Google Earth Engine

Hands on Experience

Project 3 Deforestation

Introduction

Deforestation

Deforestation is the process of clearing or removing large areas of forests or trees, typically for the purpose of converting the land for agricultural, industrial, or urban use. This often involves the complete removal of trees and vegetation, resulting in the transformation of forested areas into non-forested ones. Deforestation has resulted in habitat damage, biodiversity loss, and aridity. Deforestation also causes extinction, changes to climatic conditions, desertification, and displacement of populations.

Between 15 million to 18 million hectares of forest, an area the size of Bangladesh, are destroyed every year. On average 2,400 trees are cut down each minute.

Using Google Earth Engine, this project focuses on utilising Synthetic Aperture Radar (SAR) images from ALOS-2, optical data from Sentinel-2 and machine learning to identify deforestation and assess the impacts.

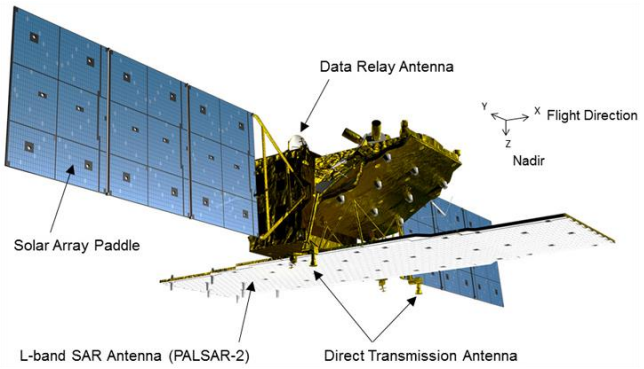
Aerial view showing a deforested area of the Amazonia rainforest in Lábrea, Brazil on September 15, 2021.

ALOS-2

Identifying deforestation is done by comparing images from a time-period (usually 1-2 years duration) to a later time-period. Synthetic Aperture Radar (SAR) is highly sensitive to forest cover and vegetation's as compared to optical datasets. SAR imagery is also not affected by clouds and other weather phenomenon thus, it provides a reliable way to identify deforested areas. In this project, we will be using ALOS-2 SAR imagery.

ALOS-2 Synthetic Aperture Radar

- Advanced Land Observing Satellite-2 (ALOS-2), also called Daichi-2, is a Japanese satellite launched in 2014
- L-band Synthetic Aperture Radar (SAR) sensor (PALSAR-2)
- Spatial Resolution: 25m
- Temporal Resolution: 14 days



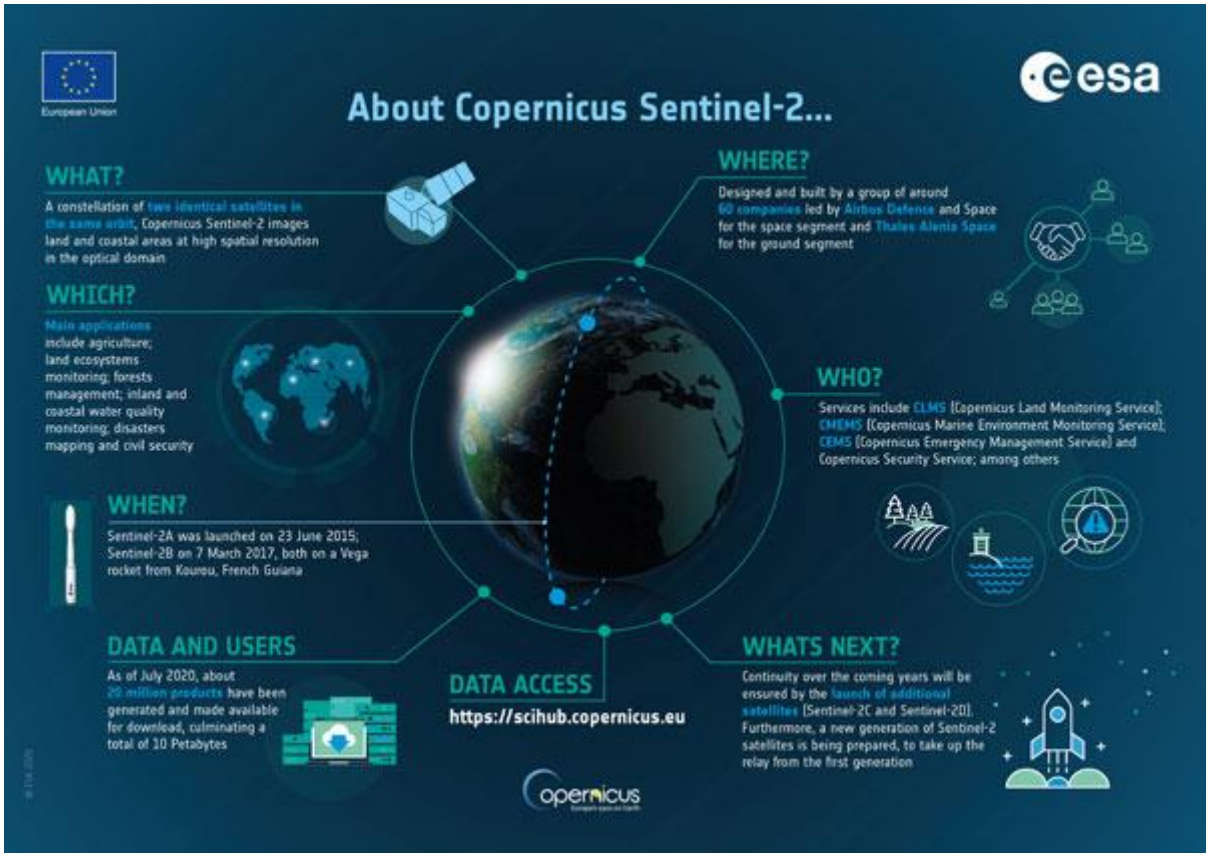
Application	Disaster, Land, Agriculture, Natural Resources, Sea Ice & Maritime Safety
L-band SAR (PALSAR-2)	Stripmap: 3 to 10m res., 50 to 70 km swath ScanSAR: 100m res., 350km/490km swath Spotlight: 1×3m res., 25km swath
Orbit	Sun-synchronous orbit Altitude: 628km Local sun time : 12:00 +/- 15min Revisit: 14days Orbit control: ≦+/-500m
Lifetime	5 years (target: 7 years)
Launch	May 24, 2014; H-IIA launch vehicle
Downlink	X-band: 800Mbps(16QAM) 400/200Mbps(QPSK) Ka-band: 278Mbps (Data Relay)

SENTINEL-2

Visual interpretation of forests using only SAR could be challenging so we also make use Sentinel-2 optical imagery to aid us in identifying forested areas. Sentinel-2 imagery used in this project is only to supplement forest identification.

Sentinel-2

- Sentinel-2 is an Earth observation mission with two satellites, Sentinel-2A and Sentinel-2B
- Multi-spectral instrument (MSI) with 13 spectral channels
- Spatial Resolution: 10m
- Temporal Resolution: 10 days



Guided Lab

1. Defining ROI and time frame

```
//Define ROI using corner coordinates
var RoIShp = ee.Geometry.Polygon(
  [[[-67.3540656,-8.7206317],
   [-67.351319,-9.2509964],
   [-66.6440742,-9.2523519],
   [-66.6495673,-8.7111297],
   [-67.3540656,-8.7206317]]]
);

//Show the layer and center it on the map
Map.addLayer(RoIShp,{color:'Red'},'RoIShp');
Map.centerObject(RoIShp,12);
//Import the image collection
var AL2ImgCol= ee.ImageCollection('JAXA/ALOS/PALSAR-
2/Level2_2/ScanSAR')
                  //Filter the scenes overlapping with the
ROI
                  .filterBounds(RoIShp)
                  //Filter the scenes observed from 2020 to
end of 2022
                  .filterDate('2020-01-01','2022-12-31');

//Print the number of available scenes within ROI and
timeframe
print('Number of available scenes over
ROI',AL2ImgCol.size());
//Print the list and feature properties of all the
available scenes
print('List of scenes over RoI in 2020-9', AL2ImgCol);
```

Guided Lab

2. Pre-processing functions

```
//Remove images that do not have all required bands
var Cleaning_filter  = function(img){
  var size = (img.bandNames()).size();
    return ee.Algorithms.If(size.eq(4), img.set('A1_Stat', 'Keep'),
img.set('A1_Stat', 'Drop'));
};

//Convert digital numbers to decibels
var DN2dB= function (img) {
  img = ee.Image(img);
  var imgdB = ((img.pow(2)).log10()).multiply(10.0).subtract(83.0);
  return imgdB};

//Create a ratio function of HH/VV
var Ratio= function(img){
  var ratio=((img.select('HH')).divide(img.select('HV'))).rename('Ratio');
  return img.addBands(ratio);
};

//Noise reduction and smoothing function
var Smoothing=function (img){
  return img.focalMode({radius:1, kernelType:'square', units:'pixels',
iterations:1});
};

//Remove images with no data
var nodata_filter  = function(img1){
  var img2 = ee.Image(img1);
  var img_nodata  = img2.select('HH').neq(0);
  var img3 = img2.updateMask(img_nodata);
  return img3;
};
```

Guided Lab

3. Get and show SAR Images

```
//Apply the filters and get the SAR images from 2020 to 2021
var Desc_Clean_AL2ImgCol_BEFORE= (AL2ImgCol.filterDate('2020-01-01','2020-12-31')

.map(Cleaning_filter)).filter(ee.Filter.eq('A1_Stat','Keep'))
                                        //choose a particular pass direction
                                        .filter(ee.Filter.eq('PassDirection','Descending'))
                                        .map(nodata_filter)
                                        .map(DN2dB)
                                        .map(Ratio)
                                        .map(Smoothing)
                                        .median()
                                        .clip(RoIShp)
                                        .select(['HH','HV','Ratio']);
//Show SAR images on layer with color palette
Map.addLayer(Desc_Clean_AL2ImgCol_BEFORE, {bands:['HH','HV','Ratio'],min:-25, max: 3},
'Desc_Clean_AL2ImgCol_BEFORE');

//Apply the filters and get the SAR images from 2022 to 2023
var Desc_Clean_AL2ImgCol_AFTER= (AL2ImgCol.filterDate('2022-01-01','2022-12-31')

.map(Cleaning_filter)).filter(ee.Filter.eq('A1_Stat','Keep'))
                                        .filter(ee.Filter.eq('PassDirection','Descending'))
                                        .map(nodata_filter)
                                        .map(DN2dB)
                                        .map(Ratio)
                                        .map(Smoothing)
                                        .median()
                                        .clip(RoIShp)
                                        .select(['HH','HV','Ratio']);
Map.addLayer(Desc_Clean_AL2ImgCol_AFTER, {bands:['HH','HV','Ratio'],min:-25, max: 3},
'Desc_Clean_AL2ImgCol_AFTER');
```

Guided Lab

4. Get and show Optical Images

```
//Cloud masking function
function maskS2clouds(image) {
  var qa = image.select('QA60');
  var cloudBitMask = 1 << 10;
  var cirrusBitMask = 1 << 11;
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
      .and(qa.bitwiseAnd(cirrusBitMask).eq(0));
  return image.updateMask(mask).divide(10000);
}

//Apply the filters and get the Optical images from 2020 to 2021
var S2_BEFORE= ee.ImageCollection('COPERNICUS/S2_HARMONIZED')
                          .filterDate('2020-01-01','2020-12-31')
                          .filterBounds(RoIShp)
                          //Only choose images with <20% clouds
                          .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
                          .map(maskS2clouds)
                          .median()
                          .clip(RoIShp);

//Apply the filters and get the Optical images from 2022 to 2023
var S2_AFTER= ee.ImageCollection('COPERNICUS/S2_HARMONIZED')
                          .filterDate('2022-01-01','2022-12-31')
                          .filterBounds(RoIShp)
                          .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
                          .map(maskS2clouds)
                          .median()
                          .clip(RoIShp);
Map.addLayer(S2_BEFORE, {min: 0.0,max: 0.3, bands: ['B4', 'B3', 'B2']},
'S2_BEFORE');
Map.addLayer(S2_AFTER, {min: 0.0,max: 0.3, bands: ['B4', 'B3', 'B2']},
'S2_AFTER');
```

Guided Lab

5. Machine learning

```
//Create your own data set for training and testing
var Points= Class2.merge(Class1);
var training = Desc_Clean_AL2ImgCol_AFTER.sampleRegions({
  collection: Points,
  properties: ['Class'],
  scale: 25
});

//Pre-determined dataset is also available for import at P3_Deforestation.txt

//Separate dataset into training and testing
var split = 0.7;  // Roughly 70% training, 30% testing.
var withRandom = training.randomColumn("Random",0);
    var trainingPart = withRandom.filter(ee.Filter.lt("Random", split));
    var testingPart  = withRandom.filter(ee.Filter.gte("Random", split));

//Training the classifier
var Bands=Desc_Clean_AL2ImgCol_BEFORE.bandNames();
var classifier = ee.Classifier.smileRandomForest(40).train({
  features: trainingPart,
  classProperty : 'Class',
  inputProperties: Bands
});

//Show the trained forest/non forest data from 2020 to 2021
var RFclassified_BEFORE =
(Desc_Clean_AL2ImgCol_BEFORE.select(Bands)).classify(classifier);
Map.addLayer(RFclassified_BEFORE, {min: 1, max:2, opacity:1,
palette:["10ff0c","ff0656"]}, 'RFclassified_BEFORE', 1);

//Show the trained forest/non forest data from 2022 to 2023
var RFclassified_AFTER =
(Desc_Clean_AL2ImgCol_AFTER.select(Bands)).classify(classifier);
Map.addLayer(RFclassified_AFTER, {min: 1, max:2, opacity:1,
palette:["10ff0c","ff0656"]},'RFclassified_AFTER', 1);

//Accuracy assessement
var Testing= testingPart.classify(classifier);
    var confusionMatrix = Testing.errorMatrix("Class", 'classification');
    var OverallAccuracy= confusionMatrix.accuracy();
    var PA=confusionMatrix.producersAccuracy();
    var user=confusionMatrix.consumersAccuracy();
    print('User Accuracy',user);
    print('confusionMatrix', confusionMatrix);
    print('producers Accuracy',PA);
    print('Overall Accuracy', OverallAccuracy);
```

Guided Lab

6. Change detection

```
//Define a transition matrix to show deforested pixels
var TransitionMatrix=
(RFclassified_BEFORE.multiply(100).add(RFclassified_AFTER)).remap([102],[1]);
Map.addLayer(TransitionMatrix,{palette:['Red']}, 'Detected Changes', 1);

//Count the number of pixels in ROI
var countimage = RFclassified_BEFORE.select('classification').reduceRegion({
  reducer: ee.Reducer.count(),
  geometry: RoIShp,
  scale: 25,
});

//Get and print the number of pixels in ROI
var getcountimage = countimage.get('classification');
print('Number of pixels of ROI Image: ', getcountimage);

//Count the number of pixels in deforested areas
var countmatrix = TransitionMatrix.select('remapped').reduceRegion({
  reducer: ee.Reducer.count(),
  geometry: RoIShp,
  scale: 25,
});

//Get and print the number of pixels in deforested areas
var getcountmatrix = countmatrix.get('remapped');
print('Number of pixels of Deforested area: ', getcountmatrix);

//Get and print the total deforested area
var squarearea = getcountmatrix.getInfo()*0.000625;
print('Total Deforested area in square kilometres: ', squarearea);

//Get and print the percentage forest lost
var percentloss = ((getcountmatrix.getInfo()/getcountimage.getInfo())*100);
print('Percentage loss of forest: ', percentloss);
```

Guided Lab

7. Comparison with a pre-defined default dataset

```
var WorldBoundary = ee.FeatureCollection('FAO/GAUL/2015/level0');
var RoI = WorldBoundary.filter(ee.Filter.eq('ADM0_CODE',196));
var DefaultTD= ee.FeatureCollection('projects/ee-
alos2geeapp/assets/PhilSA/FNF_Samples_2020_3Countries')
                                    .filterBounds(RoI)
                                    .filter(ee.Filter.eq('Orbit','Descending'));

Map.addLayer(DefaultTD.filter(ee.Filter.eq('Class',1)),{color:'Green'},'Forest');
Map.addLayer(DefaultTD.filter(ee.Filter.eq('Class',2)),{color:'Red'},'Non-Forest');
print('Size of TD for class 1:',DefaultTD.filter(ee.Filter.eq('Class',1)).size());
print('Size of TD for class 2:',DefaultTD.filter(ee.Filter.eq('Class',2)).size());

var withRandomnew = DefaultTD.randomColumn("Random",0);
    var trainingPartnew = withRandomnew.filter(ee.Filter.lt("Random", split));
    var testingPartnew  = withRandomnew.filter(ee.Filter.gte("Random", split));

var classifiernew = ee.Classifier.smileRandomForest(10).train({
  features: trainingPartnew,
  classProperty : 'Class',
  inputProperties: Bands
});

var RFclassified_BEFOREnew =
(Desc_Clean_AL2ImgCol_BEFORE.select(Bands)).classify(classifiernew);
Map.addLayer(RFclassified_BEFOREnew, {min: 1, max:2, opacity:1, palette:["10ff0c","ff0656"]},
'RFclassified_BEFOREnew', 1);

var RFclassified_AFTERnew = (Desc_Clean_AL2ImgCol_AFTER.select(Bands)).classify(classifiernew);
Map.addLayer(RFclassified_AFTERnew, {min: 1, max:2, opacity:1,
palette:["10ff0c","ff0656"]},'RFclassified_AFTERnew', 1);

var Testing1= testingPartnew.classify(classifiernew);
    var confusionMatrix1 = Testing1.errorMatrix("Class", 'classification');
    var OverallAccuracy1= confusionMatrix1.accuracy();
    var PA1=confusionMatrix1.producersAccuracy();
    var user1=confusionMatrix1.consumersAccuracy();
    print('User Accuracy1',user1);
    print('confusionMatrix1', confusionMatrix1);
    print('producers Accuracy1',PA1);
    print('Overall Accuracy1', OverallAccuracy1);
```