

Introduction

This document provides the codes written for this study¹ and available GitHub repository <https://github.com/rdandrimont/AGREE> (AGRicultural Enhanced Evidence) with a MIT license:

- 1_Sentinel1-VV-VH_7-day_ParcelAverage.js
- 2_A_SNAP_TOPSAR_Coherence_Single_Swath.xml
- 2_B_SNAP_TOPSAR_merge_ML_TC.xml
- 3_Sentinel2_CloudAndShadowMask_BSI-sumAndNDVItrend_ParcelAverage.js
- 4_Sentinel1-Coherence_15-day-max_ParcelAverage.js
- 5_TensorFlow_Classification.ipynb

For this study, 5 different scripts are available (Figure 1). See the paper for detailed information.

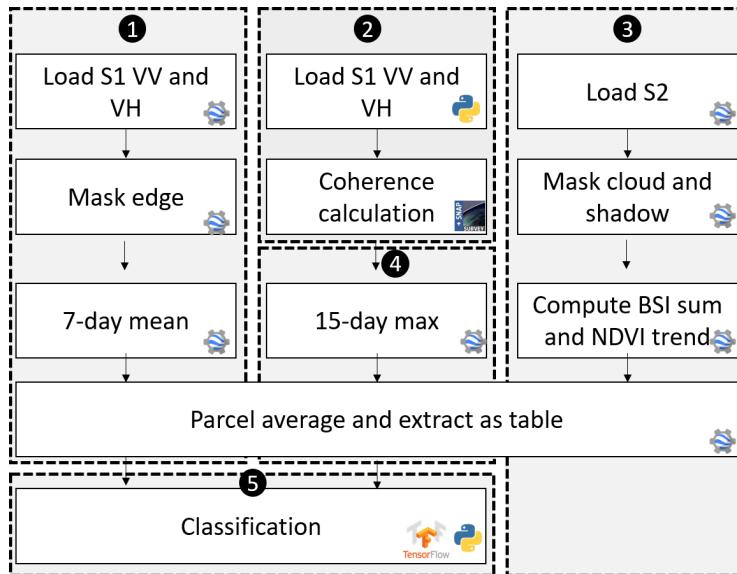


Figure 1: Code developed in this study are open and available to public. They are divided in 7 chunks using different format and library including JavaScript for GEE, Python, SNAP and TensorFlow.

¹Raphaël d'Andrimont, Guido Lemoine, Marijn van der Velde. Targeted Grassland Parcel-Level Monitoring with Sentinels, Street-Level Imagery, and Field Observations, *Remote Sensing*, 2018

1 GEE code to process Sentinel-1 : Backscattering

```
1  ***** Start of imports. If edited, may not auto-convert in the
2   → playground. *****
3  var brp2017 =
4   ee.FeatureCollection("users/gglemoine62/BRP_gewaspercelen_2017_concept");
5  ***** End of imports. If edited, may not auto-convert in the
6   → playground. *****
7
8 // A / FUNCTIONS
9
10 // Functions to convert from/to dB
11 function toNatural(img) {
12   return
13     ee.Image(10.0).pow(img.select('..').divide(10.0)).copyProperties(img,
14       ['system:time_start'])
15 }
16
17 // Remove ugly edges
18 function maskEdge(img) {
19   var mask = img.select(0).unitScale(-25,
20     5).multiply(255).toByte().connectedComponents(ee.Kernel.rectangle(1,1),
21     100);
22   return img.updateMask(mask.select(0));
23 }
24
25 // B / LOAD INPUTS
26
27
28 // 1. Date
29 var start_date = '2017-04-01'
30 var end_date = '2017-08-01'
31
32 // 2. Get Netherlands municipalities
33 var gemeenten =
34   ee.FeatureCollection('ft:1B3v8wxCk01aGd8jF4byitKEjolHvQFyMF9nZFsA8');
```

```

35 // 3. Set the AOI to the collections of municipalities of interest
36 // (1) Gelderse Vallei area
37 var aoi = gemeenten.filter(ee.Filter.inList('gemnaam', ['Ede',
    ↵ 'Wageningen', 'Renkum', 'Barneveld', 'Arnhem', 'Putten',
    ↵ 'Nijkerk']));
38 // (2) Utrecht/Groene Hart area
39 // var aoi = gemeenten.filter(ee.Filter.inList('gemnaam', ['De
    ↵ Ronde Venen', 'Woerden', 'Breukelen', 'Maarssen', 'Soest',
    ↵ 'Zeist', 'Baarn', 'De Bilt']));
40
41 var step = 7 // in days (time window for meaning)
42
43
44 ///////////////////////////////////////////////////////////////////
45 // C / CODE
46 ///////////////////////////////////////////////////////////////////
47
48 brp2017 = brp2017.map(function(f) { return f.set({'id': f.id(),
    ↵ 'area': f.area(), 'perimeter': f.perimeter()}) })
49 brp2017 = brp2017.filterMetadata('area', 'less_than', 10000000)
    ↵ // Remove degenerate right winding polygons
50
51 // Internally buffer parcels to avoid boundary pixels
52 brp2017 = brp2017.map(function(f) { return f.buffer(-10) })
53 brp2017 = brp2017.map(function(f) { return f.set({'bufferedarea':
    ↵ f.area()}) })
54
55 // Clip to AOI
56 brp2017 = brp2017.filterBounds(aoi)
57
58 // get the data from S1 (VV pol.)
59 var s1 =
    ↵ ee.ImageCollection('COPERNICUS/S1_GRD').filterMetadata('instrumentMode',
    ↵ 'equals', 'IW').
        filter(ee.Filter.eq('transmitterReceiverPolarisation', ['VV',
            ↵ 'VH'])).filterBounds(aoi).filterDate(start_date, end_date).
        sort('system:time');
56
57
58 // Remove ugly edges
59 s1 = s1.map(maskEdge)
60
61
62 // Extracts are made from natural (non-logarithmic) values
63 s1 = s1.map(toNatural)
64
65
66 // Olha Danylo's procedure to create weekly means (adapted)

```

```

71
72 var days = ee.List.sequence(0,
    ↳ ee.Date(end_date).difference(ee.Date(start_date), 'day'),
    ↳ step).
73   map(function(d) { return ee.Date(start_date).advance(d, "day")
    ↳ })
74
75 var dates = days.slice(0,-1).zip(days.slice(1))
76
77 var s1res = dates.map(function(range) {
78   var dstamp = ee.Date(ee.List(range).get(0)).format('YYYYMMdd')
79   var temp_collection = s1.filterDate(ee.List(range).get(0),
80     ee.List(range).get(1)).mean().select(['VV', 'VH'],
81     ↳ [ee.String('VV_').cat(dstamp),
82       ↳ ee.String('VH_').cat(dstamp)])
83   return temp_collection
84 })
85
86 // Convert ImageCollection to image stack
87 function stack(i1, i2)
88 {
89   return ee.Image(i1).addBands(ee.Image(i2))
90 }
91
92 var s1stack = s1res.slice(1).iterate(stack, s1res.get(0))
93 s1stack = ee.Image(s1stack).clip(aoi)
94
95 // Export the parcel means for this image stack for use in
96 // tensorflow runs
97 Export.table.toDrive(ee.Image(s1stack).reduceRegions({collection:
98   ↳ brp2017, reducer: ee.Reducer.mean(), scale: 10}).
99   select(ee.List(['id', 'area', 'bufferedarea', 'perimeter',
100    ↳ 'gws_gewasc',
101    ↳ 'gws_gewas']).cat(ee.Image(s1stack).bandNames()), null,
102    ↳ false), "Landsense_GH_Dump_Region1")
103
104 // Optionally, display some combination to see if all went well
105 Map.addLayer(toDB(ee.Image(s1res.get(0)).addBands(ee.Image(s1res.get(s1res.size().divide(2)
106    {bands: ['VV_20170401', 'VV_20170527', 'VV_20170722'], min:
107      ↳ -25, max: 0 }, "S1 VV stack")
108
109 Map.addLayer(toDB(ee.Image(s1res.get(0)).addBands(ee.Image(s1res.get(s1res.size().divide(2)
110    {bands: ['VH_20170401', 'VH_20170527', 'VH_20170722'], min:
111      ↳ -30, max: -5 }, "S1 VH stack"))

```

2 Python code to obtain Coherence

```
1 <graph id="Graph">
2   <version>1.0</version>
3   <node id="Read">
4     <operator>Read</operator>
5     <sources/>
6     <parameters class="com.bc.ceres.binding.dom.XppDomElement">
7       <file>$infile_master</file>
8       <formatName>SENTINEL-1</formatName>
9     </parameters>
10    </node>
11    <node id="Read(2)">
12      <operator>Read</operator>
13      <sources/>
14      <parameters class="com.bc.ceres.binding.dom.XppDomElement">
15        <file>$infile_slave</file>
16        <formatName>SENTINEL-1</formatName>
17      </parameters>
18    </node>
19    <node id="TOPSAR-Split">
20      <operator>TOPSAR-Split</operator>
21      <sources>
22        <sourceProduct refid="Apply-Orbit-File"/>
23      </sources>
24      <parameters class="com.bc.ceres.binding.dom.XppDomElement">
25        <subswath>IW$sub</subswath>
26        <selectedPolarisations>VH,VV</selectedPolarisations>
27        <firstBurstIndex>1</firstBurstIndex>
28        <lastBurstIndex>9</lastBurstIndex>
29        <wktAoi/>
30      </parameters>
31    </node>
32    <node id="TOPSAR-Split(2)">
33      <operator>TOPSAR-Split</operator>
34      <sources>
35        <sourceProduct refid="Apply-Orbit-File(2)"/>
36      </sources>
37      <parameters class="com.bc.ceres.binding.dom.XppDomElement">
38        <subswath>IW$sub</subswath>
39        <selectedPolarisations>VH,VV</selectedPolarisations>
40        <firstBurstIndex>1</firstBurstIndex>
41        <lastBurstIndex>9</lastBurstIndex>
42        <wktAoi/>
43      </parameters>
44    </node>
```

```

45 <node id="Apply-Orbit-File">
46   <operator>Apply-Orbit-File</operator>
47   <sources>
48     <sourceProduct refid="Read"/>
49   </sources>
50   <parameters class="com.bc.ceres.binding.dom.XppDomElement">
51     <orbitType>Sentinel Restituted (Auto Download)</orbitType>
52     <polyDegree>3</polyDegree>
53     <continueOnFail>true</continueOnFail>
54   </parameters>
55 </node>
56 <node id="Apply-Orbit-File(2)">
57   <operator>Apply-Orbit-File</operator>
58   <sources>
59     <sourceProduct refid="Read(2)"/>
60   </sources>
61   <parameters class="com.bc.ceres.binding.dom.XppDomElement">
62     <orbitType>Sentinel Restituted (Auto Download)</orbitType>
63     <polyDegree>3</polyDegree>
64     <continueOnFail>true</continueOnFail>
65   </parameters>
66 </node>
67 <node id="Back-Geocoding">
68   <operator>Back-Geocoding</operator>
69   <sources>
70     <sourceProduct refid="TOPSAR-Split"/>
71     <sourceProduct.1 refid="TOPSAR-Split(2)"/>
72   </sources>
73   <parameters class="com.bc.ceres.binding.dom.XppDomElement">
74     <demName>SRTM 1Sec HGT</demName>
75
76     ↳ <demResamplingMethod>BILINEAR_INTERPOLATION</demResamplingMethod>
77     <externalDEMfile/>
78     <externalDEMNoDataValue>0.0</externalDEMNoDataValue>
79     <resamplingType>BILINEAR_INTERPOLATION</resamplingType>
80
81     ↳ <maskOutAreaWithoutElevation>false</maskOutAreaWithoutElevation>
82     <outputRangeAzimuthOffset>false</outputRangeAzimuthOffset>
83     <outputDerampDemodPhase>false</outputDerampDemodPhase>
84     <disableReramp>false</disableReramp>
85   </parameters>
86 </node>
87 <node id="TOPSAR-Deburst">
88   <operator>TOPSAR-Deburst</operator>
89   <sources>
90     <sourceProduct refid="Coherence"/>

```

```

89      </sources>
90      <parameters class="com.bc.ceres.binding.dom.XppDomElement">
91          <selectedPolarisations/>
92      </parameters>
93  </node>
94  <node id="Coherence">
95      <operator>Coherence</operator>
96      <sources>
97          <sourceProduct refid="Back-Geocoding"/>
98      </sources>
99      <parameters class="com.bc.ceres.binding.dom.XppDomElement">
100          <cohWinAz>3</cohWinAz>
101          <cohWinRg>10</cohWinRg>
102          <subtractFlatEarthPhase>false</subtractFlatEarthPhase>
103          <srpPolynomialDegree>5</srpPolynomialDegree>
104          <srpNumberPoints>501</srpNumberPoints>
105          <orbitDegree>3</orbitDegree>
106          <squarePixel>true</squarePixel>
107      </parameters>
108  </node>
109  <node id="Write">
110      <operator>Write</operator>
111      <sources>
112          <sourceProduct refid="TOPSAR-Deburst"/>
113      </sources>
114      <parameters class="com.bc.ceres.binding.dom.XppDomElement">
115          <file>$outfile</file>
116          <formatName>BEAM-DIMAP</formatName>
117      </parameters>
118  </node>
119  <applicationData id="Presentation">
120      <Description/>
121      <node id="Read">
122          <displayPosition x="9.0" y="68.0"/>
123      </node>
124      <node id="Read(2)">
125          <displayPosition x="9.0" y="192.0"/>
126      </node>
127      <node id="TOPSAR-Split">
128          <displayPosition x="161.0" y="101.0"/>
129      </node>
130      <node id="TOPSAR-Split(2)">
131          <displayPosition x="160.0" y="162.0"/>
132      </node>
133      <node id="Apply-Orbit-File">
134          <displayPosition x="10.0" y="102.0"/>

```

```

135   </node>
136   <node id="Apply-Orbit-File(2)">
137     <displayPosition x="8.0" y="161.0"/>
138   </node>
139   <node id="Back-Geocoding">
140     <displayPosition x="325.0" y="102.0"/>
141   </node>
142   <node id="TOPSAR-Deburst">
143     <displayPosition x="766.0" y="100.0"/>
144   </node>
145   <node id="Coherence">
146     <displayPosition x="545.0" y="104.0"/>
147   </node>
148   <node id="Write">
149     <displayPosition x="1046.0" y="123.0"/>
150   </node>
151 </applicationData>
152 </graph>

1 <graph id="Graph">
2   <version>1.0</version>
3   <node id="Read">
4     <operator>Read</operator>
5     <sources/>
6     <parameters class="com.bc.ceres.binding.dom.XppDomElement">
7       <file>${file_path}_coh_IW1.dim</file>
8     </parameters>
9   </node>
10  <node id="Read(2)">
11    <operator>Read</operator>
12    <sources/>
13    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
14      <file>${file_path}_coh_IW2.dim</file>
15    </parameters>
16  </node>
17  <node id="Read(3)">
18    <operator>Read</operator>
19    <sources/>
20    <parameters class="com.bc.ceres.binding.dom.XppDomElement">
21      <file>${file_path}_coh_IW3.dim</file>
22    </parameters>
23  </node>
24  <node id="TOPSAR-Merge">
25    <operator>TOPSAR-Merge</operator>
26    <sources>
27      <sourceProduct refid="Read"/>

```

```

28      <sourceProduct.1 refid="Read(2)"/>
29      <sourceProduct.2 refid="Read(3)"/>
30  </sources>
31  <parameters class="com.bc.ceres.binding.dom.XppDomElement">
32      <selectedPolarisations/>
33  </parameters>
34 </node>
35 <node id="Multilook">
36     <operator>Multilook</operator>
37     <sources>
38         <sourceProduct refid="TOPSAR-Merge"/>
39     </sources>
40     <parameters class="com.bc.ceres.binding.dom.XppDomElement">
41         <sourceBands/>
42         <nRgLooks>4</nRgLooks>
43         <nAzLooks>1</nAzLooks>
44         <outputIntensity>true</outputIntensity>
45         <grSquarePixel>true</grSquarePixel>
46     </parameters>
47 </node>
48 <node id="Terrain-Correction">
49     <operator>Terrain-Correction</operator>
50     <sources>
51         <sourceProduct refid="Multilook"/>
52     </sources>
53     <parameters class="com.bc.ceres.binding.dom.XppDomElement">
54         <sourceBands/>
55         <demName>SRTM 1Sec HGT</demName>
56         <externalDEMFile/>
57         <externalDEMNoDataValue>0.0</externalDEMNoDataValue>
58         <externalDEMApplyEGM>true</externalDEMApplyEGM>
59
60         ← <demResamplingMethod>BILINEAR_INTERPOLATION</demResamplingMethod>
61
62         ← <imgResamplingMethod>BILINEAR_INTERPOLATION</imgResamplingMethod>
63         <pixelSpacingInMeter>20.0</pixelSpacingInMeter>
64
65         ← <pixelSpacingInDegree>1.796630568239043E-4</pixelSpacingInDegree>
66         <mapProjection>PROJCS["UTM Zone 31 / World Geodetic
67             ← System 1984",
68             GEOGCS["World Geodetic System 1984",
69                 DATUM["World Geodetic System 1984",
70                     SPHEROID["WGS 84", 6378137.0, 298.257223563,
71                         ← AUTHORITY["EPSG", "7030"]],
72                     AUTHORITY["EPSG", "6326"]],
```

```

68     PRIMEM[&quot;Greenwich&quot;, 0.0,
69         ↳ AUTHORITY[&quot;EPSG&quot;, &quot;8901&quot;]], 
70     UNIT[&quot;degree&quot;, 0.017453292519943295],
71     AXIS[&quot;Geodetic longitude&quot;, EAST],
72     AXIS[&quot;Geodetic latitude&quot;, NORTH]],
73     PROJECTION[&quot;Transverse_Mercator&quot;],
74     PARAMETER[&quot;central_meridian&quot;, 3.0],
75     PARAMETER[&quot;latitude_of_origin&quot;, 0.0],
76     PARAMETER[&quot;scale_factor&quot;, 0.9996],
77     PARAMETER[&quot;false_easting&quot;, 500000.0],
78     PARAMETER[&quot;false_northing&quot;, 0.0],
79     UNIT[&quot;m&quot;, 1.0],
80     AXIS[&quot;Easting&quot;, EAST],
81     AXIS[&quot;Northing&quot;, NORTH]]</mapProjection>
82         <nodataValueAtSea>true</nodataValueAtSea>
83         <saveDEM>false</saveDEM>
84         <saveLatLon>false</saveLatLon>
85
86         ↳ <saveIncidenceAngleFromEllipsoid>false</saveIncidenceAngleFromEllipsoid>
87         <saveLocalIncidenceAngle>false</saveLocalIncidenceAngle>
88
89         ↳ <saveProjectedLocalIncidenceAngle>false</saveProjectedLocalIncidenceAngle>
90         <saveSelectedSourceBand>true</saveSelectedSourceBand>
91         <outputComplex>false</outputComplex>
92
93         ↳ <applyRadiometricNormalization>false</applyRadiometricNormalization>
94         <saveSigmaNought>false</saveSigmaNought>
95         <saveGammaNought>false</saveGammaNought>
96         <saveBetaNought>false</saveBetaNought>
97         <incidenceAngleForSigma0>Use projected local incidence
98             ↳ angle from DEM</incidenceAngleForSigma0>
99         <incidenceAngleForGamma0>Use projected local incidence
100            ↳ angle from DEM</incidenceAngleForGamma0>
101         <auxFile>Latest Auxiliary File</auxFile>
102         <externalAuxFile/>
103
104     </parameters>
105 </node>
106 <node id="Write">
107     <operator>Write</operator>
108     <sources>
109         <sourceProduct refid="Terrain-Correction"/>
110     </sources>
111     <parameters class="com.bc.ceres.binding.dom.XppDomElement">
112         <file>${file_path}_coh_mrg_ML_TC.tif</file>
113         <formatName>GeoTIFF-BigTIFF</formatName>
114     </parameters>

```

```

108     </node>
109     <applicationData id="Presentation">
110         <Description/>
111         <node id="Read">
112             <displayPosition x="45.0" y="54.0"/>
113         </node>
114         <node id="Read(2)">
115             <displayPosition x="43.0" y="117.0"/>
116         </node>
117         <node id="Read(3)">
118             <displayPosition x="42.0" y="185.0"/>
119         </node>
120         <node id="TOPSAR-Merge">
121             <displayPosition x="230.0" y="119.0"/>
122         </node>
123         <node id="Multilook">
124             <displayPosition x="455.0" y="120.0"/>
125         </node>
126         <node id="Terrain-Correction">
127             <displayPosition x="610.0" y="121.0"/>
128         </node>
129         <node id="Write">
130             <displayPosition x="848.0" y="121.0"/>
131         </node>
132     </applicationData>
133 </graph>
```

3 GEE code to process Sentinel-2

```

1  ***** Start of imports. If edited, may not auto-convert in the
2      → playground. *****
3  var brp2017 =
4      → ee.FeatureCollection("users/gglemoine62/BRP_gewaspercelen_2017_concept"),
5      natura2000 =
6          → ee.FeatureCollection("users/rdandrimont/Natura2000_end2016_NL");
7  ***** End of imports. If edited, may not auto-convert in the
8      → playground. *****
9
10 // ADD A NDVI BAND
11 function addNdvi(img) {
12     var nd = img.normalizedDifference(['nir', 'red']);
```

```

13     return img.addBands(nd.float().rename('NDVI'));
14 }
15
16
17 function addBSI(img) {
18     var bsi =
19         img.expression('((swir1+red)-(nir+blue))/(((swir1+red)+(nir+blue))*1.0)', {
20             'swir1': img.select('swir1'),
21             'red': img.select('red'),
22             'nir': img.select('nir'),
23             'blue': img.select('blue')
24         });
25     return img.addBands(bsi.float().rename('BSI'));
26 }
27
28
29 function addBare(img) {
30     var bsi = img.expression('BSI>0', {
31         'BSI': img.select('BSI'),
32     });
33     return img.addBands(bsi.int().rename('Bare'));
34 }
35
36
37
38 // CONVERT SENTINEL TO TOA
39 function sentinel2toa(img) {
40     var toa =
41         img.select(['B1', 'B2', 'B3', 'B4', 'B6', 'B8', 'B8A', 'B9', 'B10',
42                     'B11', 'B12'],
43                     ['aerosol', 'blue', 'green', 'red',
44                     'red2', 'nir', 'red4', 'h2o',
45                     'cirrus', 'swirl1', 'swirl2'])
46         .divide(10000)
47         .addBands(img.select(['QA60']))
48
49         .set('solar_azimuth', img.get('MEAN_SOLAR_AZIMUTH_ANGLE'))
50
51         .set('solar zenith', img.get('MEAN_SOLAR_ZENITH_ANGLE'))
52
53         .set('date', ee.Date(img.get('system:time_start')))
54
55         .set('system:time_start', img.get('system:time_start')));

```

```

49     return toa
50 }
51
52 // FLAG CLOUD AND CIRRUS
53 function ESAcloud(toa) {
54     // author: Nick Clinton
55     var qa = toa.select('QA60');
56
57     // Bits 10 and 11 are clouds and cirrus, respectively.
58     var cloudBitMask = Math.pow(2, 10);
59     var cirrusBitMask = Math.pow(2, 11);
60
61     // clear if both flags set to zero.
62     var clear = qa.bitwiseAnd(cloudBitMask).eq(0).and(
63         qa.bitwiseAnd(cirrusBitMask).eq(0));
64     var cloud = clear.eq(0)
65     return cloud
66 }
67
68 // FLAG SHADOW
69 function shadowMask(toa,cloud){
70     // Author: Gennadii Donchyts
71     // License: Apache 2.0
72
73     // solar geometry (radians)
74     var azimuth
75         ← =ee.Number(toa.get('solar_azimuth')).multiply(Math.PI).divide(180.0).add(ee.Number(0.5));
76     var zenith =ee.Number(0.5).multiply(Math.PI
77         ← ).subtract(ee.Number(toa.get('solar_zenith')).multiply(Math.PI).divide(180.0));
78
79     // find where cloud shadows should be based on solar geometry
80     var nominalScale = cloud.projection().nominalScale();
81     var cloudHeights = ee.List.sequence(200,10000,500);
82     var shadows = cloudHeights.map(function(cloudHeight){
83         cloudHeight = ee.Number(cloudHeight);
84         var shadowVector = zenith.tan().multiply(cloudHeight);
85         var x =
86             ← azimuth.cos().multiply(shadowVector).divide(nominalScale).round();
87         var y =
88             ← azimuth.sin().multiply(shadowVector).divide(nominalScale).round();
89         return cloud.changeProj(cloud.projection(),
90             ← cloud.projection().translate(x, y));
91     });
92     var potentialShadow =
93         ← ee.ImageCollection.fromImages(shadows).max();
94
95     return potentialShadow
96 }

```

```

89     // shadows are not clouds
90     var potentialShadow = potentialShadow.and(cloud.not());
91
92     // (modified by Sam Murphy) dark pixel detection
93     var darkPixels = toa.normalizedDifference(['green',
94         'swir2']).gt(0.25).rename(['dark_pixels']);
95
96     // shadows are dark
97     var shadow = potentialShadow.and(darkPixels).rename('shadows');
98
99     return shadow
100 }
101
102 // CONVERT TO TOA, MASK SHADOW AND MASK CLOUD
103 function cloud_and_shadow_mask(img) {
104     var toa = sentinel2toa(img)
105     var cloud = ESAcloud(toa)
106     var shadow = shadowMask(toa,cloud)
107     var mask = cloud.or(shadow).eq(0)
108     return toa.updateMask(mask)
109 }
110
111 // This function adds a band representing the image timestamp.
112 var addTime = function(image) {
113     return image.addBands(image.metadata('system:time_start')
114         .divide(1000 * 60 * 60 * 24 * 365));
115 };
116
117 /////////////////
118 // B / LOAD INPUTS
119 /////////////////
120
121 // 1. Date
122 var startDateStr='2017-01-01'
123 var stopDateStr='2017-08-01'
124
125 // 2. Get Netherlands municipalities
126 var gemeenten =
127     ee.FeatureCollection('ft:1B3v8wxCk01aGd8jF4byitKEjolHvQFyMF9nZFSA8');
128
129 // 3. Set the AOI to the collections of municipalities of interest
130 // (1) Gelderse Vallei area
131 var aoi = gemeenten.filter(ee.Filter.inList('gemnaam', ['Ede',
132     'Wageningen', 'Renkum', 'Barneveld', 'Arnhem', 'Putten',
133     'Nijkerk']));

```

```

131 // (2) Utrecht/Groene Hart area
132 // var aoi = gemeenten.filter(ee.Filter.inList('gemnaam', ['De
133   ↵ Ronde Venen', 'Woerden', 'Breukelen', 'Maarssen', 'Soest',
134   ↵ 'Zeist', 'Baarn', 'De Bilt']));
135 ///////////////////////////////////////////////////
136 // C / CODE
137 ///////////////////////////////////////////////////
138
139 // C.0 LOAD PARCELS
140 brp2017 = brp2017.map(function(f) { return f.set({'id': f.id(),
141   ↵ 'area': f.area(), 'perimeter': f.perimeter()}) })
142 brp2017 = brp2017.filterMetadata('area', 'less_than', 10000000)
143   ↵ // Remove degenerate right winding polygons
144
145 // Internally buffer parcels to avoid boundary pixels
146 brp2017 = brp2017.map(function(f) { return f.buffer(-10)})
147 brp2017 = brp2017.map(function(f) { return f.set({'bufferedarea':
148   ↵ f.area()}) })
149
150
151 // C.1 COMPOSITE
152 ///////////////////////////////////////////////////
153 // LOAD Sentinel-2 collections for a given temporal window
154 var startDate = ee.Date(startDateStr)
155 var stopDate = ee.Date(stopDateStr)
156 var images = ee.ImageCollection('COPERNICUS/S2')
157   .filterDate(startDate, stopDate).filterBounds(aoi);
158   ↵ // .limit(10);
159
160 // CONVERT TO TOA AND APPLY THE SHADOW AND CLOUD MASK
161 var s2_cleaned = images.map(cloud_and_shadow_mask);
162
163 // C.2 ADD BANDS
164 ///////////////////////////////////////////////////
165
166 // ADD NDVI BANDS
167 var s2_cleaned = s2_cleaned.map(addNdvi);
168
169 // ADD BSI BAND
170 var s2_cleaned = s2_cleaned.map(addBSI);
171

```

```

172 // ADD Bare BAND
173 var s2_cleaned = s2_cleaned.map(addBare);
174
175 // C.3 BSI SUM
176 /////////////////
177 // SUM OF BARE EVENT
178 var result = s2_cleaned.select('Bare').reduce('sum') ;
179
180 // C.4 / LINEAR REGRESSION
181 /////////////////
182
183 // USE NDVI and TIME
184 var s2_ts = s2_cleaned.map(
185   function(image) {
186     // Rename that band to something appropriate
187     return image.select(['NDVI']).set('system:time_start',
188       → image.get('system:time_start'));
189   }
190 );
191 var s2_ts = s2_ts.map(addTime);
192
193 // Compute the linear trend over time.
194 var trend = s2_ts.select(['system:time_start',
195   → 'NDVI']).reduce(ee.Reducer.linearFit());
196
197 // C.5 /Combine all results
198 /////////////////
199 var s2stack =
200   → result.addBands(trend.select('scale')).addBands(trend.select('offset'))
201
202 // C.6 Export the parcel means for this image stack for use in
203   → tensorflow runs
204 ///////////////
205 Export.table.toDrive(ee.Image(s2stack).reduceRegions({collection:
206   → brp2017, reducer: ee.Reducer.mean(), scale: 10}).
207   select(ee.List(['id', 'area', 'bufferedarea', 'perimeter',
208     → 'gws_gewasc',
209     → 'gws_gewas']).cat(ee.Image(s2stack).bandNames()), null,
210     → false), "Landsense_S2_Dump_Region1")

```

4 GEE code to process Sentinel-1 : Coherence

```
1  ***** Start of imports. If edited, may not auto-convert in the
2    ↳ playground. *****/
3  var brp2017 =
4    ee.FeatureCollection("users/gglemoine62/BRP_gewaspercelen_2017_concept"),
5    s1coh = ee.ImageCollection("users/gglemoine62/nl_coh");
6  ***** End of imports. If edited, may not auto-convert in the
7    ↳ playground. *****/
8  brp2017 = brp2017.map(function(f) { return f.set({'id': f.id(),
9    'area': f.area(), 'perimeter': f.perimeter()}) })
10
11 s1coh = s1coh.map(function(f) {
12   return f.set('system:time_start', ee.Date.parse('YYYYMMdd',
13     ee.String(f.id()).split('_').get(3)))
14 })
15
16 brp2017 = brp2017.filterMetadata('area', 'less_than', 10000000)
17   // Remove degenerate right winding polygons
18 brp2017 = brp2017.map(function(f) { return f.buffer(-10)})
19 brp2017 = brp2017.map(function(f) { return f.set({'bufferedarea':
20   f.area()}) })
21
22 var gemeenten =
23   ee.FeatureCollection('ft:1B3v8wxCk01aGd8jF4byitKEjolHvQFyMF9nZFsA8');
24
25 var aoi = gemeenten.filter(ee.Filter.inList('gemnaam', ['Ede',
26   'Wageningen', 'Renkum', 'Barneveld', 'Arnhem', 'Putten',
27   'Nijkerk']));
28 // var aoi = gemeenten.filter(ee.Filter.inList('gemnaam', ['De
29   Ronde Venen', 'Woerden', 'Breukelen', 'Maarssen', 'Soest',
30   'Zeist', 'Baarn', 'De Bilt']));
31
32 brp2017 = brp2017.filterBounds(aoi)
33
34 Map.centerObject(aoi, 12);
35
36 var start_date = '2017-01-01'
37 var end_date = '2017-08-01'
38
39
40 // Olha's idea to create mean images
41 var step = 15 // in days
42
```

```

31  var days = ee.List.sequence(0,
32    ee.Date(end_date).difference(ee.Date(start_date), 'day'),
33    step).
34    map(function(d) { return ee.Date(start_date).advance(d, "day")
35      })
36
37  var dates = days.slice(0,-1).zip(days.slice(1))
38
39  var s1coh_res = dates.map(function(range) {
40    var dstamp = ee.Date(ee.List(range).get(0)).format('YYYYMMdd')
41    var temp_collection = s1coh.filterDate(ee.List(range).get(0),
42      ee.List(range).get(1)).max()
43    return temp_collection.select(temp_collection.bandNames(),
44      [ee.String('COHVV_').cat(dstamp),
45       ee.String('COHVH_').cat(dstamp)])
46  })
47
48  function stack(i1, i2)
49  {
50    return ee.Image(i1).addBands(ee.Image(i2))
51  }
52
53  var s1stack = s1coh_res.slice(1).iterate(stack, s1coh_res.get(0))
54
55  Map.addLayer(ee.Image(s1stack).select([26,12,0]).clip(aoi), {max:
56    0.8}, "coherence")
57
58  Export.table.toDrive(ee.Image(s1stack).reduceRegions({collection:
59    brp2017, reducer: ee.Reducer.mean(), scale: 10}).
60    select(ee.List(['id', 'area', 'bufferedarea', 'perimeter',
61      'gws_gewasc',
62      'gws_gewas']).cat(ee.Image(s1stack).bandNames()), null,
63      false), "Landsense_GH_COH_Dump_Region1")

```

5 TensorFlow classification to process Sentinel-1

```

1  # PACKAGE
2  import os, sys
3
4  # DATA
5  # directory of the data
6  datafile='./data/NL_Veluwe_2017_S1_bsc' # Sentinel-1
7    backscattering

```

```

7  #datafile='./data/NL_Veluwe_2017_S1_coh' # Sentinel-1 coherence
8
9
10 # input parameters
11 rootname = datafile+'_cropselect'
12 nclass = 5
13 nepoch = 80
14 nrun = 2
15
16
17 # # 1 / Aggregate the crop by categories
18 # Remove unuseful columns, aggragate the classes, set a
19 # numerical label and save results with '_cropselect.csv' at
20 # the end
21
22 import sys
23 import pandas as pd
24
25 #load the data
26 df = pd.read_csv(datafile+ '.csv')
27
28 #remo small parcels
29 df = df[df.area > 1000]
30
31 # remove empty lines
32 df= df[df.iloc[:,3] > 0]
33
34 # remove unused columns
35 df.drop(['.geo', 'area', 'gws_gewas','bufferedarea', 'id',
36         'perimeter'], axis=1, inplace=True)
37 df["gws_gewasc"] = df.gws_gewasc.astype(int)
38
39 # aggregate the crops by classes
40 gra = df['gws_gewasc'].isin([265, 266, 331, 336, 383, 332])
41 mai = df['gws_gewasc'].isin([259, 316, 317])
42 cer = df['gws_gewasc'].isin([234, 236, 235, 237])
43 pot = df['gws_gewasc'].isin([2016, 2015 , 2017, 2014 ]) #2014,
44         # 2015, 2016,
45
46 # convert the class to label integer strating from 0,1,2...
47 df['gws_gewasc'] = 4
48 df.loc[gra, 'gws_gewasc'] = 0
49 df.loc[mai, 'gws_gewasc'] = 1

```

```

49 df.loc[cer, 'gws_gewasc'] = 2
50 df.loc[pot, 'gws_gewasc'] = 3
51
52 # remove unlabelled parcels
53 subset = df.loc[df['gws_gewasc'].isin([0,1,2,3,4])]

54
55 # convert 'gws_gewasc' to 'label' and remove 'gws_gewasc'
56 subset.insert(1, 'label', subset['gws_gewasc'])
57 subset.drop(['gws_gewasc'],axis=1, inplace=True)

58
59 # save the output as a .csv file
60 subset.to_csv(datafile + '_cropselect.csv')

61
62
63 # # 2. / Select the training and test data
64 # Select the training data by sampling in the different classes
65 # and save as outputs 1 file for training and 1 file for
66 # testing.
67
68 import sys
69 import pandas as pd
70 import numpy as np
71
72 samplesizeGRA=300
73 samplesizeMAI=300
74 samplesizeCER=150
75
76 df = pd.read_csv(datafile + '_cropselect.csv', low_memory=False)
77
78
79 for i_nrun in range(nrun):
80     trainingGRA=df.loc[df['label'] ==
81         ↳ 0].take(np.random.permutation(len(df.loc[df['label'] ==
82         ↳ 0]))[:samplesizeGRA])
83     trainingMAI=df.loc[df['label'] ==
84         ↳ 1].take(np.random.permutation(len(df.loc[df['label'] ==
85         ↳ 1]))[:samplesizeMAI])
86     trainingCER=df.loc[df['label'] ==
87         ↳ 2].take(np.random.permutation(len(df.loc[df['label'] ==
88         ↳ 2]))[:samplesizeCER])
89     training=trainingGRA.append(trainingMAI).append(trainingCER)
90     testing = df.drop(training.index)
91     # save outputs

```

```

86         → training.to_csv(datafile+'_cropselect_train_{}'.format(len(training))+'_nrun{}'.format(nrun))
87         → testing.to_csv(datafile+'_cropselect_test_{}'.format(len(testing))+'_nrun{}'.format(nrun))
88
89
90 # # 3 / Learning : building the neural network model, training
91     → the model, applying it to the parcel
92 #
93 # Loading the training data
94
95
96 import numpy as np
97 import tensorflow as tf
98 import tflearn
99 import sys
100 import glob
101
102 # Load CSV file, indicate that the first column represents labels
103 from tflearn.data_utils import load_csv
104
105 # tflearn.init_graph(gpu_memory_fraction=0.0)
106
107 for i_nrun in range(nrun):
108     tf.reset_default_graph() #reset before starting
109     # load training
110     flist = glob.glob(rootname +
111                         '_train_*_nrun{}.csv'.format(i_nrun))
112     if len(flist) > 1:
113         print("FATAL: Only single training set allowed for {},"
114               " found {}"
115               ".format(rootname, len(flist)))"
116               sys.exit(1)
117     elif len(flist) == 0:
118         print("FATAL: No training set found for"
119               "{}".format(rootname))
120         sys.exit(1)
121
122     # load testing
123     glist = glob.glob(rootname +
124                         '_test_*_nrun{}.csv'.format(i_nrun))
125     if len(glist) > 1:
126         print("FATAL: Only single test set allowed for {}, found"
127               "{}"
128               ".format(rootname, len(flist)))"

```

```

124         sys.exit(1)
125     elif len(glist) == 0:
126         print("FATAL: No test set found for {}".format(rootname))
127         sys.exit(1)
128
129     fname = plist[0]
130     gname = glist[0]
131
132     data, labels = load_csv(fname, target_column=3,
133                             categorical_labels=True,
134                             ↪ n_classes=nclass)
135
136     test_data, test_labels = load_csv(gname, target_column=3,
137                                         categorical_labels=True,
138                                         ↪ n_classes=nclass)
139
140     # Preprocessing function
141     def preprocess(profiles, columns_to_delete):
142         # Sort by descending id and delete columns
143         for column_to_delete in sorted(columns_to_delete,
144                                         ↪ reverse=True):
145             [profile.pop(column_to_delete) for profile in
146              ↪ profiles]
147
148         return np.array(profiles, dtype=np.float32)
149
150     # Ignore 'id'
151     to_ignore=[0,1,2]
152
153     # Preprocess data
154     data = preprocess(data, to_ignore)
155
156     # Build neural network
157     net = tflearn.input_data(shape=[None, len(data[0])])
158     net = tflearn.fully_connected(net, 32)
159     net = tflearn.fully_connected(net, 32)
160     net = tflearn.fully_connected(net, nclass,
161                                 ↪ activation='softmax')
162     net = tflearn.regression(net)
163
164     # Define model
165     model = tflearn.DNN(net)
166     # Start training (apply gradient descent algorithm)
167     model.fit(data, labels, n_epoch=nepoch, batch_size=32,
168               ↪ show_metric=True)

```

```

164     fw = open('{}_{}_predictions.csv'.format(rootname, i_nrun),
165             'w')
166     fw.write("id,klass")
167     for i in range(nclass):
168         fw.write(",prob{}".format(i))
169     fw.write('\n')
170
171     # Check predictions for the samples not used in training
172     for i in range(len(test_data)):
173         sample = test_data[i][3:]
174         slabel = test_labels[i].tolist().index(1)
175         #print(labels[i])
176         pred = model.predict([sample])
177         fw.write("{} ,{} ".format(test_data[i][2], str(slabel)))
178         for i in range(nclass):
179             fw.write(",{:6.2f} ".format(100*pred[0][i]))
180         fw.write('\n')
181
182     tf.reset_default_graph()
183
184
185
186     # # 4/ Select the class with the highest probability for each
187     #     parcel and each run
188
189     for i_nrun in range(nrun):
190         # load the predictions probability
191         df = pd.read_csv(glob.glob(rootname +
192             '{}_predictions.csv'
193             .format(i_nrun))[0], index_col=0, low_memory = False)
194         # index
195         r_index = df.columns[1:]
196         # Select the class with the probability maximum as the
197         #     prediction
198         df['pred'] = df.apply(lambda x:
199             np.array(x[r_index]).argmax(), axis=1)
200         # Get the maximum probability
201         df['pred_max'] = df.apply(lambda x:
202             np.array(x[r_index]).max(), axis=1)
203
204         # When the maximum probability is below 70 %, do not change
205         #     the class prediction
206         df_ok = df[df.pred_max > 70]
207         df_nok = df[df.pred_max <= 70]

```

```

204     df_nok['pred']=df_nok['klass']
205     df=df_ok.append(df_nok)
206
207     df.drop(r_index, axis=1, inplace=True)
208     df.drop('pred_max', axis=1, inplace=True)
209
210     # save output
211     df.to_csv(rootname+ '_{}_{class}.csv'.format(i_nrun))
212
213
214 # # 5 / Combine the different run and retrieve the majority class
215
216
217 import pandas as pd
218 import numpy as np
219 import sys
220
221 from collections import Counter
222
223
224
225 df0 = pd.read_csv('{}_0_{class}.csv'.format(rootname), index_col =
226   ↵ 0, low_memory=False)
226 df1 = pd.read_csv('{}_1_{class}.csv'.format(rootname), index_col =
227   ↵ 0, low_memory=False)
228
229
230 # Create the join and retain 'klass' label as 'klass_1'
231 df = df0.join(df1, how="outer", rsuffix= '_1')
232 # Records that were not yet in df0 have 'klass' label missing
233   ↵ (NA)
234 # so, overwrite with those of 'klass_1'
235 df['klass'].loc[df['klass'].isnull()] =
236   ↵ df['klass_1'].loc[df['klass'].isnull()]
235 # and drop the now redundant 'klass_1' label
236 df.drop('klass_1', axis=1, inplace=True)
237
238
239
240 df.fillna(-1, inplace=True)
241 r_index = df.columns[1:]
242
243
244

```

```
245 df['majclass'] = df.apply(lambda x:  
246     ↪ Counter(x[r_index]).most_common(1)[0][0], axis=1)  
246 df['majcount'] = df.apply(lambda x:  
247     ↪ Counter(x[r_index]).most_common(1)[0][1], axis=1)  
247  
248 df.astype(int).to_csv('{}_classes.csv'.format(rootname))
```