VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science Engineering and Information Systems

TECHNICAL ANSWERS FOR REAL WORLD PROBLEMS

PROJECT REVIEW

TITLE OF THE PROJECT: Deforestation Detection Using TensorFlow

TEAM MEMBERS: 1) MOHAN RAJAN A – 20MIS0059

2) GOKUL R – 20MIS0332

3) DINESH RAJAN - 20MIS0449

SLOT: TCC1

COURSE CODE: SWE1901

FACULTY: DAPHNE LOPEZ

Abstract

Various technologies, such as remote sensing, machine learning, and geographic information systems (GIS), have emerged as powerful tools in deforestation detection and monitoring. Satellite-based remote sensing provides frequent and wide-ranging coverage, enabling real-time monitoring of forest cover changes. Machine learning algorithms, have demonstrated remarkable capabilities in analysing large datasets and identifying deforestation patterns automatically. Furthermore, geospatial technologies like GIS facilitate data integration, visualization, and analysis, aiding in the identification of deforestation hotspots and trends. The integration of these technologies allows for the creation of comprehensive deforestation monitoring systems that combine satellite imagery, ground-based data, and predictive models. We will be using TensorFlow as base and to build a EfficientNetV2 on top of to classify the satellite images. Then we will be comparing the forest images alone with time series of satellite images and with its density to check whether that area is afforested or deforested.

1. Introduction

Detecting and monitoring changes in land cover, particularly in forested regions, is of critical importance for environmental conservation and sustainable land management. The use of satellite imagery and deep learning techniques has revolutionized our ability to analyze these changes with high accuracy and efficiency. This project leverages TensorFlow as the foundation and introduces EfficientNetV2, a powerful convolutional neural network architecture, to classify satellite images, with a specific focus on identifying afforestation and deforestation events.

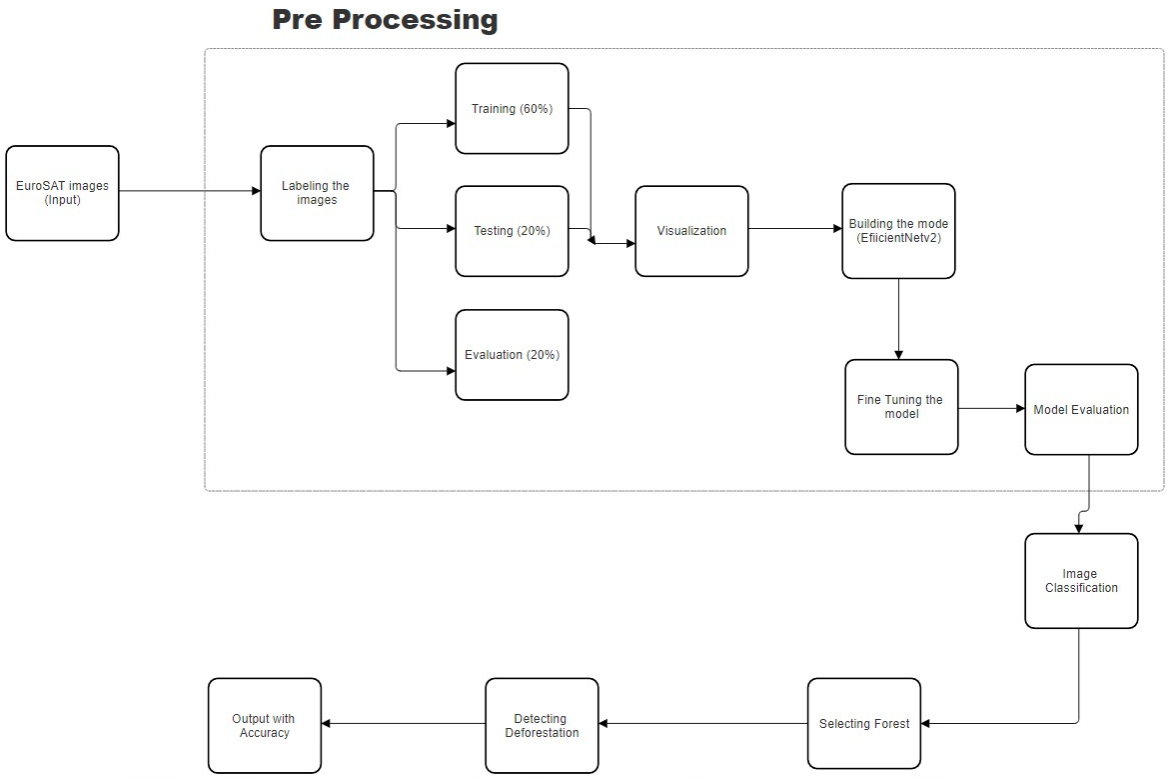
Project Objectives:

1. **EfficientNetV2 Classification:** The project centers around the implementation of the EfficientNetV2 architecture on TensorFlow. EfficientNetV2 is known for its state-of-the-art performance in image classification tasks. It will be fine-tuned to classify satellite images, specifically those showing changes in land cover, such as forested areas.
2. **Change Detection:** The primary goal is to detect land cover changes over time. By analysing time series of satellite images, the project aims to identify afforestation and deforestation events. This is achieved by

comparing images from different time periods and leveraging the deep learning model to highlight significant alterations in land cover.

3. **Density Analysis:** In addition to change detection, the project investigates the concept of density analysis. It assesses the degree of forestation or deforestation in specific regions over time, providing a quantitative understanding of land cover changes.

2. Methodology



TensorFlow

TensorFlow Keras is a popular deep learning framework that provides a high-level API for building and training deep learning models. A Keras layer in TensorFlow is a reusable building block of neural networks. It is a class that defines the functionality of a single step in a neural network. Layers can be stacked together to create more complex models.

In deforestation detection, TensorFlow Keras layers can be used to build a deep learning model that can classify satellite images as forested or deforested. One approach to using TensorFlow Keras layers for deforestation detection is to use a convolutional neural network (CNN) architecture. A CNN is a type of deep

learning model that is well-suited for image classification tasks. The CNN architecture consists of several layers, including convolutional layers, pooling layers, and fully connected layers.

EfficientNetV2

- EfficientNetV2 is a family of convolutional neural network (CNN) models that are designed to be both computationally efficient and accurate. These models were introduced in a research paper titled "EfficientNetV2: Smaller Models and Faster Training" 1.

- EfficientNetV2 models are built using a scaling method that optimizes the depth, width, and resolution of the CNN architecture. This scaling method allows EfficientNetV2 models to achieve state-of-the-art performance on image classification tasks while using fewer parameters and less computation than other CNN architectures 1.

- In your case, you can use an EfficientNetV2 model to classify satellite images as forested or deforested. You can use the TensorFlow Keras implementation of EfficientNetV2 to build your model. The Keras implementation provides several pre-trained models that you can use for transfer learning or fine-tuning on your dataset 1.

Reason to use EfficientNetV2 instead of V1:

Problems with EfficientNet (Version 1)

The EfficientNet (original version) has the following bottlenecks —

a. EfficientNets are generally faster to train than other large CNN models. But, when large image resolution was used to train the models (B6 or B7 models), the training was slow. This is because larger EfficientNet models require larger image sizes to get optimal results, and when larger images are used, the batch size needs to be lowered to fit these images in the GPU/TPU memory, making the overall process slow.

b. In the early layers of the network architecture, depthwise convolutional layers (MBConv) were slow. Depthwise convolutional layers generally have fewer parameters than regular convolutional layers, but the problem is that they cannot fully make use of modern accelerators. To overcome this problem EfficientNetV2

uses a combination of MBConv and Fused MBConv to make the training faster without increasing parameters (discussed later in the article).
c. Equal scaling was applied to the height, width, and image resolution to create the various EfficientNet models from B0 to B7. This equal scaling of all layers is not optimal. For example, if the depth is scaled by 2, all the blocks in the network get scaled up 2 times, making the network very large/deep. It might be more optimal to scale one block two times and the other 1.5 times (non-uniform scaling), to reduce the model size while maintaining good accuracy.

3. Experimental Setup

Dataset: Eurosat Dataset

We will be using the EuroSAT dataset based on Sentinel-2 satellite images covering 13 spectral bands. It consists of 27,000 labeled samples of 10 different classes: annual and permanent crop, forest, herbaceous vegetation, highway, industrial, pasture, residential, river, and sea lake.

EuroSAT dataset comes in two varieties:

- rgb (default) with RGB that contain only the R, G, B frequency bands encoded as JPEG images.
- all: contains all 13 bands in the original value range.

Dataset: Hansen Dataset

The Hansen Global Forest Change dataset is **a time-series analysis of Landsat images that shows global forest extent and change**. The dataset is based on remote sensing and uses a global definition of "forest".
Details

- The dataset quantifies global annual forest loss between 2000 and 2012.
- The dataset has a spatial resolution of 30 meters.
- The dataset is consistent with IPCC principles.
- The dataset is no-cost, transparent, and globally available.
- The dataset is divided into 10x10 degree tiles.

Definitions

- Tree cover: All vegetation greater than 5 meters in height.
- Tree cover loss: The complete removal of tree cover canopy at the Landsat pixel scale.

- **Google Colab**
    - Google Colab is a cloud-based service that allows you to write and run code in a Jupyter Notebook environment. Jupyter Notebooks are a popular tool for data scientists and developers, as they allow for an interactive coding experience.
    - With Google Colab, you can write and execute Python code in your browser, with zero configuration required. You can also access GPUs free of charge, which can be useful for training deep learning models.
    - Google Colab is integrated with Google Drive, which allows you to share, comment, and collaborate on the same document with multiple people [3]. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them.

- **Google Earth Engine**

    - Google Earth Engine is a cloud-based geospatial analysis platform that enables users to visualize and analyze satellite images of our planet. Scientists, researchers, and developers use Earth Engine to detect changes, map trends, and quantify differences on the Earth's surface.
    - Earth Engine combines a multi-petabyte catalog of satellite imagery and geospatial datasets with planetary-scale analysis capabilities. It provides several pre-built algorithms for image processing, such as image classification, object detection, and change detection.
    - The Earth Engine API is available in Python and JavaScript, making it easy to harness the power of Google's cloud for your own geospatial analysis.
    - Google Earth Engine is available for commercial use and remains free for academic and research use

- **Google Earth Pro**

Google Earth Pro is a desktop application that allows users to explore the world in 3D. It provides a wealth of geographic information, including satellite imagery, terrain data, and 3D buildings. Google Earth Pro is designed for professional use and includes advanced features such as the ability to import and export GIS data, measure distances and areas, and create custom maps. It also allows users to go back in time with historical imagery and explore the world's oceans with underwater imagery. Google Earth Pro is available for Windows, Mac, and Linux operating systems. It can be downloaded for free from the Google Earth website. Google Earth Pro is a powerful tool for geospatial analysis and visualization that can be used by professionals in a variety of fields, including urban planning, environmental science, and archaeology.

4. Expected Result

Density Maps: You will likely generate density maps that provide spatial information about the distribution and density of forested regions in the area of interest. These maps can be visualized and analyzed to identify areas with high or low forest density.

Change Detection: Density analysis can complement change detection by quantifying the extent of land cover changes, allowing you to distinguish between minor and major changes in forest density.

Validation and Accuracy: It's important to assess the accuracy and reliability of your results. Ground-truth data or validation measures can help confirm the accuracy of your density analysis.

Earth Engine Code:

```
//Hansen dataset visualization

var gfc2014 = ee.Image('UMD/hansen/global_forest_change_2015');
Map.addLayer(gfc2014);

//forest cover in the year 2000
Map.addLayer(gfc2014, {bands: ['treecover2000']}, 'treecover2000');
```

```
//3 bands of year 2015 for healthu vegetation
Map.addLayer(
    gfc2014, {bands: ['last_b50', 'last_b40', 'last_b30']}, 'false color');

//in year 2000, forest extent - greeen
// forest loss - red
//forest gain - blue
Map.addLayer(gfc2014, {bands: ['loss', 'treecover2000', 'gain']}, 'green');

//brightening blue (forest gain) & red (forest loss)
Map.addLayer(gfc2014, {
  bands: ['loss', 'treecover2000', 'gain'],
  max: [1, 255, 1]
}, 'forest cover, loss, gain');

//pallete
Map.addLayer(gfc2014, {
  bands: ['treecover2000'],
  palette: ['000000', '00FF00']
}, 'forest cover palette');

//brighten up the image
Map.addLayer(gfc2014, {
  bands: ['treecover2000'],
  palette: ['000000', '00FF00'],
  max: 100
}, 'forest cover percent');

//masking
Map.addLayer(gfc2014.mask(gfc2014), {
  bands: ['treecover2000'],
  palette: ['000000', '00FF00'],
  max: 100
}, 'forest cover masked');

////////////////////////////////////////
//Quantifying forest change
// Load the data and select the bands of interest.
var gfc2014 = ee.Image('UMD/hansen/global_forest_change_2015');
var lossImage = gfc2014.select(['loss']);
var gainImage = gfc2014.select(['gain']);

// Use the and() method to create the lossAndGain image.
var gainAndLoss = gainImage.and(lossImage);

// Show the loss and gain image.
Map.addLayer(gainAndLoss.updateMask(gainAndLoss),
    {palette: 'FF00FF'}, 'Gain and Loss');
```

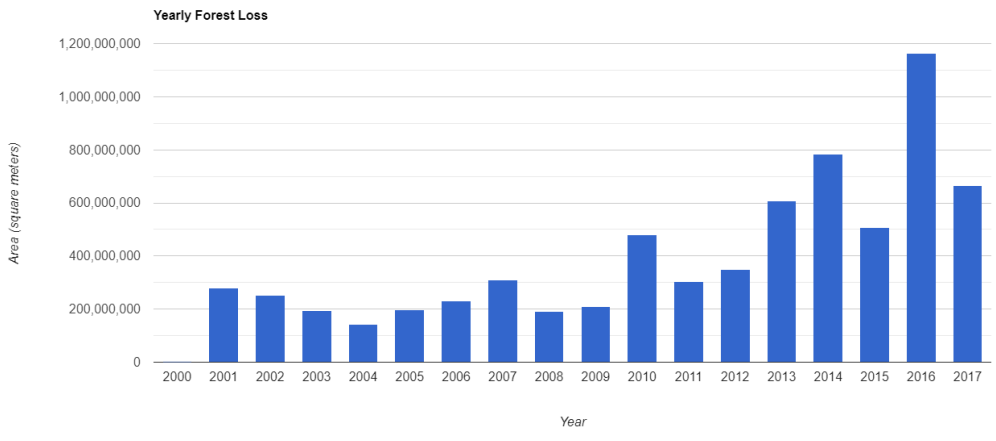
```
// Displaying forest, loss, gain, and pixels where both loss and gain occur.
var gfc2014 = ee.Image('UMD/hansen/global_forest_change_2015');
var lossImage = gfc2014.select(['loss']);
var gainImage = gfc2014.select(['gain']);
var treeCover = gfc2014.select(['treecover2000']);

// Use the and() method to create the lossAndGain image.
var gainAndLoss = gainImage.and(lossImage);

// Add the tree cover layer in green.
Map.addLayer(treeCover.updateMask(treeCover),
    {palette: ['000000', '00FF00'], max: 100}, 'Forest Cover');

// Add the loss layer in red.
Map.addLayer(lossImage.updateMask(lossImage),
    {palette: ['FF0000']}, 'Loss');

// Add the gain layer in blue.
Map.addLayer(gainImage.updateMask(gainImage),
    {palette: ['0000FF']}, 'Gain');

// Show the loss and gain image.
Map.addLayer(gainAndLoss.updateMask(gainAndLoss),
    {palette: 'FF00FF'}, 'Gain and Loss');

// Load country features from Large Scale International Boundary (LSIB)
dataset.
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');

// Subset the ind Republic feature from countries.
var ind = countries.filter(ee.Filter.eq('country_na', 'India'));

// Get the forest loss image.
var gfc2014 = ee.Image('UMD/hansen/global_forest_change_2015');
var lossImage = gfc2014.select(['loss']);

// Sum the values of forest loss pixels in the ind Republic.
var stats = lossImage.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: ind,
  scale: 30,
  maxPixels: 1e9
});
print(stats);

//loss
print('pixels representing loss: ', stats.get('loss'));
```

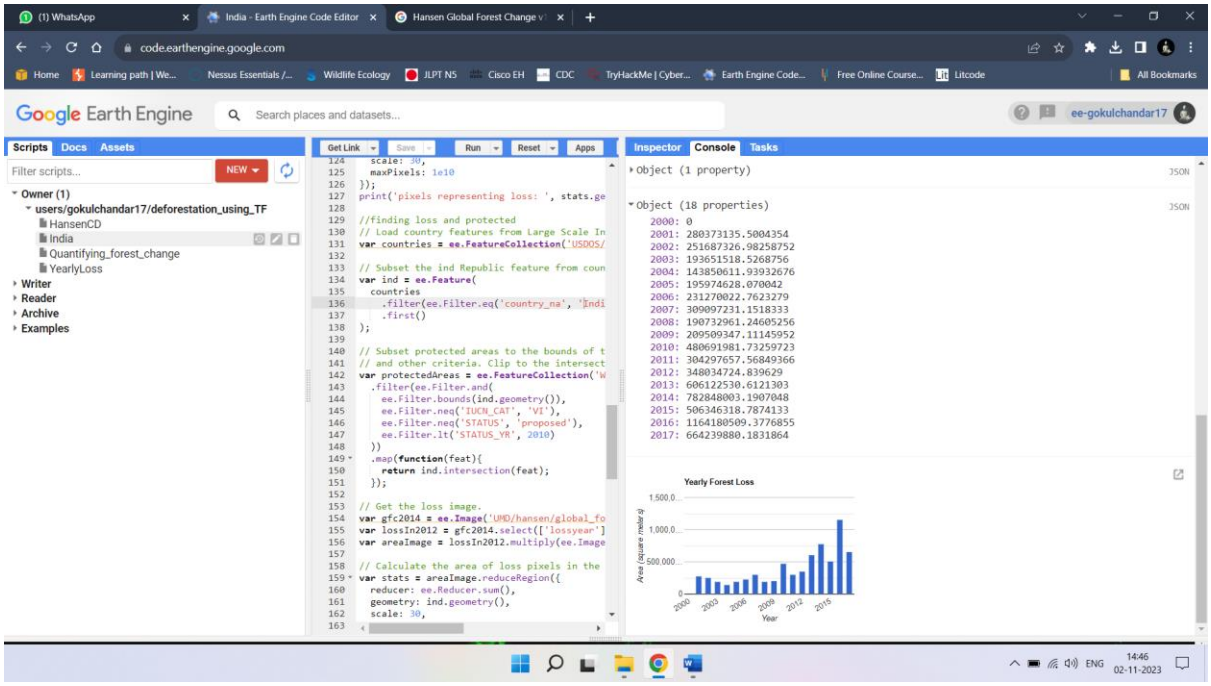
```
//calculating pixel areas

// Load country features from Large Scale International Boundary (LSIB)
dataset.
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');

// Subset the ind Republic feature from countries.
var ind = countries.filter(ee.Filter.eq('country_na', 'Rep of the ind'));

// Get the forest loss image.
var gfc2014 = ee.Image('UMD/hansen/global_forest_change_2015');
var lossImage = gfc2014.select(['loss']);
var areaImage = lossImage.multiply(ee.Image.pixelArea().divide(1e6));

// Sum the values of forest loss pixels in the ind Republic.
var stats = areaImage.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: ind,
  scale: 30,
  maxPixels: 1e10
});
print('pixels representing loss: ', stats.get('loss'), 'square meters');

//finding loss and protected
// Load country features from Large Scale International Boundary (LSIB)
dataset.
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');

// Subset the India  feature from countries.
var ind = ee.Feature(
  countries
    .filter(ee.Filter.eq('country_na', 'India'))
    .first()
);

// Subset protected areas to the bounds of the ind feature
// and other criteria. Clip to the intersection with ind.
var protectedAreas = ee.FeatureCollection('WCMC/WDPA/current/polygons')
  .filter(ee.Filter.and(
    ee.Filter.bounds(ind.geometry()),
    ee.Filter.neq('IUCN_CAT', 'VI'),
    ee.Filter.neq('STATUS', 'proposed'),
    ee.Filter.lt('STATUS_YR', 2010)
  ))
  .map(function(feat){
    return ind.intersection(feat);
  });
```

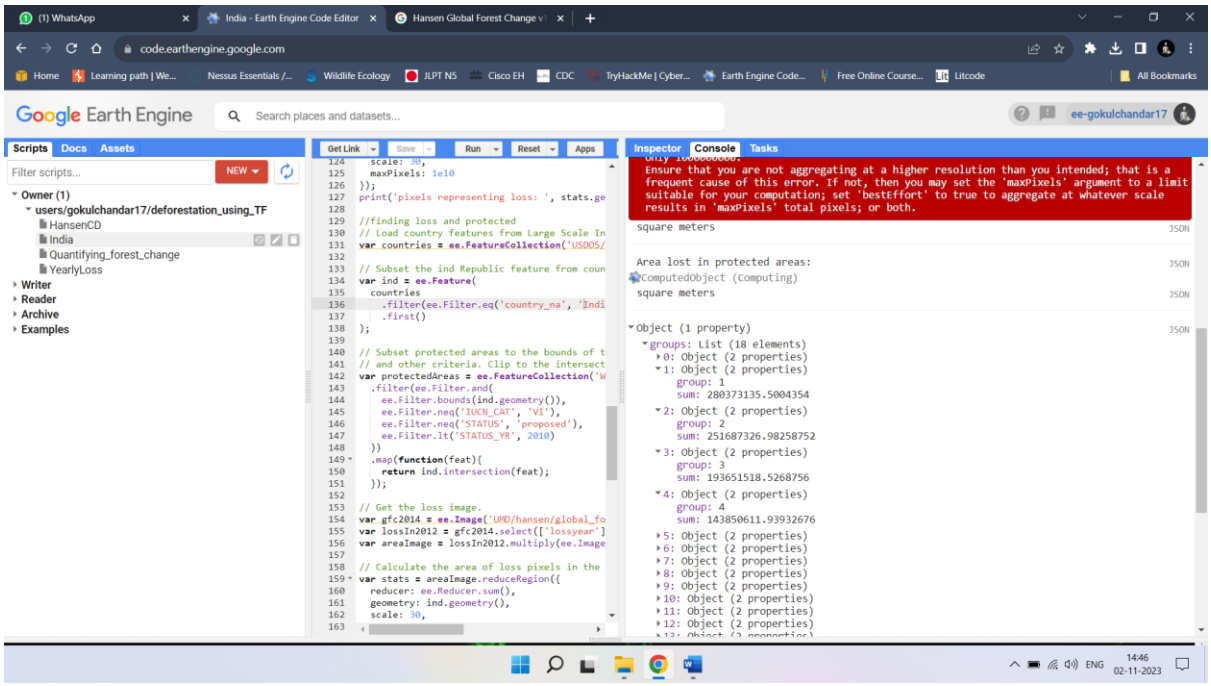
```
// Get the loss image.
var gfc2014 = ee.Image('UMD/hansen/global_forest_change_2015');
var lossIn2012 = gfc2014.select(['lossyear']).eq(12);
var areaImage = lossIn2012.multiply(ee.Image.pixelArea());

// Calculate the area of loss pixels in the India.
var stats = areaImage.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: ind.geometry(),
  scale: 30,
  maxPixels: 1e9
});
print(
  'Area lost in the India',
  stats.get('lossyear'),
  'square meters'
);

// Calculate the area of loss pixels in the protected areas.
var stats = areaImage.reduceRegion({
  reducer: ee.Reducer.sum(),
  geometry: protectedAreas.geometry(),
  scale: 30,
  maxPixels: 1e10
});
print(
  'Area lost in protected areas:',
  stats.get('lossyear'),
  'square meters'
);

////////////////////////////////
// Load country boundaries from LSIB.
var countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017');
// Get a feature collection with just the ind feature.
var ind = countries.filter(ee.Filter.eq('country_co', 'CF'));

// Get the loss image.
// This dataset is updated yearly, so we get the latest version.
var gfc2017 = ee.Image('UMD/hansen/global_forest_change_2017_v1_5');
var lossImage = gfc2017.select(['loss']);
var lossAreaImage = lossImage.multiply(ee.Image.pixelArea());

var lossYear = gfc2017.select(['lossyear']);
var lossByYear = lossAreaImage.addBands(lossYear).reduceRegion({
  reducer: ee.Reducer.sum().group({
    groupField: 1
```

```
    }),
  geometry: ind,
  scale: 30,
  maxPixels: 1e10
});
print(lossByYear);

var statsFormatted = ee.List(lossByYear.get('groups'))
  .map(function(el) {
    var d = ee.Dictionary(el);
    return [ee.Number(d.get('group')).format("20%02d"), d.get('sum')];
  });
var statsDictionary = ee.Dictionary(statsFormatted.flatten());
print(statsDictionary);

//making a chart
var chart = ui.Chart.array.values({
  array: statsDictionary.values(),
  axis: 0,
  xLabels: statsDictionary.keys()
}).setChartType('ColumnChart')
  .setOptions({
    title: 'Yearly Forest Loss',
    hAxis: {title: 'Year', format: '####'},
    vAxis: {title: 'Area (square meters)'},
    legend: { position: "none" },
    lineWidth: 1,
    pointSize: 3
  });
print(chart);
```

Output:




Google Earth Engine
Search places and datasets...
ee-gokulchandar17
Scripts
Docs
Assets
Filter scripts...
NEW
Owner (1)
users/gokulchandar17/deforestation_using_TF
HansenCD
India
Quantifying_forest_change
YearlyLoss
Writer
Reader
Archive
Examples
Get Link
Save
Run
Reset
Apps
Inspector
Console
Tasks
Object (1 property)
Object (18 properties)
2000: 0
2001: 280373135.5004354
2002: 251687326.98258752
2003: 193651518.5268756
2004: 143850611.93932676
2005: 195974628.070042
2006: 231270022.7623279
2007: 309097231.1518333
2008: 190732961.24605256
2009: 209509347.11145952
2010: 480691981.73259723
2011: 304297657.56849366
2012: 348034724.839629
2013: 606122530.6121303
2014: 782848003.1907048
2015: 506346318.7874133
2016: 1164180509.3776855
2017: 664239880.1831864
Yearly Forest Loss
Area (square meters)
Year
ENG
14:46
02-11-2023


Google Earth Engine
Search places and datasets...
ee-gokulchandar17
Scripts
Docs
Assets
Filter scripts...
NEW
Owner (1)
users/gokulchandar17/deforestation_using_TF
HansenCD
India
Quantifying_forest_change
YearlyLoss
Writer
Reader
Archive
Examples
Get Link
Save
Run
Reset
Apps
Inspector
Console
Tasks
Ensure that you are not aggregating at a higher resolution than you intended; that is a frequent cause of this error. If not, then you may set the 'maxPixels' argument to a limit suitable for your computation; set 'bestEffort' to true to aggregate at whatever scale results in 'maxPixels' total pixels; or both.
square meters
Area lost in protected areas:
ComputedObject (Computing)
square meters
Object (1 property)
groups: List (18 elements)
0: Object (2 properties)
1: Object (2 properties)
group: 1
sum: 280373135.5004354
2: Object (2 properties)
group: 2
sum: 251687326.98258752
3: Object (2 properties)
group: 3
sum: 193651518.5268756
4: Object (2 properties)
group: 4
sum: 143850611.93932676
5: Object (2 properties)
6: Object (2 properties)
7: Object (2 properties)
8: Object (2 properties)
9: Object (2 properties)
10: Object (2 properties)
11: Object (2 properties)
12: Object (2 properties)
ENG
14:46
02-11-2023