. Broceliande -

The new set of samples is then converted in GeoTIFF format.

Now all the input have been harmonized to fit with Broceliande and the software can be launched for classification through docker. Table 3-39 presents the input and output data of the Broceliande classification in itself, while the associated parameters are listed in Table 3-40. An example of Broceliande launch command is provided in Code 6-1 (Appendix 2).

Table 3-39. Input and output data for the Broceliande classification

Input data	Description	Default value [format]
s2_{tile}_{date}_full_stack	Computed stack; {date} = date interval of the time series; {tile} = S2 tile name	[GeoTIFF]
{tile}_{year}_CalPix_final_updatesamples	Final sample set	SHP
Output data	**Description**	**Default value [format]**
{tile}_{year}_classification	The output classification of Broceliande software containing the class labels	GeoTIFF

Table 3-40. Parameters for the Broceliande classification algorithm

Parameters	Description	Default value [format]
docker run	docker containing Broceliande. “run” command the launch of the docker	docker
name	Name of the docker session	string
--cpuset-cpus	Number of CPU used for the computation	20-40 [integer-integer]
--memory	Memory allocated to Broceliande for the computation	100G [integer]
-v	Specification of volume driver options. 3 field separated by “:” (see docker documentation)	/mnt:/mnt [string]
Path	Docker containing Broceliande at Inria gitlab	registry.gitlab.inria.fr/ obelix/Broceliande/ develop:2.4.20200614 [string]
-i	Computed stack of image	[string]
-o	Broceliande output	[string]
--timeFlag	Display information about execution time of the main steps of the software. This option is also intended for debugging purpose	True
--ndviBands	Produces NDVI bands from 2 input bands. We remind that the bands are numbered starting from 0. We remind that this option needs to be used with option --copyBands in order to keep this band	1,1[intergers]
-b	When dealing with multi-band images (multispectral or hyperspectral images), it is possible to select just some of the available bands. For example is the input image is a RGB image, selecting bands “0,2” means working with only band “Red” and “Blue”	[integer]
-f	--featureType arg ; Select the type of features used to reconstruct the images, from the previously pruned trees, and gather them into a stack of filtered images. We remind that if this feature is the gray-scale level, the stack of images will be called Attribute Profiles (AP), whereas if this feature is any other feature, the stack of images will be called Feature Profiles (FP)	
-t	Select the type of tree of create. The possible values are: • Min (Min-Tree) • Max (Max-Tree) • Med (Median-Tree) • Alpha (Alpha-Tree)	Min [string]

-a	Select the type of attributes used to prune the tree, according to a given list of thresholds. The possible values are: • Area • Perimeter • ZLength (Δ(BoundingBox.z)) • Compactness (= area/perimeter2) • Complexity (= perimeter/area) • Simplicity (= area/perimeter) • Rectangularity (= area/BoundingBoxArea) • Weight (i.e. gray-scale level) • SD (StandardDeviation2 base on mean gray value of nodes) • SDW (StandardDeviation2 b	Area [string]
--thresholds	List of thresholds used to prune the tree, according to the attribute previously chosen.	[interger]
-- autoThreadFlag	Set the threadPerImage and tilePerThread parameters to maximized the memory used and reduce the comptation time.	True
-c	Copy the selected bands. Please note that those bands can be either: • bands from input image (numbers starting from 0) • NDVI or Sobel bands just produced. In that case you have to known the corresponding number.	[integer]
--bgRate	Number of background of ground truth rate of foreground	125% [integer]
-bgTagRate	Number of tagged background of ground truth (the other are taken randomly).	80% [integer]
--showChannel	Displays the current configuration on the screen. N	
--tagValue	Values use as labels in ground truth. This option define list or range of integer.	[integer]
-bgValue	Value use as non foreground in ground truth (nan => no bg tag).	nan [integer]

3.4.6 Neural Network

Coming in the next version

3.4.7 Strata merging

Coming in the next version

3.5 Reclassification

The crop mapping processor can generate different types of crop maps, i.e. maps with legends of different levels of details. For the calibration of the model, the minimum level of the crop LUT defined in the "*In situ* Data Preparation" ATBD, represented by the *crop_code*, is always used, whatever the product to be generated. In other terms, the calibration is used considering as many different classes as possible. The different legends are obtained at the end through a reclassification step.

For this purpose, a new table is provided to the system which maps each '*crop_code*' to a new class. For instance, in order to obtain a Crop Mask, the value "1" can be set to all crop types belonging to "Cropland" and "2" to all crop types belonging to "Non-cropland" (Table 3-41).

Table 3-41. Reclassification table example

crop_code	crop_name	cm	cm name
1111	Winter wheat	1	Cropland
1151	Barley two-row	1	Cropland
2211	Vineyards	1	Cropland
6111	Conifers	2	Non-cropland
8111	Urban	2	Non-cropland

3.6 Validation

A validation is performed within the processor using the pixels that were selected for the classification (*trajectory* = 1) and for the validation (*purpose* = 2). The generated Crop Map is compared with these pixels to build a confusion matrix, using the OTB "*Compute Confusion Matrix*" application (Code 3-22). From this matrix, the Overall Accuracy and the Kappa of the classification are calculated, as well as the producer's and user's accuracy by crop type. These two last values are then used to calculate the F-Score of each crop type. Finally, a plot that present the results of the validation is generated.

The input and output data for the validation are presented in Table 3-42.

Table 3-42. Input and output data for validation step

Input data	Description	Default value [format]
{country}_{year}_{featGroup} _{stratumID}_RF_OpenCV_classif	Image containing class labels	[GeoTIFF]
{country}_{year}_DeclSTD _classif_flags	The standardized version of the NSO *in situ* statistical dataset with geometry, S2 pixels number, quality control and classification flags	[PostGIS]
Output data	**Description**	**Default value [format]**
{country}_{year}_{featGroup}	Confusion matrix	[csv]

_{stratumID}_CM		

Table 3-43. Parameters for validation step

Parameters	Description	Default value [format]
label_field	Field name containing the label values	[string]
no_data	No data value	-10000 [integer]

```
otbcli_ComputeConfusionMatrix
   -in {country}_{year}_{featGroup}_{stratumID}_RF_OpenCV_classif
   -out {country}_{year}_{featGroup}_{stratumID}_CM
   -format contingencytable
   -ref vector
   -ref.vector.in {country}_{year}_DeclSTD_classif_flags (where purpose = 2)
   -ref.vector.field label_field
   -ref.vector.nodata no_data
```

Code 3-22. OTB's application — Compute Confusion Matrix

3.7 Large-area mapping & Stratification

The "Crop Mapping" processor must be functional at national scale. As the amount of calibration data is limited and will probably not be distributed uniformly over the country, it must be used to the full extent. This means that a single classification model should be used for the entire area of interest instead of one per input tile.

The "Crop Mapping" processor offers the possibility to stratify the area of interest, i.e. to split it into multiple agro-climatic regions - called strata - which are homogeneous in terms of climate, agro-ecological conditions (relief, soil, etc.), cropping systems and agricultural practices. The use of such stratification allows reducing the natural variability existing when working at national scale by coping with agro-climatic gradients inducing a very diversity of crop calendars and growing conditions.

Each stratum is classified independently, i.e. with his own set of calibration pixels and his own classification model. The advantages of this approach are twofold: (i) using stratum-specific models gives better classification results and (ii) training a classifier for a single stratum can require fewer resources than for a full country.

From the implementation point of view, the system should allow users defining a list of strata for a site in the form of a **shapefile with one polygon per stratum**. Besides the extent, the only required information is a **stratum numeric identifier**, which is then used in the output product metadata.

When available, the stratification shapefile is used as follows:

1. the Standardized Sen4Stat *in situ* statistical dataset (with geometry, S2 pixels number, quality control flags) is intersected with the stratification shapefile and split into one data set for each stratum;

2. the *in situ* dataset preparation (section 3.1), features extraction and features augmentation for minor crops (sections 3.4.1 and 3.4.2) and the classifier training is performed for each stratum;
3. during the classification, the corresponding model is applied to each pixel, generating a single classification for each stratum;
4. the validation is applied for each stratum and the by-stratum figures will need to be brought together in a next step;
5. any post-filtering that might be required is applied afterwards.

Because of the size of the input data, it's preferable to use an online approach, where the classification features are computed on-the-fly instead of saving them to disk. This is especially important for larger sites, as the amount of disk space taken by the extracted features would be prohibitive. Another advantage of an online method is that computing the features is faster than saving them to disk and reading them back, even when multiple passes over are needed.

4 Output

4.1 Binary cropland — non-cropland map

The map is a raster file, in the GeoTIFF format where the legend is binary (cropland - non cropland), with the extra class of “no data”.

4.2 Annual vs Permanent crop map

The map is a raster file, in the GeoTIFF format where the legend is either binary (annual crop - permanent crop) or ternary (if the non cropland is also included). It also has the extra class of “no data”.

4.3 Crop groups

The map is a raster file, in the GeoTIFF format. The legend of this map counts a limited number of classes corresponding to the main crop groups, as for instance cereals, oil crops, etc. The groups are defined by the user and are specific to each AOI.

4.4 Crop type map

The map is a raster file, in the GeoTIFF format. This product is the map with the highest level of thematic detail, where the legend identifies the crop types. The legend is defined by the user and is specific to each AOI.

4.5 Log file

A log file is generated with a summary of all the steps performed as well as the features and parameters used.

- **General**
  - product
  - projection
  - LUT
- **Features**
  - TimeseriesFeatures
  - SingleFeatures
  - CustomFeatures
- **InSitu**
  - LC_monitored
  - SUB_monitored
  - S2pixMIN
  - pixRatioMIN
  - polyMIN

    - S2pixBEST
    - pixRatioH
    - pixRatioL
    - smoteRatio
    - sampleRatioH
    - sampleRatioL
- **TempResampGapfill**
    - noData
    - interpTechnique (=linear)
        - samplingPeriod
        - radiusWindow
        - maxTempDist
    - interpTechnique (=whittaker)
- **Classifier**
    - classifierType (=OpencvRF)
        - rf_nbtrees
        - rf_max
        - rf_min
    - classifierType (=RangerRF)
        - rg_nbtrees
        - rg_max
        - rg_min
    - classifierType (=RNN)
    - classifierType (=Transformer)

5 Appendix 1. Summary of EO features

Table 5-1. Summary optical time series features

ID	Feature name	Description
A{band}	S2_{date}_{band}_{tile}	Sentinel-2 surface reflectance for each resampled date; {band} = B03, B04, B05, B06, B07, B08, B11, B12
B	NDVI_{date}_{tile}	NDVI for each resampled date
C	NDWI_{date}_{tile}	NDWI for each resampled date
D	BRIGHT_{date}_{tile}	Brightness for each resampled date
E	LAI_{date}_{tile}	Leaf Area Index for each resampled date
F	fAPAR_{date}_{tile}	fAPAR for each resampled date
G	fCOVER_{date}_{tile}	fCOVER for each resampled date
H	NDVIredge_{date}_{tile}	Red edge NDVI for each resampled date (B08 -B06) / (B08+B06)
I	Redge_pos_{date}_{tile}	Sentinel-2 Red Edge Position for each resampled date 705 + 35 *(0.5*(B07+B04)-B05) /(B06-B05)
J	PSRI_{date}_{tile}	Plant Senescence Reflectance Index for each resampled date (B04-B02)/B05
K	Chl_Redge_{date}_{tile}	Chlorophyll Red-Edge for each resampled date B05/B08

Table 5-2. Summary SAR features

ID	Feature name	Description
L	BCK_VH_ASC_W{week}_{tile}	Backscattering – VH – ASC – mean over 2 weeks
M	BCK_VV_ASC_W{week}_{tile}	Backscattering – VV – ASC – mean over 2 weeks
N	BCK_RATIO_ASC_W{week}_{tile}	Backscattering – RATIO – ASC – mean over 2 weeks
O	BCK_VH_DESC_W{week}_{tile}	Backscattering – VH – DESC – mean over 2 weeks
P	BCK_VV_DESC_W{week}_{tile}	Backscattering – VV – DESC – mean over 2 weeks
Q	BCK_RATIO_DESC_W{week}_{tile}	Backscattering – RATIO – DESC – mean over 2 weeks
R	COHE_VH_ASC_W{week}_{tile}	Coherence – VH – ASC – mean over 2 weeks
S	COHE_VV_ASC_W{week}_{tile}	Coherence – VV – ASC – mean over 2 weeks
T	COHE_VH_DESC_W{week}_{tile}	Coherence – VH – DESC – mean over 2 weeks
U	COHE_VV_DESC_W{week}_{tile}	Coherence – VV – DESC – mean over 2 weeks

Lm	BCK_VH_ASC_Mean_M{mth}_{tile}	Backscattering – VH – ASC – mean over 2 months
Mm	BCK_VV_ASC_ Mean_M{mth}_{tile}	Backscattering – VV – ASC – mean over 2 months
Nm	BCK_RATIO_ASC_ Mean_M{mth}_{tile}	Backscattering – RATIO – ASC – mean over 2 months
Om	BCK_VH_DESC_ Mean_M{mth}_{tile}	Backscattering – VH – DESC – mean over 2 months
Pm	BCK_VV_DESC_Mean_M{mth}_{tile}	Backscattering – VV – DESC – mean over 2 months
Qm	BCK_RATIO_DESC_Mean_M{mth}_{tile}	Backscattering – RATIO – DESC – mean over 2 months
Lc	BCK_VH_ASC_CoV_M{mth}_{tile}	Backscattering – VH – ASC – CoV over 2 months
Mc	BCK_VV_ASC_ CoV _M{mth}_{tile}	Backscattering – VV – ASC – CoV over 2 months
Nc	BCK_RATIO_ASC_ CoV _M{mth}_{tile}	Backscattering – RATIO – ASC – CoV over 2 months
Oc	BCK_VH_DESC_ CoV _M{mth}_{tile}	Backscattering – VH – DESC – CoV over 2 months
Pc	BCK_VV_DESC_ CoV _M{mth}_{tile}	Backscattering – VV – DESC – CoV over 2 months
Qc	BCK_RATIO_DESC_CoV _M{mth}_{tile}	Backscattering – RATIO – DESC – CoV over 2 months
RTs	COHE_VH_Std_{tile}	Coherence – VH – std along the whole period of interest
SUs	COHE_VV_Std_{tile}	Coherence – VV – std along the whole period of interest
RTq	COHE_VH_q10_M{mth}_{tile}	Coherence – VH – quantile10 over 1 month
SUq	COHE_VV_q10_M{mth}_{tile}	Coherence – VV – quantile10 over 1 month
RTm	COHE_VH_Mean_M{mth}_{tile}	Coherence – VH – mean over 1 month
SUm	COHE_VV_Mean_M{mth}_{tile}	Coherence – VV – mean over 1 month

Table 5-3. Summary of single features

ID	Feature name	Description
1	NDVImax_{tile}	Average of the 3 maximum NDVI
2	NDVImin_{tile}	Average of the 3 minimum NDVI
3	NDVImean_{tile}	Mean NDVI
4	NDVImedian_{tile}	Median NDVI
5	NDVIstd_{tile}	Standard deviation NDVI
6	NDWImax_{tile}	Average of the 3 maximum NDWI
7	NDWImin_{tile}	Average of the 3 minimum NDWI
8	NDWImean_{tile}	Mean NDWI
9	NDWImedian_{tile}	Median NDWI
10	NDWIstd_{tile}	Standard deviation NDWI
11	BRIGHTmax_{tile}	Average of the 3 maximum Brightness
12	BRIGHTmin_{tile}	Average of the 3 minimum Brightness

13	BRIGHTmean_{tile}	Mean Brightness
14	BRIGHTmedian_{tile}	Median Brightness
15	BRIGHTstd_{tile}	Standard deviation Brightness
16	NDVIdifMax_{tile}	Maximum NDVI difference found in a slicing temporal neighbourhood having a size "win_size"
17	NDVIdifMin_{tile}	Minimum NDVI difference found in a slicing temporal neighbourhood having a size "win_size"
18	NDVIdifDif_{tile}	Difference between the two before-mentioned features, estimating the transition jump
19	NDVImaxm_{tile}	Maximum mean NDVI value found in a slicing neighbourhood having a size "win_size"
20	NDVImaxmLg_{tile}	Length of the flat zone containing the peak area associated to the maximum mean NDVI value
21	NDVImaxmSr_{tile}	Surface of the flat zone containing the peak area associated to the maximum mean NDVI value
22	NDVIposSr_{tile}	Maximum surface of the first positive derivative period
23	NDVIposLg_{tile}	Length of the first positive derivative period associated to the maximum surface
24	NDVIposRt_{tile}	Rate of the first positive derivative period associated to the maximum surface
25	NDVInegSr_{tile}	Maximum surface of the first negative derivative period
26	NDVInegLg_{tile}	Length of the first negative derivative period associated to the maximum surface
27	NDVInegRt_{tile}	Rate of the first negative derivative period associated to the maximum surface
28	NDVIposTr_{tile}	Transition between the bare soil before the greenness onset
29	NDVInegTr_{tile}	Transition between the bare soil after the senescence onset
1{band}	B{band}_maxNDVI_{tile}	Reflectance B{band} at **max**imum value of NDVI date
2{band}	B{band}_minNDVI_{tile}	Reflectance B{band} at **min**imum value of NDVI date
3{band}	B{band}_mpsNDVI_{tile}	Reflectance B{band} at **m**aximum **p**ositive **s**lope of the NDVI time series date
4{band}	B{band}_mnsNDVI_{tile}	Reflectance B{band} at the **m**aximum **n**egative **s**lope of the NDVI time series date
5{band}	B{band}_minRED_{tile}	Reflectance B{band} at **min**imum value of RED date

6 Appendix 2. Broceliande launch command

An example of Broceliande launch command is provided below (Code 6-1). Note that the input date only contains Red and NIR bands for the NDVI computation.

```
docker run
--rm
--name sen4stat
--cpuset-cpus="20-40"
--memory="100G"
-v /mnt:/mnt registry.gitlab.inria.fr/obelix/broceliande/develop:2.4.20200614
-i s2_{tile}_{date}_full_stack
-o {tile}_{year}_classification
-g {tile}_{year}_CalPix_final_updatesamples
--timeFlag
--ndviBands 0,1 --ndviBands 2,3 --ndviBands 4,5 --ndviBands 6,7 --ndviBands 8,9
--ndviBands 10,11 --ndviBands 12,13 --ndviBands 14,15 --ndviBands 16,17
--ndviBands 18,19 --ndviBands 20,21 --ndviBands 22,23 --ndviBands 24,25
--ndviBands 26,27 --ndviBands 28,29 --ndviBands 30,31 --ndviBands 32,33
--ndviBands 34,35 --ndviBands 36,37 --ndviBands 38,39 --ndviBands 40,41
--ndviBands 42,43 --ndviBands 44,45 --ndviBands 46,47 --ndviBands 48,49
--ndviBands 50,51 --ndviBands 52,53 --ndviBands 54,55 --ndviBands 56,57
--ndviBands 58,59 --ndviBands 60,61 --ndviBands 62,63 --ndviBands 64,65
--ndviBands 66,67 --ndviBands 68,69
-b 70 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 71 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 72 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 73 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 74 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 75 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 76 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 77 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 78 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 79 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 80 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 81 -f AP -t Max -a area --thresholds 400,1000,10000,30000
```

```
-b 82 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 83 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 84 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 85 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 86 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 87 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 88 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 89 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 90 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 91 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 92 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 93 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 94 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 95 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 96 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 97 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 98 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 99 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 100 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 101 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 102 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 103 -f AP -t Max -a area --thresholds 400,1000,10000,30000
-b 104 -f AP -t Max -a area --thresholds 400,1000,10000,30000
--autoThreadFlag
-c 70-104
--showChannel
--tagValue 1,2,3,4
--bgValue 100
```

Code 6-1. Example of Broceliande launch command