

ESA Sen4Stat

Sentinels for Agricultural Statistics



D7.0 – DDF - ATBD

“Algorithm Theoretical Basis Document: Spectral Indices & Biophysical Indicators”

Issue/Rev : 1.0
Date Issued : 25-08-2021
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 Polytechnica 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/eli>

Disclaimer

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.
To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA



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
1.0	25/08/2021	1 st version related to the Acceptance Review

Detailed record sheet

From version 1.0 to 1.x

Comment	Section	Problem description	Change

Table of contents

1	Logical model and processor overview	7
2	Input data	8
2.1	Sentinel-2 L2A BOA reflectance	8
2.2	Geometry of acquisition — only for BI	8
2.2.1	Extracting the local viewing angles of S2 products	8
2.2.2	Extracting the local viewing angles of Landsat-8 products	11
2.3	Artificial Neural Network — only for BI	11
3	Detailed workflow	12
3.1	Spectral Indices computation	12
3.1.1	NDVI	12
3.1.2	NDWI	13
3.1.3	Brightness	13
3.2	Biophysical Indicators computation	14
3.2.1	Normalization of the inputs	18
3.2.2	Neural network	18
3.2.3	Denormalization of the output	18
3.2.4	Pseudo-code	18
4	Output	20
4.1	Index timeseries	20
4.2	Quality flags	20
4.2.1	Status flag	20
4.2.2	Input/output values range	20

List of figures

Figure 1-1. Workflow of the spectral indices and biophysical indicators computation	7
Figure 3-1. Illustration of the 3 BIs targeted by the Sen4Stat system.....	14
Figure 3-2. Example of parameters text file for LAI generation using S2 acquisitions.....	18

List of tables

Table 2-1. S2 spectral characteristics	8
Table 2-2. L8 spectral characteristics	8
Table 3-1. Input and output data of the SIs computation step.....	12
Table 3-2. S2 spectral characteristics for NDVI computation	12
Table 3-3. L8 spectral characteristics for NDVI computation	12
Table 3-4. S2 spectral characteristics for NDWI computation.....	13
Table 3-5. L8 spectral characteristics for NDWI computation	13
Table 3-6. S2 spectral characteristics for Brightness computation	13
Table 3-7. L8 spectral characteristics for NDVI computation	14
Table 3-8. Input and output data of the BIs computation step	15
Table 4-1. Definition domain of the S2 reflectance data.....	20
Table 4-2. Definition domain of the L8 reflectance data	21
Table 4-3. Definition domain for the output index values	21

Acronyms

Acronym	Definition
AD	Applicable Document
ANN	Artificial Neural Network
ATBD	Algorithm Theoretical Basis Document
BI	Biophysical Indicator
EO	Earth Observation
ESA	European Space Agency
FAPAR	Fraction of Absorbed Photosynthetically Active Radiation
FCOVER	Fraction of Vegetation Cover
GAI	Green Area Index
L1, L2, L3, L4	Level 1, Level 2, Level 3, Level 4
L8	Landsat 8
LAI	Leaf Area Index
NDVI	Normalized Difference Vegetation Index
NDWI	Normalized Difference Water Index
NIR	Near InfraRed
S2	Sentinel-2
Sen4Stat	Sentinels for Agricultural Statistics
SI	Spectral Index
SWIR	Short-Wave InfraRed
USGS	United States Geological Survey

1 Logical model and processor overview

This document is the Algorithm Theoretical Basis Document (ATBD) that describes the algorithm developed to produce the Biophysical Indicators (BIs) and Spectral Indices (SI) product from Sentinel-2 (S2) and Landsat 8 (L8) acquisitions.

For the BIs - Leaf Area Index (LAI), Fraction of Vegetation Cover (FCOVER) and Fraction of Absorbed Photosynthetically Active Radiation (FAPAR), this ATBD is derived from one already proposed by Marie Weiss and Frédéric Baret in the frame of the ESA Sentinel-2 toolbox¹.

Both BIs and SIs are provided for each S2 / L8 acquisition.

The workflow is summarized in Figure 1-1 and detailed in the following sections of this document.

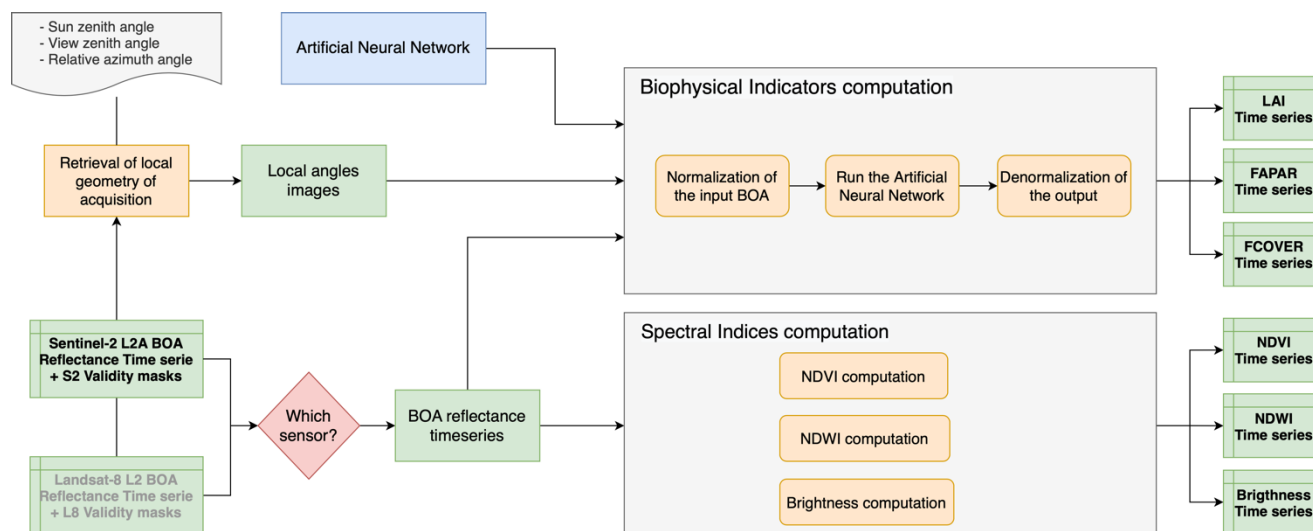


Figure 1-1. Workflow of the spectral indices and biophysical indicators computation

¹ Weiss M. and Baret F., 2016. S2 Toolbox Level 2 products: LAI, FAPAR, FCOVER, V1.1

2 Input data

2.1 Sentinel-2 L2A BOA reflectance

Individual daily observations are considered. All S2 bands are used except the blue one (B2): B3, B4, B5, B6, B7, B8a, B8, B11 and B12. All L8 bands are used except the blue one (B2): B3, B4, B5, B6, B8.

Table 2-1 and Table 2-2 presents the spectral characteristics of the S2 and L8 bands used for the BI retrieval.

Table 2-1. S2 spectral characteristics

Acronym	Central wavelength (nm)	Width (nm)	Spatial resolution (m)	Band name
B3	560	35	10	Green
B4	665	30	10	Red
B5	705	15	20	Red edge 1
B6	740	15	20	Red edge 2
B7	783	20	20	Red edge 3
B8	842	115	10	NIR wide
B8a	865	20	20	NIR narrow
B11	1610	90	20	SWIR 1
B12	2190	180	20	SWIR 2

Table 2-2. L8 spectral characteristics

Acronym	Central wavelength (nm)	Width (nm)	Spatial resolution (m)	Band name
B3	560	30	30	Green
B4	650	20	30	Red
B5	865	15	30	NIR narrow
B6	1610	40	30	SWIR 1
B7	2200	90	30	SWIR 2

2.2 Geometry of acquisition — only for BI

The chosen implementation of the BI retrieval makes use of a global Artificial Neural Network (ANN), and this ANN takes as input the characteristics of the acquisition geometry. Consequently, the directional information must be considered when training the neural networks, similarly to the spectral information. The cosine of the sun zenith angle (θ_s), view zenith angle (θ_v) and relative azimuth angle (ϕ) at the time of the image acquisition are required. The step of extracting the observation angles of each pixel is described here below.

2.2.1 Extracting the local viewing angles of S2 products

The local observation angles (i.e. for every pixel) of S2 acquisitions can be derived from the general L2A HDR file which is providing the observation angles for the 12 detectors of each band on a 5x5 km grid. This

can be done using the pseudo-code provided in the Code 2-1. The output is a 3-band raster file respectively containing the information of the cosine of the sun zenith angle (θ_s) (band 1), view zenith angle (θ_v) (band 2) and relative azimuth angle (ϕ) (band 3) of each pixel.

```
import os
import argparse
from lxml import etree
import numpy as np
from osgeo import gdal, osr

parser = argparse.ArgumentParser(description='Take the 5x5 km angles and convert it to a user defined
resolution vrt with bilinear interpolation. The code produces three .tif (5kmx5km) and .vrt (10mx10m)
containing "cos(angle)" with angle being view_zenith, solar_zenith and solar_azimuth-view_azimuth. NB: since
we store integers, the output is actually 10000*cos(x).')
parser.add_argument('--inputAngleFile', help='Path to the input file containing the angles (xml/HDR
format).', required=True)
parser.add_argument('--inputRasterFile', help='Path to an input raster corresponding to the inputAngleFile
(used to extract origin and projection.', required=True)
parser.add_argument('--outputFile', help='Path and name of the output files. Do not include extension, the
script writes a .tif with 5x5km and a .vrt with the user defined resolution.', required=True)
parser.add_argument('--target_res', default=10, help='Target vrt resolution (do not support non-square
pixels). Default value is 10m.')

args = parser.parse_args()
nodata = 32767

# Dump array information into a raster
def array2raster(outRasterName, rasterOrigin, pixelWidth, pixelHeight, projection, array,
dataType=gdal.GDT_Int16):
    cols = array.shape[1]
    rows = array.shape[0]
    originX = rasterOrigin[0]
    originY = rasterOrigin[1]
    driver = gdal.GetDriverByName('GTiff')
    outRaster = driver.Create(outRasterName, cols, rows, 1, dataType, ["TILED=YES", "COMPRESS=LZW"])
    outRaster.SetGeoTransform((originX, pixelWidth, 0, originY, 0, pixelHeight))
    outband = outRaster.GetRasterBand(1)
    outband.SetNoDataValue(nodata)
    outband.WriteArray(array)
    outRasterSRS = osr.SpatialReference(projection)
    outRaster.SetProjection(outRasterSRS.ExportToWkt())
    outband.FlushCache()
    print("File %s written."%outRasterName)

# Convert the string matrices to numpy arrays and computes the 10000*cos(angle)
def string2array(string, computeCos=True):
    if computeCos:
        return np.array([[int(10000 * np.cos(np.radians(float(angle)))) for angle in line.split(" ") for line
in string.splitlines()])
    else:
        return np.array([[float(angle) for angle in line.split(" ") for line in string.splitlines()])

# Get input tif informations
input_raster = gdal.Open(args.inputRasterFile)
projection = input_raster.GetProjectionRef()
raster_x_size = input_raster.RasterXSize
raster_y_size = input_raster.RasterYSize
raster_origin = [input_raster.GetGeoTransform()[0], input_raster.GetGeoTransform()[3]]

# Parse the xml file
namespaces = {'ns': 'http://eop-cfi.esa.int/CFI'}
tree = etree.parse(args.inputAngleFile)

# Get the angle matrix as string
def get_matrixAngle_string(xpath):
```

```

out_string = ""
for entry in tree.xpath(xpath, namespaces=namespaces)[0].getchildren():
    out_string += entry.text + "\n"
return out_string

def write_tif_and_vrt(matrixAngle_array, stringToAppend):
    # Dump the array with 1000*cos(angle) in a tif
    angle_rawResolution_outName = args.outputFile + "_rawResolution_" + stringToAppend + ".tif"
    #print(angle_rawResolution_outName)
    #print(matrixAngle_array)
    array2raster(angle_rawResolution_outName, raster_origin, pixel_x_size_input_angles,
    pixel_y_size_input_angles * -1, projection, matrixAngle_array)
    #Produce a vrt with the user define resolution
    print("Writing %s vrt:%s" % (stringToAppend, stringToAppend))
    os.system("gdalbuildvrt %s %s -tr %d %d -r bilinear -srcnodata %d -vrtnodata %d" % (args.outputFile +
    "_userResolution_" + stringToAppend + ".vrt", angle_rawResolution_outName, args.target_res, args.target_res,
    nodata, nodata))
    #os.system("gdal_translate %s %s" % (args.outputFile + "_userResolution_" + stringToAppend + ".vrt",
    args.outputFile + "_userResolution_" + stringToAppend + ".tif"))

# Let's assume for now that the resolution are the same for all the angles (Solar, Observation, Zenith and
Azimuth
pixel_x_size_input_angles =
float(tree.xpath("/ns:Earth_Explorer_Header/ns:Variable_Header/ns:Specific_Product_Header/ns:Product_Infor
mation/ns:Solar_Angles/ns:Zenith/ns:COL_STEP", namespaces=namespaces)[0].text)
pixel_y_size_input_angles =
float(tree.xpath("/ns:Earth_Explorer_Header/ns:Variable_Header/ns:Specific_Product_Header/ns:Product_Infor
mation/ns:Solar_Angles/ns:Zenith/ns:ROW_STEP", namespaces=namespaces)[0].text)

# Solar Zenith Angle (never nodata...)
solarAngle_Zenith_string =
get_matrixAngle_string("/ns:Earth_Explorer_Header/ns:Variable_Header/ns:Specific_Product_Header/ns:Product
_Information/ns:Solar_Angles/ns:Zenith/ns:Values_List")
write_tif_and_vrt(string2array(solarAngle_Zenith_string), "cosSolarAngleZenith")

# for the view angle, we average over the detectors and bands
def get_mean_over_matrices(xpath):
    view_angle_matrices = []
    for value_list in (tree.xpath(xpath, namespaces=namespaces)):
        temp_matrix_string = ""
        for line in value_list.getchildren():
            temp_matrix_string += line.text + "\n"
        view_angle_matrices.append(np.array([[float(angle) for angle in line.split(" ")] for line in
temp_matrix_string.splitlines()]])
    #print(len(view_angle_matrices))
    #print(view_angle_matrices[0].shape)
    view_angle_mean = np.zeros(view_angle_matrices[0].shape)
    view_angle_countNonNan = np.zeros(view_angle_matrices[0].shape)
    for matrix in view_angle_matrices:
        for (i,j), value in np.ndenumerate(matrix):
            if not (np.isnan(value)):
                view_angle_mean[i,j] += value
                view_angle_countNonNan[i,j] += 1
            # to see variations across the bands and det id:
            # if (i,j) == (1,1):
            #     print(value)
    #print(view_angle_mean)
    #print(view_angle_countNonNan)
    for (i,j), value in np.ndenumerate(view_angle_countNonNan):
        if value != 0:
            view_angle_mean[i, j] = view_angle_mean[i, j] / value
        else:
            view_angle_mean[i, j] = nodata
    #print(view_angle_mean)
    return view_angle_mean

```

```
#View Zenith Angle
view_angle_zenith_mean =
get_mean_over_matrices("/ns:Earth_Explorer_Header/ns:Variable_Header/ns:Specific_Product_Header/ns:Product
_Information/ns:List_of_Viewing_Angles/ns:Viewing_Incidence_Angles_Grids/ns:Zenith/ns:Values_List")
cos_view_angle_zenith_mean = nodata*np.ones(view_angle_zenith_mean.shape)
for (i,j), value in np.ndenumerate(view_angle_zenith_mean):
    if not value == nodata:
        cos_view_angle_zenith_mean[i,j] = int(10000 * np.cos(np.radians(value)))
write_tif_and_vrt(cos_view_angle_zenith_mean, "cosViewAngleZenith")

#View Azimuth Angle
view_angle_azimuth_mean =
get_mean_over_matrices("/ns:Earth_Explorer_Header/ns:Variable_Header/ns:Specific_Product_Header/ns:Product
_Information/ns:List_of_Viewing_Angles/ns:Viewing_Incidence_Angles_Grids/ns:Azimuth/ns:Values_List")

# Solar Azimuth Angle
solarAngle_Azimuth_string =
get_matrixAngle_string("/ns:Earth_Explorer_Header/ns:Variable_Header/ns:Specific_Product_Header/ns:Product
_Information/ns:Solar_Angles/ns:Azimuth/ns:Values_List")
solarAngle_Azimuth_array = string2array(solarAngle_Azimuth_string, computeCos=False)

relative_angle_azimuth = nodata*np.ones(view_angle_azimuth_mean.shape)
for (i,j), value in np.ndenumerate(view_angle_azimuth_mean):
    if value == nodata or solarAngle_Azimuth_array[i,j] == nodata:
        relative_angle_azimuth[i,j] = nodata
    else:
        relative_angle_azimuth[i,j] = solarAngle_Azimuth_array[i,j] - value
cos_relative_angle_azimuth = nodata*np.ones(relative_angle_azimuth.shape)
for (i,j), value in np.ndenumerate(relative_angle_azimuth):
    if not value == nodata:
        cos_relative_angle_azimuth[i,j] = int(10000 * np.cos(np.radians(value)))
write_tif_and_vrt(cos_relative_angle_azimuth, "cosRelativeAngleAzimuth")
```

Code 2-1. Extraction of local observation angles in S2 L2A acquisition

2.2.2 Extracting the local viewing angles of L8 products

The local observation angles (i.e. for every pixel) of L8 acquisitions can be derived from the Angle Coefficient File contained in the L1C product ("ANG.txt"). To do so, an executable file is provided on the United States Geological Survey (USGS) web site (<https://landsat.usgs.gov/solar-illumination-and-sensor-viewing-angle-coefficient-file>).

2.3 Artificial Neural Network — only for BI

The LAI, FCOVER and FAPAR BI retrieval is performed by applying a global ANN on each pixel considering the reflectance values of all the available bands pre-processed at the Level 2A (L2A) and some geometric configuration as input. The training of the ANN, that consists in generating the training database, defining the neural network architecture and calibrating the network, will not be performed within the Sen4Stat system. Instead, the Sen4Stat system will benefit from an **already trained ANN**, made openly and freely available by INRA. From the system implementation point of view, this trained ANN will be given as auxiliary data to the processor.

3 Detailed workflow

3.1 Spectral Indices computation

Three SIs are computed for each acquisition date: Normalized Difference Vegetation Index (NDVI), Normalized Difference Water Index (NDWI) and the Brightness index (Table 3-2).

Table 3-1. Input and output data of the SIs computation step

Input data	Description	Default value [format]
s2_b{xx}_ts_{tile}	Original S2 L2A surface reflectance timeseries; {xx} = band number; {tile} = S2 tile name	[GeoTIFF]
l8_b{xx}_ts_{tile}	Original L8 L2A surface reflectance timeseries; {xx} = band number; {tile} = S2 tile name	[GeoTIFF]
Output data	Description	Default value [format]
{spectral_name}_ts_{tile}	Spectral indices timeseries; {spectral_name} = NDVI, NDWI, BRIGHT; {tile} = S2 tile name	[GeoTIFF]

3.1.1 NDVI

The well-known NDVI^{2,3} is the most commonly used index to qualify the vegetated areas. It is a simple optical band ratio computed from the surface reflectance in the Near-InfraRed (NIR) and the red channels.

The inputs required for the NDVI computation correspond to the BOA reflectance values for each considered pixel. Individual daily observations are used. For S2, bands 4 and 8 are used while for L8, this is the bands 4 and 5. Their spectral characteristics are described in Table 3-2 and Table 3-3.

Table 3-2. S2 spectral characteristics for NDVI computation

Acronym	Central wavelength (nm)	Width (nm)	Spatial resolution (m)	Band name
B4	665	30	10	Red
B8	842	115	10	NIR wide

Table 3-3. L8 spectral characteristics for NDVI computation

Acronym	Central wavelength (nm)	Width (nm)	Spatial resolution (m)	Band name
B4	650	20	30	Red
B5	865	15	30	NIR narrow

² Rouse, J.W., Haas, R.H., Schell, J.A. and Deering, D.W., 1973, Monitoring vegetation systems in the Great Plains with ERTS. In 3rd ERTS Symposium, NASA SP-351 I, pp. 309–317

³ Tucker, C. J., 1979: Red and photographic infrared linear combinations for monitoring vegetation. Remote Sens. Environ., 8, 127–150

Computing the NDVI over each acquisition is a simple band math operation :

$$NDVI = \frac{\rho(NIR) - \rho(RED)}{\rho(NIR) + \rho(RED)}$$

3.1.2 NDWI

The inputs required for the NDWI computation correspond to the BOA reflectance values for each considered pixel. Individual daily observations are used. For S2, bands 8 and 11 are used - the band 11 (SWIR) being resampled at 10m resolution, while bands 5 and 6 are used for L8. Their spectral characteristics are described in Table 3-4 and Table 3-5.

Table 3-4. S2 spectral characteristics for NDWI computation

Acronym	Central wavelength (nm)	Width (nm)	Spatial resolution (m)	Band name
B8	842	115	10	NIR wide
B11	1610	90	20	SWIR 1

Table 3-5. L8 spectral characteristics for NDWI computation

Acronym	Central wavelength (nm)	Width (nm)	Spatial resolution (m)	Band name
B5	865	15	30	NIR narrow
B6	1609	85	30	SWIR 1

Computing the NDWI over each acquisition is a simple band math operation :

$$NDWI = \frac{\rho(SWIR) - \rho(NIR)}{\rho(SWIR) + \rho(NIR)}$$

3.1.3 Brightness

The brightness is defined as the Euclidean norm of the surface reflectance values in green ($\rho(0.560)$), red ($\rho(0.665)$), NIR ($\rho(0.842)$) and SWIR ($\rho(1.610)$).

The inputs required for the Brightness computation correspond to the BOA reflectance values for each considered pixel. Individual daily observations are used. For S2, bands 3, 4, 8 and 11 are used - the band 11 (SWIR) being resampled at 10m resolution. For L8, the algorithm uses the bands 3, 4, 5 and 6. Their spectral characteristics are described in Table 3-6 and

Table 3-7.

Table 3-6. S2 spectral characteristics for Brightness computation

Acronym	Central wavelength (nm)	Width (nm)	Spatial resolution (m)	Band name
B3	560	35	10	Green
B4	665	30	10	Red
B8	842	115	10	NIR wide
B11	1610	90	20	SWIR 1

Table 3-7. L8 spectral characteristics for NDVI computation

Acronym	Central wavelength (nm)	Width (nm)	Spatial resolution (m)	Band name
B3	562	57	30	Green
B4	650	20	30	Red
B5	865	15	30	NIR narrow
B6	1609	85	30	SWIR 1

Computing the Brightness over each acquisition is a simple band math operation :

$$\text{Brightness} = \sqrt{\rho(\text{GREEN})^2 + \rho(\text{RED})^2 + \rho(\text{NIR})^2 + \rho(\text{SWIR})^2}$$

3.2 Biophysical Indicators computation

As already mentioned, three different BIs are computed: LAI, FAPAR and FCOVER (Figure 3-1). Each BI retrieval will be performed for every 10-meter pixel covering the area of interest and for each acquisition date.

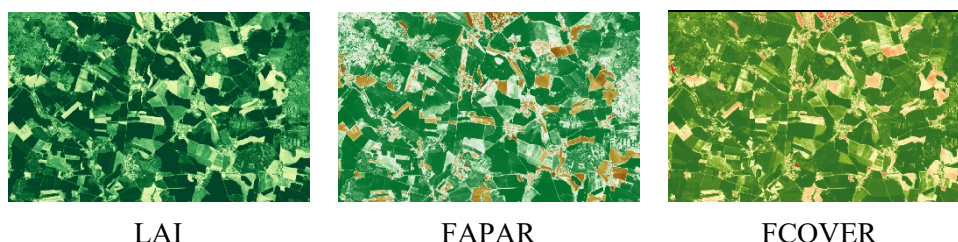


Figure 3-1. Illustration of the 3 BIs targeted by the Sen4Stat system

LAI is defined as half the developed area of photosynthetically active elements of the vegetation per unit horizontal ground area. It determines the size of the interface for exchange of energy (including radiation) and mass between the canopy and the atmosphere. This is an intrinsic canopy primary variable that should not depend on observation conditions. LAI is strongly non-linearly related to reflectance. Therefore, its estimation from remote sensing observations will be strongly scale dependent^{4,5}. Note that vegetation LAI as estimated from remote sensing will include all the green contributors, i.e. including understory when existing under forests canopies. However, except when using directional observations⁶, LAI is not directly accessible from remote sensing observations due to the possible heterogeneity in leaf distribution within the

⁴ Garrigues, S., Allard, D., Baret, F., & Weiss, M. (2006). Influence landscape spatial heterogeneity on the non-linear estimation of leaf area index from moderate spatial resolution remote sensing data. *Remote Sensing of Environment*, 105, 286-298

⁵ Weiss, M., Baret, F., Myneni, R., Pragnère, A., & Knyazikhin, Y. (2000). Investigation of a model inversion technique for the estimation of crop characteristics from spectral and directional reflectance data. *Agronomie*, 20, 3-22

⁶ Chen, J.M., Menges, C.H., & Leblanc, S.G. (2005). Global mapping of foliage clumping index using multi-angular satellite data. *Remote Sensing of Environment*, 97, 447-457

canopy volume. Therefore, remote sensing observations are rather sensitive to the “effective” LAI, i.e. the value that would produce the same remote sensing signal as that actually recorded one, while assuming a random distribution of leaves. The difference between the actual and effective LAIs may be quantified by the clumping index⁶ that roughly varies between 0.5 (very clumped canopies) and 1.0 (randomly distributed leaves)⁷. The LAI measured from remote sensing is often referred to as the Green Area Index (GAI) since it includes all the green contributors of the canopy.

LAI values range from 0 to 8.

FAPAR corresponds to the fraction of photosynthetically active radiation absorbed by the canopy. The FAPAR value results directly from the radiative transfer model in the canopy which is computed instantaneously. It depends on canopy structure, vegetation element optical properties and illumination conditions. FAPAR is very useful as input to several primary productivity models based on simple efficiency considerations⁸. Most of the primary productivity models using this efficiency concept are running at the daily time step. Consequently, the product definition should correspond to the daily integrated FAPAR value that can be approached by computation of the clear sky daily integrated FAPAR values as well as the FAPAR value computed for diffuse conditions. To improve the consistency with other FAPAR products that are sometimes considering the instantaneous FAPAR value at the time of the satellite overpass under clear sky conditions (e.g. MODIS), a study was proposed to investigate the differences between alternative FAPAR definitions⁹. Results show that the instantaneous FAPAR value at 10:00 (or 14:00) solar time is very close to the daily integrated value under clear sky conditions. FAPAR is relatively linearly related to reflectance values and will be little sensitive to scaling issues. Note also that the FAPAR refers only to the green parts (leaf chlorophyll content higher than $15\mu\text{g.cm}^{-2}$) of the canopy.

FAPAR values range from 0 to 1.

FCOVER corresponds to the gap fraction for nadir direction. FCOVER is used to separate vegetation and soil in energy balance processes, including temperature and evapotranspiration. It is computed from the leaf area index and other canopy structural variables and does not depend on variables such as the geometry of illumination as compared to FAPAR. For this reason, it is a very good candidate for the replacement of classical vegetation indices for the monitoring of green vegetation. Because of its quasi-linear relationship with reflectance, FCOVER will be only marginally scale dependent⁵. Note that similarly to LAI and FAPAR, only the green elements (leaf chlorophyll content higher than $15\mu\text{g.cm}^{-2}$) will be considered.

FCOVER values range from 0 to 1.

Input and output data needed to compute these BIs are presented in Table 3-8.

Table 3-8. Input and output data of the BIs computation step

Input data	Description	Default value [format]
s2_b{xx}_ts_{tile}	Original S2 L2A surface reflectance timeseries; {xx} = band number; {tile} = S2 tile name	[GeoTIFF]

⁷ Weiss M., Baret F., Lero M., Hauteceura O., Bacourd C., Prévotb L. and Bruguierb N., 2002. Validation of neural net techniques to estimate canopy biophysical variables from remote sensing data. *Agronomie*, 22, 547-554.

⁸ Prince, S.D. (1991). A model of regional primary production for use with coarse resolution satellite data. *International Journal of Remote Sensing*

⁹ Baret, F., Leroy, M., Roujean, J.L., Knorr, W., Lambin, E., & Linderman, M. (2003). CYCLOPES User Requirement Document. In. Avignon: INRA-CSE

l8_b{xx}_ts_{tile}	Original L8 L2A surface reflectance timeseries; {xx} = band number; {tile} = S2 tile name	[GeoTIFF]
ann	Artificial Neural Network trained by INRA	[txt]
local_geom	Local geometry of acquisition	
Output data	Description	Default value [format]
{bv_name}_ts_{tile}	Original biophysical indicators timeseries; {bv_name} = LAI, FAPAR, FCOVER; {tile} = S2 tile name	[GeoTIFF]

Considering that the ANN models are provided to the processor as auxiliary datasets, the algorithm implementation can be summarized in 3 steps:

1. normalizing the inputs;
2. running the ANN;
3. denormalizing the outputs.

From the system implementation point of view, a specific effort should be made to allow easy update of the BI processor in the future. Therefore, all the coefficients useful for the normalization/denormalization, the neural network models and the domain definition should be stored in the form of parameter tables that can easily be changed whenever an update is available. These parameters tables are provided by INRA in the form of Excel files: there is one file for each product and for each sensor. Six parameters tables are therefore needed to be able to generate the 3 BI (LAI, FCOVER and FAPAR) using both S2 and L8 acquisitions. To be easily readable by the processor, these excel sheets are converted into text files using the Python script provided in Code 3-1.

```
import openpyxl, sys, pystache

wb = openpyxl.load_workbook(filename=sys.argv[1], read_only=True)

data = {
    'inputs': [], # {min_norm, max_norm, label}
    'layers': [], # {(inputs: [labels,..],) neurons: [{bias, weights}], function, number}
    'output': {} # min_norm, max_norm, min, max, tolerance

ws = wb['Normalisation']
inputs=[]
for row in ws.iter_rows(row_offset=3):
    if row[0].value == None:
        if data['inputs']:
            break
        continue
    data['inputs'].append({'label':row[0].value, 'min_norm':row[1].value, 'max_norm':row[2].value})
    inputs.append(row[0].value)

row_output = tuple(ws.rows)[-1] # defined as the last data of this sheet
data['output']['name'] = row_output[0].value
data['output']['min_norm'] = row_output[1].value
data['output']['max_norm'] = row_output[2].value

ws = wb['Weights']

lay_indexes = []
index = 0
for row in ws.iter_rows():
    if row[0].value == 'neuron1':
        lay_indexes.append(index)
```

```

index += 1

nbInputs = len(inputs)
for index in lay_indexes:
    newlayer = {'neurons': []}
    newlayer['number'] = 0
    for row in ws.iter_rows(row_offset=index):
        if row[0].value == None:
            break
        elif row[0].value == 'bias':
            for i in range(newlayer['number']):
                newlayer['neurons'][i]['bias'] = row[i+1].value
        elif row[0].value == 'Transfert Fct':
            newlayer['function'] = row[1].value
        else: # neuron
            newlayer['neurons'].append({'weights': [row[i+1].value for i in range(nbInputs)]})
            newlayer['number'] += 1
    if not data['layers']: # first
        newlayer['inputs'] = inputs
    else:
        newlayer['inputs'] = ['neuron'+str(i+1) for i in range(nbInputs)]
    data['layers'].append(newlayer)
    nbInputs = newlayer['number']

ws = wb['Extreme Cases']

data['output']['tolerance'] = ws['B10'].value
data['output']['min'] = ws['C10'].value
data['output']['max'] = ws['D10'].value

with open(sys.argv[2], 'r') as tpl_f:
    tpl = tpl_f.read()
    rendered = pystache.render(tpl, data)
    with open(sys.argv[3], 'w') as out_f:
        out_f.write(rendered)

```

Code 3-1. Conversion of excel files into text files, for the parameters' tables

Figure 3-2 provides an example of the extracted text file for the LAI ANN using S2 images.

```

# number of layers, defined by their transfert function and size
tansig 5 purelin 1
# min/max for normalisation of inputs
0 0.24227150564744757 0 0.29180643729483435 0.023617798406067352 0.7940468337225911 0 0.3352184967986723
0.02202129476436714 0.6082236185467771 0.0282828940537622 0.7593032519236953 0.027033167693721202
0.7762830093591506 0.008452168326508781 0.4951157687164573 0 0.4939202941174727 0.9185974648964176
0.999999998075627 0.3420445117819629 0.9359109848078707 -0.9999999998938771 0.9999999949266675
# definition of layers (first column is the bias)
# bias B3 B4 B8 B5 B6 B7 B8A B11 B12 cos(View_Zenith) cos(Sun_Zenith) cos(Rel_Azimuth)
-1.2942679301598696 -0.3742492703985799 -1.0879940321273938 0.9073468650336846 0.3615250088966526 -
3.118927273401686 2.77498648736835 -0.057300763562444526 -1.234084059620391 -0.8564958529719705 -
0.4841042554196577 0.1515712348243339 0.6152433844319842
0.8734788297065322 0.44320224366190497 0.8887592231132775 -0.5056153268593457 0.9098195307014335 -
0.21344245581533045 -2.4093186017926205 1.5692209777841764 1.19567905031086 0.08276325717685079 -
0.10053882578929706 -0.3750623159288322 -0.023841736554610356
1.6077395606205696 -0.04979059593238454 0.05545236241312816 -2.050827282340712 -0.038057479931813895
2.7314636384400606 0.0621089209933646 -2.520042453088982 1.6787370971513154 0.014589040863996896 -
0.026092775177453207 0.004630525433094241 0.04475965644686002
-0.7690197124128048 -0.2809034831800579 -0.6919131312848309 0.425638116628382 -0.9578575668912771
0.5258728878442835 1.3642064064465487 -0.5814229260282106 -0.7416025252569921 -0.2112277745932927
0.1374457596308841 0.3242486913111365 -0.04969549164971736
-4.3601288190008844 0.0326867623298655 -0.4572803297467024 -0.33274696775405693 -0.26145005507079555
1.6331052298364763 2.19881936121678 0.06499309497213597 0.2848516621138459 -0.6458758003998314
0.024346869568991975 -0.22837933063794427 -0.24834066469162608
# bias neuron1 neuron2 neuron3 neuron4 neuron5

```

```
1.468201298006802 0.06116052624682594 0.33858438067795643 -0.39458071876841083 0.46139774610556217
1.878696702247
# min/max form denormalisation of outputs
3.614536369589416e-05 14.238223860863263
# min, max and tolerance for output
0 8 0.2
```

Figure 3-2. Example of parameters text file for LAI generation using S2 acquisitions

The 3 steps of the BI algorithm implementation are detailed here below.

3.2.1 Normalization of the inputs

The input variables (9 S2 bands and 3 observation angles in the case of S2 and 5 bands and 3 observation angles in the case of L8) should be normalized before running the neural network, using the following equation:

$$X^* = 2 \times \left(\frac{X - X_{min}}{X_{max} - X_{min}} \right) - 1$$

where X^* is the normalized input value, and X_{min} and X_{max} are the minimum and maximum values computed over the neural network training dataset.

3.2.2 Neural network

The neural network is described by its architecture, i.e. the number of hidden layers and the output layer. Each layer is described by its number of neurons, associated weight and biases, and transfer function.

For the neurons of the hidden layers, the transfer function is a tangent sigmoid function given by:

$$Y = Tansig(x) = \left(\frac{2}{1 + \exp(-2x)} \right) - 1$$

For the output layer, the transfer function is linear ($y=x$).

For each neural net (one per product and per sensor), the weight and biases for each neuron are provided in the form of tables.

3.2.3 Denormalization of the output

This last step simply consists in applying the inverse function than the one used for input normalization:

$$Y = 0.5 \times (Y^* + 1) \times (Y_{max} - Y_{min}) + Y_{min}$$

3.2.4 Pseudo-code

The generic C++ code provided in Code 3-2 can be used to run the 3 steps in a row, similarly for S2 and L8 acquisitions.

```
#include "itkImageRegionIterator.h"
#include "belcamApplyTrainedNeuralNetworkFilter.h"

template <class TI, class TO>
void belcamApplyTrainedNeuralNetworkFilter<TI, TO>
::ThreadedGenerateData(const typename TO::RegionType& outputRegionForThread, itk::ThreadIdType threadId)
{
```

```

unsigned int nbBands = this->GetInput(0)->GetNumberOfComponentsPerPixel();
unsigned int nbLayers = m_net.layers.size();

// initialize iterators
typename TI::RegionType inputRegionForThread;
this->CallCopyOutputRegionToInputRegion(inputRegionForThread, outputRegionForThread);
itk::ImageRegionIterator<TO> oIt(this->GetOutput(), outputRegionForThread);
itk::ImageRegionConstIterator<TI> iIt(this->GetInput(), inputRegionForThread);

typename TO::PixelType pixel0(1);
typename TI::PixelType pixelI(nbBands);

if(m_net.layers[0].weights[0].size() != nbBands) {
    printf("the number of bands and the number of inputs in txt file have to be identical !! (%d vs %d)\n",
nbBands, m_net.layers[0].weights[0].size());
    exit(1);
}

// init size of processing variables
std::vector<std::vector<double> > inputs(nbLayers+1);
for(int iN = 0; iN < nbLayers; ++iN)
    for(int i = 0; i < m_net.layers[iN].weights[0].size(); ++i)
        inputs[iN].push_back(0.);
inputs[nbLayers].push_back(0.); // one output

// loop over pixels
for(oIt.GoToBegin(), iIt.GoToBegin(); ! oIt.IsAtEnd(); ++oIt, ++iIt)
{
    // normalisation of inputs
    pixelI = iIt.Get();
    for(int iB = 0; iB < nbBands; ++iB)
        inputs[0][iB] = 2.0 * (pixelI[iB] / m_scale - m_net.min_norm_in[iB]) / (m_net.max_norm_in[iB] -
m_net.min_norm_in[iB]) - 1.0;

    // loop over layers
    for(int iL = 0; iL < nbLayers; ++iL) {
        trained_network_layer & lay = m_net.layers[iL];

        // loop over neurons (apply layer rules)
        for(int iN = 0; iN < lay.weights.size(); ++iN) {
            inputs[iL+1][iN] = lay.bias[iN];
            for(int iB = 0; iB < lay.weights[iN].size(); ++iB)
                inputs[iL+1][iN] += inputs[iL][iB] * lay.weights[iN][iB];
            inputs[iL+1][iN] = (*lay.func)(inputs[iL+1][iN]);
        }
    }

    // de-normalisation of output
    pixel0[0] = 0.5 * (inputs[nbLayers][0] + 1.0) * (m_net.max_norm_out - m_net.min_norm_out) +
m_net.min_norm_out;

    oIt.Set( pixel0 );
}
}

template <class TI, class TO>
void belcamApplyTrainedNeuralNetworkFilter<TI, TO>
::GenerateOutputInformation()
{
    Superclass::GenerateOutputInformation();
    this->GetOutput()->SetNumberOfComponentsPerPixel( 1 );
}

```

Code 3-2. BI retrieval implementation, including inputs normalization, neural network run and outputs denormalization

4 Output

4.1 Index timeseries

The outputs are provided over each valid pixel of each S2 or L8 L2A images. One image is produced for each BI: LAI, FAPAR and FCOVER and for each SI: NDVI, NDWI and Brightness.

The spatial resolution of the output is the finer spatial resolution available, i.e. 10 meters for S2 (even if 20 meters bands are used, they are re-scaled to 10 meters) and 30 meters for L8.

The non-valid pixels (snow/cloud/cloud shadow) should be masked and replaced with no data. Water mask is sometimes too approximative, pixels flagged with water should not be masked.

4.2 Quality flags

In addition to the output index values, two quality flags are generated:

- the status of the input L2A pixels (no data / cloud / snow / water / land);
- the definition domain of the input spectral bands and of the output variables.

4.2.1 Status flag

This first flag gives the status of each pixel (no data, cloud, snow, water, land). It is directly derived from the similar quality flag of the corresponding S2 or L8 L2A image.

4.2.2 Input/output values range

Several quality flags shall also be generated to inform about the validity of the input reflectance values and the output index values. The bands values are put to 0 if the values are valid and to 1 if the values are invalid, i.e. out of range. The two following sections defined the validity range of the input and outputs.

4.2.2.1 Input definition domains

The definition domain of the input reflectance values for each spectral band are presented in Table 4-1 for S2 and in Table 4-2 for L8. These definition domains are also provided to the system in the Excel file along with the algorithm parameters.

Table 4-1. Definition domain of the S2 reflectance data

Acronym	Minimum value	Maximum value
B3	0	0.242
B4	0	0.292
B5	0	0.335
B6	0	0.608
B7	0	0.759

B8a	0	0.776
B8	0	0.790
B11	0	0.495
B12	0	0.494

Table 4-2. Definition domain of the L8 reflectance data

Acronym	Minimum value	Maximum value
B3	0	0.300
B4	0	0.266
B8a	0	0.788
B11	0	0.476
B12	0	0.504

4.2.2.2 Output definition domain and tolerance

In the situation where the retrieved index values are outside of their definition domain (given in Table 4-3), the band 2 value of the quality flag is set to 1 (invalid value) and the product value is adjusted and set to the closest bound of the domain (i.e. either the minimum or the maximum bound). In the event that the output value is within a tolerance interval around the bound, the initial value is kept; yet, the “output out of range” flag is raised.

Table 4-3. Definition domain for the output index values

Index	Tolerance	Minimum value	Maximum value
Biophysical Indicators			
LAI	0.2	0	8.00
FAPAR	0.1	0	0.94
FCOVER	0.1	0	1.00
Spectral Indices			
NDVI		-1.00	1.00
NDWI		-1.00	1.00
Brightness			