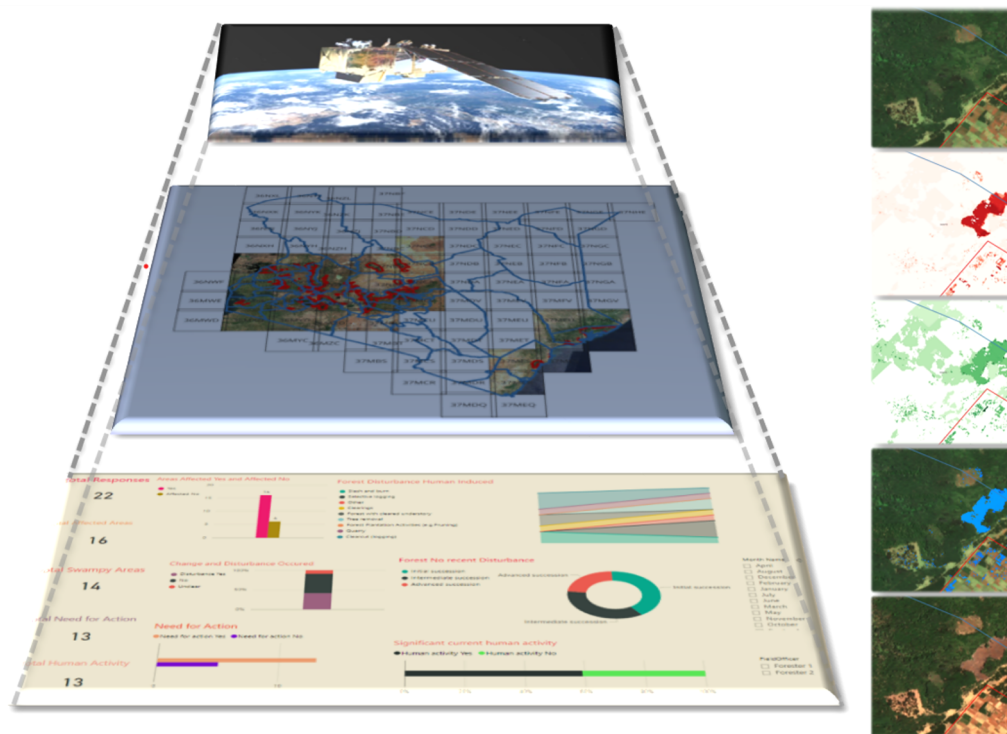


PYEO: FOREST ALERT SYSTEM

User Guide



Authors:

H. Balzter

I. Reading

M. Payne

University of Leicester 2023

TABLE OF CONTENTS

INTRODUCTION	2
REQUIREMENTS	3
INSTALLATION.....	5
Python Environment Management.....	5
PyEO Software Installation.....	6
Installation Test	6
TUTORIALS	8
Jupyter notebooks	8
OPERATION.....	9
Customisation Recommendations.....	9
Credentials Folder Creation	10
Key Folder Creation	10
Configuration	11
Pipeline Parameters	12

PyEO Workflow & Control.....	12
Workflow Step 1: Initialisation	13
Workflow Step 2: Run Configuration Logging	13
Workflow Step 3: RoI and Tile Intersection	14
Workflow Step 4: Raster Processing and Report Generation for each Sentinel-2 Tile.....	14
Workflow Step 5: Vector Analysis of Tile Raster Reports	21
Workflow Step 6: Integrate Vector Analyses to National Scope	22
Workflow Step 7: Filter National Scope Vectorised Forest Alerts	23
Workflow Step 8: Manual Filtering of National Scope Vectorised Forest Alerts.....	24
Workflow Step 9: Distribution of Manually Filtered Forest Alerts	24
PyEO Pipeline Execution.....	25
PYEO Operational Recommendations	25

PYEO DATA FOLDER STRUCTURE 27

CITATION..... 29

How to cite this software	29
---------------------------------	----

INTRODUCTION

Welcome to Python for Earth Observation Forest Alert System.

PyEO is designed to provide a set of portable, extensible, and modular Python scripts for machine learning in earth observation and GIS, including downloading, pre-processing, creation of base layers, classification and change detection. It was developed by the University of Leicester and has primarily been applied to the generation of deforestation alerts to assist in maintaining forest cover including in Guatemala, Brazil, and Kenya. This manual describes the use of release 1.0 developed under the IMPRESS project in 2022-2023 to provide forest alerts at national scale for the Kenyan Forestry Service on the SEPAL platform of the United Nations Food and Agriculture Organisation.

The software repository is available at:

<https://github.com/clcr/pyeo>

Training notebooks are available within the repository at:

<https://github.com/clcr/pyeo/notebooks>

Detailed documentation of functions is available at:

<https://clcr.github.io/pyeo/build/html/index.html>

REQUIREMENTS

To work with PyEO Forest Alert System you will need access to certain software packages (which are provided on the SEPAL platform) and also an account and access credentials to download European Space Agency (ESA) Sentinel-2 satellite imagery. From September 2023 this is provided by the Copernicus Data Space Ecosystem (CDSE).

To use the CDSE, you will need to open an account at: <https://dataspace.copernicus.eu>

Once you have created your account, you will need to record your email address and password as they will need to be entered into a credentials.ini file to allow the software to automatically download data. This is described in more detail further below.

To enable conversion of Sentinel-2 Top of Atmosphere (L1C) imagery to Bottom Of Atmosphere (L2A) images the software requires a path to an installation of the European Space Agency (ESA) conversion program Sen2Cor. This is provided pre-installed on the SEPAL platform but can also be obtained directly from <http://step.esa.int/main/third-party-plugins-2/sen2cor/> for other platforms.

The program [git](#) is also required to obtain the PyEO software from GitHub repository and to keep it up to date. This program is provided by default on the SEPAL platform and its presence and version can be verified with the command:

```
git --version
```

The program will respond with the software's version number e.g.:

```
git version 2.25.1
```

Please note that PyEO needs to make temporary use of the user's home directory when decompressing downloaded images and so it is important that at least 2Gb of free storage space are available in this folder when PyEO is downloading images. Normally the temporary folders created for this purpose are automatically deleted but if a PyEO run terminates unexpectedly or is aborted these temporary folders (with names of the form 'tmpxxxxx') may be left in the user's home folder – they can however be safely deleted manually.

INSTALLATION

PYTHON ENVIRONMENT MANAGEMENT

PyEO requires a python virtual environment containing its software package dependencies. On the **SEPAL** platform this has already been generated and is available for selection from the dropdown list of kernels in Jupyter Notebooks to enable following the tutorials as described below. When using notebooks please select the kernel named:

(venv) Python for Earth Observation (PyEO)

For standalone command line execution (outside of notebooks) a local copy of the provided virtual environment can be made into a user's home folder as follows:

```
cd /sepal-user/home  
  
cp -R /usr/local/share/jupyter/kernels/venv-  
pyeo_1/venv .
```

This virtual environment can then be activated to allow command line execution of PyEO by issuing the following command from the user's home folder:

```
source venv/bin/activate
```

The command prompt will change to be prefixed with (venv) to indicate the virtual environment has been correctly activated.

PYEO SOFTWARE INSTALLATION

Once a virtual environment has been activated a working copy of PyEO can then be cloned from GitHub and installed into this local virtual environment as follows:

```
cd ~  
  
mkdir my_working_directory  
  
cd my_working_directory  
  
git clone https://github.com/clcr/pyeo.git  
  
cd pyeo  
  
pip install -e .
```

Thereafter entering the command **pip list** at the terminal will show all installed packages including PyEO itself.

INSTALLATION TEST

You can test your installation by typing the following in the terminal:

```
python  
  
>>> import pyeo.classification
```

or, by running the same import command above, after having started a Jupyter notebook from within SEPAL (by clicking on the 'Spanner icon on the left and selecting Jupyter Lab). For more information on

the use of SEPAL, please see:
<https://docs.sepal.io/en/latest/setup/index.html>

TUTORIALS

JUPYTER NOTEBOOKS

Once installation of PyEO is complete, you can follow the tutorial notebooks, which demonstrate the utility of the PyEO library. The PyEO software repository you have downloaded includes the following set of notebooks giving step by step instructions on how to:

- *Setup the PyEO environment:*
https://github.com/clcr/pyeo/notebooks/PyEO_sepal_orientation.ipynb
- *Train a customised machine learning classifier:*
https://github.com/clcr/pyeo/notebooks/PyEO_sepal_model_training.ipynb
- *Operate the Forest Alert Pipeline to: Download Sentinel-2 Imagery, Create a Baseline Composite and Perform Automatic Change Detection*
https://github.com/clcr/pyeo/notebooks/PyEO_sepal_pipeline_training.ipynb

OPERATION

CUSTOMISATION RECOMMENDATIONS

PyEO is designed to run from its built-in folder structure whilst you become familiar with it and learn using the provided notebooks. However, when in regular or production use it is good practice to move (or reproduce) certain key files and folders so that they will not be affected by any updates to the GIT-managed content of the repository and also so that, for developers, there is no risk of accidentally exposing credentials when publishing any code modifications back up to GitHub. For the following we will assume these key files and folders will be placed at the level of the installation directory we made earlier namely 'my_working_directory' (although you may place them wherever you wish if you adjust paths accordingly).

The first step is to make a copy of the pyeo_sepal.ini file in our working directory

```
cd ~/my_working_directory  
cp pyeo/pyeo_sepal.ini .
```

The pyeo_sepal.ini file is central to PyEO's operation. It controls which parts of the pipeline will be executed and contains path variables to the key other code and data folders required by PyEO.

We can now edit and customise it and adjust paths to folders as will be described below.

An example of moving the credentials folder is set out below. It would also be good practice to similarly move your data, geometry, log, model and roi folders up to the installation level to preserve them and ensure privacy.

CREDENTIALS FOLDER CREATION

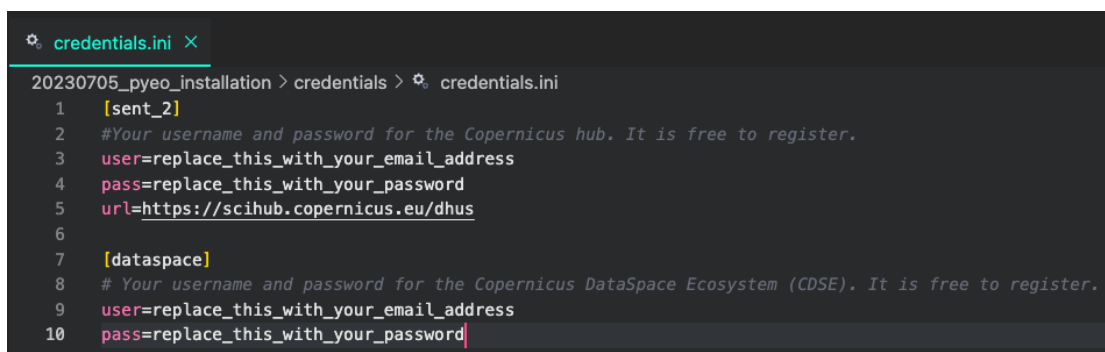
To use the CDSE, you will need your account details obtained by registering at <https://dataspace.copernicus.eu>

Once you have created your account, you need to:

1. Create a copy of the credentials folder in my_working_directory above pyeo

```
cd ~/my_working_directory  
cp -R pyeo/credentials .
```

2. Rename the contained file credentials_dummy.ini to credentials.ini
3. Enter your email address and password into the dataspace section of the credentials.ini file, following the pattern shown below:



```
credentials.ini X  
20230705_pyeo_installation > credentials > credentials.ini  
1  [sent_2]  
2  #Your username and password for the Copernicus hub. It is free to register.  
3  user=replace_this_with_your_email_address  
4  pass=replace_this_with_your_password  
5  url=https://scihub.copernicus.eu/dhus  
6  
7  [dataspace]  
8  # Your username and password for the Copernicus DataSpace Ecosystem (CDSE). It is free to register.  
9  user=replace_this_with_your_email_address  
10 pass=replace_this_with_your_password
```

KEY FOLDER CREATION

As indicated above it would also be good practice to similarly move your data, geometry, integrated, log, model and roi folders up to the

installation level to preserve them and ensure privacy. This can be achieved by following the pattern show below for each of the above-listed folders:

```
cd ~/my_working_directory  
cp -R pyeo/data .
```

CONFIGURATION

Firstly, we need to edit our initialisation file `pyeo_sepal.ini` to point to the folders we created above in the `[environment]` section, as follows.

```
[environment]  
# pyeo_dir needs to be an absolute path to the installed pyeo_code  
pyeo_dir = /home/sepal-user/20230626_pyeo_installation/pyeo_1  
# tile_dir needs to be an absolute path to the output data directory  
tile_dir = /home/sepal-user/20230626_pyeo_installation  
  
# Relative paths are relative to pyeo_dir\  
integrated_dir = ../integrated  
roi_dir = ../roi  
roi_filename = kfs_roi_subset_c.shp  
geometry_dir = ../geometry  
s2_tiles_filename = kenya_s2_tiles.shp  
log_dir = ../log  
log_filename = sepal_venv.log  
credentials_path = ../credentials/credentials.ini  
  
environment_manager = venv  
  
# Path to the sen2cor preprocessor script, L2A_Process. Usually in the bin/ folder  
# of your sen2cor installation.  
sen2cor_path = /bin/L2A_Process
```

We can then edit the remainder of the `pyeo_sepal.ini` file to configure the pipeline parameters and enable the sections we wish to activate when we run as described in the following sections.

A Note on .ini file encoding on Windows: Although this guide is for SEPAL users please note that if users are running on Windows OS then after editing (for example with the Notepad app) the file `pyeo_windows.ini` will need to be saved after selecting the ANSI encoding option from the dropdown list of options instead of the

default of UTF-8. Please see this page for more details:
<https://stackoverflow.com/questions/13282189/missingsectionheadererror-file-contains-no-section-headers>

PIPELINE PARAMETERS

The overall parameters for the pipeline can then be setup using the [forest_sentinel] section in the pyeo_sepal.ini file. Here the date range of interest for forest alert generation is specified by the start_date and end_date and the reference time period to be used as an historical reference (assumed pre-change of forest cover) is specified as being between the dates assigned to composite_start and to composite_end.

```
[forest_sentinel]

# Acquisition dates in the form yyyymmdd
start_date=20230101
end_date=20230131

# Dates to download and preprocess for the initial cloud-free composite
composite_start=20220101
composite_end=20221231

# EPSG code for Kenya - north of equator and east of 36°E is EPSG:21097
# See https://epsg.io/21097 and https://spatialreference.org/ref/epsg/21097/
epsg=21097

# Cloud cover threshold for imagery to download
cloud_cover=25

# Certainty value above which a pixel is considered a cloud from sen2cor
cloud_certainty_threshold=0

# path to the trained machine learning model for land cover in Kenya
model= ./models/model_36MYE_Unoptimised_20230505_no_haze.pkl
```

The other required fields are described in the accompanying comment text in the pyeo_sepal.ini file as can be seen in the excerpt above.

PYEO WORKFLOW & CONTROL

The PyEO national forest alert system works under control of the pyeo_sepal.ini file. Based on this file it follows a workflow comprising

the steps described below drawing parameters from the initialisation file. It also reads activation flags of the form 'do_activity' and executes them selectively according to whether these values are set to True or False.

The Python functions which enact the workflow steps can be found in the entry point function `automatic_change_detection_national()` in `pyeo/acd_national.py`. This top-level function can be called from the command line using `pyeo/run_acd_national.py` passing it an absolute path to the `pyeo_sepal.ini` file you wish it to make use of.

In the workflow step descriptions, which follow below, we identify:

- *The input parameters and activation flags*
- *The key functions used to execute the pipeline stage*
- *The processed output files generated and the position where these outputs are stored in the file system*

Please note the overall workflow described incorporates both automated pipeline stages as well as manual steps to be undertaken by relevant experts of a forest management team.

Workflow Step 1: Initialisation

In this workflow step PyEO reads the pipeline configuration specified in the `pyeo_sepal.ini` file into an internal python dictionary.

```
config_dict, acd_log = acd_initialisation(config_path)
```

Workflow Step 2: Run Configuration Logging

This step simply streams the configuration python dictionary in human readable form to a log file to act as a record of the run status. The log file storage position is specified in `pyeo_sepal.ini` by the following lines:

```
log_dir = ./log
log_filename = sepal_venv.log
```

The work is carried out by:

```
acd_config_to_log(config_dict, acd_log)
```

Workflow Step 3: RoI and Tile Intersection

PyEO Forest Alerts accepts a single shapefile specifying the region of interest to be monitored. The shapefile to use is set by these parameters in pyeo_sepal.ini

```
roi_dir = ./roi  
roi_filename = kfs_roi_subset_c.shp
```

This step is activated/deactivated by setting do_intersection = True/False in the initialisation file

```
do_tile_intersection = True
```

This processing stage is carried out by:

```
tilelist_filepath =  
acd_roi_tile_intersection(config_dict, acd_log)
```

The output generated is a file tilelist.csv in pyeo/roi containing a list of all Sentinel-2 tiles that intersect with the region of interest covered by the shapefile. This list is then used to guide the download and processing of these required tiles on a tile-by-tile basis in later steps.

Workflow Step 4: Raster Processing and Report Generation for each Sentinel-2 Tile

PyEO Forest Alerts now processes each tile in tilelist according to the pipeline parameters to generate a report representing areas of potential forest change.

```
[forest_sentinel]
```

```
# Acquisition dates in the form yyyymmdd
start_date=20230101
end_date=20230131

# Dates to download and preprocess for the initial cloud-free composite
composite_start=20220101
composite_end=20221231

# EPSG code for Kenya - north of equator and east of 36°E is EPSG:21097
# See https://epsg.io/21097 and https://spatialreference.org/ref/epsg/21097/
epsg=21097

# Cloud cover threshold for imagery to download
cloud_cover=25

# Certainty value above which a pixel is considered a cloud from sen2cor
cloud_certainty_threshold=0

# path to the trained machine learning model for land cover in Kenya
model= ./models/model_36MYE_Unoptimised_20230505_no_haze.pkl
```

Specific raster processing parameters can also be set in `pyeo_sepal.ini` to control this stage:

```
# ***** STEP 4 SETUP GENERAL RASTER PROCESSING PARAMETERS *****
# download_source = scihub
download_source = dataspace
# granules below this size in MB will not be downloaded
faulty_granule_threshold = 350
# list of strings with the band name elements of the image file names in "" string
notation
# the wavebands specified here must match those used to build the random forest
model specified in the Classify section below
band_names = ["B02", "B03", "B04", "B08"]
# file name pattern to search for when identifying band file locations in "" string
notation
resolution_string = "10m"
# spatial resolution of the output raster files in metres. Can be any resolution,
not just 10, 20 or 60 as in the default band resolutions of Sentinel-2
output_resolution = 10
```

The raster processing step is activated/deactivated by setting `do_raster = True/False` in the initialisation file

```
do_raster = True
do_skip_existing = True
do_quicklooks = False
```

This processing stage is carried out by:

```
acd_integrated_raster(config_dict, acd_log,  
tilelist_filepath...)
```

Several sub-steps can be identified in this process and these can be selectively enabled and disabled on any given run. This may, for example, be useful if only composite baselines are required or, if an earlier processing run was interrupted, as a means to only repeat steps that remained incomplete. The sub-steps are listed below:

Raster-Processing Sub-Steps

STEP 4a: Download Reference Images and Build a Median Composite

This step downloads historical images and combines them with a median function into a cloud-free composite. It can be selectively enabled with the *do_build_composite* flag

```
# ***** STEP 4a DOWNLOAD REFERENCE IMAGES AND BUILD A MEDIAN COMPOSITE *****  
do_build_composite = True  
# set buffer in number of pixels for dilating the SCL cloud mask (recommend 10  
pixels of 10 m) for the composite building  
buffer_size_cloud_masking_composite = 10  
# maximum number of images to be downloaded for compositing, in order of least  
cloud cover  
download_limit = 10
```

STEP 4b: Download Change Detection Images for the Required Date Range

This step acquires (downloads) recent images in which to detect changes and can be selectively enabled with the *do_download* flag

```
# ***** STEP 4b: DOWNLOAD CHANGE DETECTION IMAGES FOR THE REQUIRED DATE RANGE *****  
do_download = True  
# set buffer in number of pixels for dilating the SCL cloud mask (recommend 30  
pixels of 10 m) for the change detection  
buffer_size_cloud_masking = 20
```

STEP 4c: Classify the Composite and Change Detection Images

This step applies the random forest model to classify land use in the composite and change images and can be selectively enabled with the *do_classify* flag

```
# ***** STEP 4c: CLASSIFIY THE COMPOSITE AND CHANGE DETECTION IMAGES *****
do_build_prob_image = False
# do_build_prob_image, consider removing
do_classify = True
# list of strings with class labels starting from class 1. Must match the trained
model that was used.
class_labels = ["primary forest", "plantation forest", "bare soil", "crops",
"grassland", "open water", "burn scar", "cloud", "cloud shadow", "haze", "sparse
woodland", "dense woodland", "artificial"]
# if sieve is 0, no sieve is applied. If >0, the classification images will be
sieved using gdal and all contiguous groups of pixels smaller than this number will
be eliminated
sieve = 0
```

STEP 4d: Detect Changes and Build Raster Reports

This step compares the classified images to detect unwanted changes in forest cover. It can be selectively enabled with the *do_change* flag

```
# ***** STEP 4d: DETECT CHANGES AND BUILD RASTER REPORTS *****
do_change = True
# find subsequent changes from any of these classes. Must match the trained model
that was used.
change_from_classes = [1, 2]
# to any of these classes. Must match the trained model that was used.
change_to_classes = [3]
```

Raster Processing Output Layers

The output generated by the above raster processing stages is a stacked GeoTIFF report file produced on a tile-by-tile basis which can be found in the output/probabilities folder within PyEO's data directory structure (see PyEO DATA Folder Structure section). These layers can be viewed in conjunction to understand land use changes using GIS tools such as QGIS. For completeness all the layers are described below as they may aid forestry analysts in determining the time and cause of forest disturbances however please note that the automated analysis and vectorisation described in later steps of the PyEO pipeline condense this large volume of information into a simple tabular output (national_geodataframe.shp) that can be viewed and

manipulated in QGIS and similar GIS tools as a way of rapidly identifying areas of land use change.

Table: Time-Series Analysis: Report Layer Structure Guide

LAYER	DESCRIPTION
1	Total Image Count: The number of images processed per pixel - number of available images within overall cloud percentage cover limit set in a .ini file
2	Occluded Image Count: Counts number of cloud occluded (or out-of-orbit) images that are thus unavailable for classification and analysis
3	Classifier Change Detection Count: Count if a from/to change of classification was detected
4	First-Change Trigger for Combined Classifier+dNDVI Validated Change Detection: Earliest classification change date where (not missing data & not cloud)
5	Combined Classifier+dNDVI Validated Change Detection Count: Count if a change was detected after a first change has already been detected
6	Combined Classifier+dNDVI Validated No Change Detection Count: Count if a no change was detected after a first change has already been detected
7	Cloud Occlusion Count: Count if a cloud occlusion (or out-of-orbit) occurred after a first change has already been detected
8	Valid Image Count: Total number of valid (no cloud) images for this pixel since first change was detected
9	Change Detection Repeatability: Repeatability of change detection after first change is detected - as a percentage of available valid images
10	Binary time-series decision: Based on percentage_probability_threshold and minimum_required_validated_detections_threshold
11	Binary time-series decision by first-change date: First change date masked by Binary Decision
12	dNDVI Only Change Detection Count: Count if a change was detected by the dNDVI test and that not cloud occluded (or out-of-orbit)
13	Binary time-series decision: Based on dNDVI Only and minimum_required_dNDVI_detections_threshold
14	Binary time-series decision: Based on Classifier Only
15	Combined Classifier+dNDVI Binary time-series decision: Based on Classifier AND dNDVI opinion
16	FROM Classification Count
17	TO Classification Count
18	Binary Decision Thresholds on FROM and TO Classification Counts

In the above table NDVI refers to the Normalised Difference Vegetation Index, a well-known measure of plant cover derived from multi-spectral image data. The term dNDVI refers to the difference in NDVI between the baseline composite reference image and that of a given change monitoring image. This differential dNDVI measure is used as an additional check to the classifier's detection of a change in land use to provide confirmation that there has been vegetation loss in a given region. Layer 15 *Combined Classifier+dNDVI Binary time-series decision: Based on Classifier AND dNDVI opinion* contains the output of the system based on these combined criteria and is used for the generation of vectorised polygon output. The separated outputs are available in other layers in case they are of use to users of the system in investigating the causation of alerts. With this in mind the raw processed NDVI and dNDVI images are also available in the output/probabilities directory to allow users to view them directly (as GeoTIFF files). To save storage space these files can be deleted if they are not required.

Layers 16 & 17 maintain a count of the number of times a region has registered as one of the FROM land use classes (e.g. Forest) and as one of the TO land use classes (e.g. Bare Earth) respectively. Layer 18 holds a binarization based on the threshold criteria described below. These outputs were generated for research purposes but may be of interest to forest analysts as they are independent of the composite baseline and if the system is run over a long time series will offer an alternative view of where transitions from forested to non-forested land use may have occurred.

Raster Processing Binarisation Criteria & Parameters

The production of the binarized decision layer 15 requires threshold criteria to be applied to the accumulated time-series evidence for each pixel of the image footprint. This step is used to enable forest analysts to rapidly identify the most likely areas of change on which to focus their investigations. It is also the basis for vectorisation of change areas in the next processing stages which enables a national picture to be built to produce the final national geo-dataframe

summarising change in regions of interest such as protected national parks. The parameters used are described below and can be found and changed in the code if required.

Table: Time-Series Analysis: Binarisation Criteria

Absolute number of valid detections for classifier opinion to be accepted. minimum_required_validated_detections_threshold = 5
Absolute number of dNDVI change detections for classifier opinion to be accepted. minimum_required_dNDVI_detections_threshold = 5
Absolute number of classifier-only detections for opinion to be accepted minimum_required_classifier_detections_threshold = 5
Minimum repeatability of detection after first detection percentage_probability_threshold = 50
Minimum number of matches to Change_From classes minimum_required_FROM_detections_threshold = 2
Minimum number of matches to Change_To classes minimum_required_TO_detections_threshold = 2

Workflow Step 5: Vector Analysis of Tile Raster Reports

In this step we now vectorise the report layers generated for each of the Sentinel-2 tiles that have been processed by step 4 turning each 4-connected cluster of pixels into a bounding vector polygon and measuring these polygons (in particular to extract the area and centroid). The centroid is then computed and used to assign each polygon to administrative areas of Kenya in the example to assist foresters in assigning different areas for analysis to staff.

```
# ***** STEP 5: VECTOR ANALYSIS OF TILE RASTER REPORTS *****
[vector_processing_parameters]
level_1_filename = gadm41_KEN_1.json
# vectorisation currently hardcoded to use level_1_filename
level_2_filename = gadm41_KEN_2.json
level_3_filename = gadm41_KEN_3.json

do_delete_existing_vector = True
```

This step is activated/deactivated by setting `do_vectorise = True/False` in the initialisation file

```
do_vectorise = True
```

This processing stage is carried out by:

```
acd_integrated_vectorisation(  
    log=acd_log,  
    tilelist_filepath=tilelist_filepath,  
    config_path=config_path
```

On completion there PyEO will output a shape file for each tile in its `/output/probabilities` folder containing polygons and metrics for all potential forest alerts.

Workflow Step 6: Integrate Vector Analyses to National Scope

In this step PyEO builds a national picture of deforestation activity by combining the vectorised alerts for each tile into a single output data file called `national_geodataframe.shp` which is placed in the folder specified by `'tile_dir'`. In addition, if the parameter `write_kml` is set to `'True'` a kml format file will also be produced.

This step is activated/deactivated by setting `do_integrate = True/False` in the initialisation file

```
# ***** STEP 6: INTEGRATE VECTOR ANALYSES TO NATIONAL SCOPE *****  
do_integrate = True
```

This processing stage is carried out by:

```
acd_national_integration(  
    root_dir=config_dict["tile_dir"],
```

```
log=acd_log,  
epsg=config_dict["epsg"],  
config_dict=config_dict,  
write_kml=True  
)
```

Upon completion of this step PyEO will have output the combined vectorised alerts for each tile into a single output data file called `national_geodataframe.shp` which is placed in the `'tile_dir'` folder specified in `pyeo_sepal.ini` .

Note: The shape file generated per tile in Step 5 contains polygons for all alert regions covered by the tile regardless of the region of interest whereas the `national_geodataframe` shape file filters polygons from the tiles to only select those that fall within the user's specified region of interest as set by the parameter `roi_filename` in `pyeo_sepal.ini`.

Workflow Step 7: Filter National Scope Vectorised Forest Alerts

```
# ***** STEP 7: FILTER NATIONAL SCOPE VECTORISED FOREST ALERTS *****  
do_filter = False  
# If there are any strings within counties_of_interest list, filtering by county  
# will be attempted  
counties_of_interest = []  
# Counties_of_interest = ["Kwale", "TransNzoia"]  
minimum_area_to_report_m2 = 120
```

This step is activated/deactivated by setting `do_filter = True/False` in the initialisation file

```
do_filter = False
```

This processing stage is carried out by:

```
acd_national_filtering(  
    log=acd_log,
```

```
config_dict=config_dict  
)
```

This step can optionally be used to further filter the `national_geodataframe.shp` file by criteria such as county of interest or the minimum disturbed area (based on each Sentinel-2 pixel covering 10m x 10m or 100m²)

Workflow Step 8: Manual Filtering of National Scope Vectorised Forest Alerts

This workflow step represents a manual step whereby GIS and forestry experts select a subset of the automatically generated alerts for on-site investigation by teams on the ground. To achieve this the `national_geodataframe.shp` file (perhaps additionally filtered in Step 7) can be loaded into the QGIS tool and viewed as a table. Table columns have been included to allow these experts to identify themselves (`'user'`), flag alerts that should be investigated (`'follow_up'`), indicate the type of event (`'eventClass'`) and to add notes on their reasoning (`'comments'`).

Workflow Step 9: Distribution of Manually Filtered Forest Alerts

Finally, forest alerts flagged for further investigation in Step 8 can be filtered from the `national_geodataframe` (using the `geopandas` python library) distributed to teams of rangers on the ground for investigation and reporting on the nature of the disturbance found. Distribution to remote teams can be effected by the use of mobile phone applications such as WhatsApp and could in principle be automated with additional Python code.

PYEO PIPELINE EXECUTION

PyEO can be run **interactively** in the Jupyter Notebooks provided in the Tutorials but for production use it will typically be run as a standalone application via the Terminal.

Once the pyeo_sepal.ini file has been configured with the parameters and run options required we can run PyEo from the command line in a terminal as follows:

First, move to where PyEO is installed:

```
cd ~/my_working_directory/pyeo
```

Then we can invoke python to run the script run_acd_national.py within the folder pyeo which will start PyEO pipeline execution. For this to work we must pass in the absolute path to the pyeo_sepal.ini initialisation file:

```
python pyeo/run_acd_national.py  
<insert_your_absolute_path_to>/pyeo_sepal.ini
```

PYEO OPERATIONAL RECOMMENDATIONS

When using PyEO to process tiles over a large geographical scope covering more than a few tiles it will be a lengthy process when they are first run as large volumes of data must be downloaded and processed to generate the composite reference. It is recommended therefore that large areas be split into multiple shape files each covering at most 2-3 tiles and that these be run sequentially to establish the baseline composite for such groups of tiles progressively. This will make it easier for the user to manage and observe any errors that may occur without impacting the processing of too many other tiles. Once the tile folders for the full geographic range are all established it is then practical to ask PyEO to work from a shape file

covering the entire geographical scope for relatively less-demanding tasks such as generating weekly update reports.

PYEO DATA FOLDER STRUCTURE

PyEO generates a specific folder hierarchy when downloading, processing and analysing Sentinel-2 imagery. An example of a typical structure is illustrated in the schematic below for a pipeline run covering two sentinel-2 tiles '36NXG' and '36NYG' respectively.

This data structure will be generated in and below the folder specified as 'tile_dir' in the pyeo_sepal.ini file. The `national_geodataframe.shp` file is created in this `tile_dir` alongside folder hierarchies for each Sentinel-2 tile that has been processed. The schematic indicates the folders that are created (in bold type) and also where particular files generated in the PyEO pipeline can be found within these folders by including typical example filenames and types (in italic type)

```
|tile_dir
| |national_geodataframe.shp
| |36NXG
| | |composite
| | | |composite_T36NXG_20221202T075301.tif
| | | |L1C
| | | |L2A
| | | |cloud_masked
| | | |S2A_MSIL2A_20220106T075321_N0301_R135_T36NXG_20220106T090414.tif
| | |images
| | | |L1C
| | | |S2A_MSIL1C_20230111T075301_N0509_R135_T36NXG_2023....SAFE
| | | |L2A
| | | |S2A_MSIL2A_20230101T075331_N0509_R135_T36NXG_2023....SAFE
```

- | | | **cloud_masked**
- | | | | *|S2A_MSIL2A_20230131T075141_NA509_R135_T36NXG_20230131T131152.tif*
- | | **log**
- | | | *|36NXG_log*
- | | **output**
- | | | **classified**
- | | | | *|composite_T36NXG_20221202T075301_class.tif*
- | | | |
- | *|S2A_MSIL2A_20230101T075331_NA509_R135_T36NXG_20230101T110554_class.tif*
- | | | **probabilities**
- | | | | *|change_20221202T075301_36NXG_20230101T075331.tif*
- | | | | *|NDVI_20221202T075301_36NXG_20230101T075331.tif*
- | | | | *|dNDVI_20221202T075301_36NXG_20230101T075331.tif*
- | | | | *|report_20221202T075301_36NXG_20230605T074619.tif*
- | | | | *|report_20221202T075301_36NXG_20230605T074619.shp*
- | | | **quicklooks**
- | | | | *|S2A_MSIL2A_20220106T075321_N0301_R135_T36NXG_20220106T090414.png*
- | **36NYG**
- | | **composite**
- | | | **L1C**
- | | | **L2A**

etc ...

CITATION

HOW TO CITE THIS SOFTWARE

Please use the following references when using PyEO:

Roberts, J.F., Mwangi, R., Mukabi, F., Njui, J., Nzioka, K., Ndambiri, J.K., Bispo, P.C., Espirito-Santo, F.D.B., Gou, Y., Johnson, S.C.M. and Louis, V., 2022. Pyeo: A Python package for near-real-time forest cover change detection from Earth observation using machine learning. *Computers & Geosciences*, 167, p.105192.

Roberts, J., Balzter, H., Gou, Y., Louis, V., Robb, C., 2020. Pyeo: Automated satellite imagery processing. <https://doi.org/10.5281/zenodo.3689674>

Pacheco-Pascagaza, A.M., Gou, Y., Louis, V., Roberts, J.F., Rodríguez-Veiga, P., da Conceição Bispo, P., Espírito-Santo, F.D., Robb, C., Upton, C., Galindo, G. and Cabrera, E., 2022. Near real-time change detection system using Sentinel-2 and machine learning: A test for Mexican and Colombian forests. *Remote Sensing*, 14(3), p.707.