Computers & Geosciences 167 (2022) 105192

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Pyeo: A Python package for near-real-time forest cover change detection from Earth observation using machine learning

J.F. Roberts [a,1], R. Mwangi [c], F. Mukabi [c], J. Njui [c], K. Nzioka [c], J.K. Ndambiri [c], P.C. Bispo [a,2], F.D.B. Espirito-Santo [a], Y. Gou [a,3], S.C.M. Johnson [a], V. Louis [a,4], A.M. Pacheco-Pascagaza [a,2], P. Rodriguez-Veiga [a,b], K. Tansey [a], C. Upton [a], C. Robb [a,5], H. Balzter [a,b,*]

[a] University of Leicester, Institute for Environmental Futures, Centre for Landscape and Climate Research, School of Geography, Geology and the Environment, University Road, Leicester, LE1 7RH, UK
[b] National Centre for Earth Observation, University of Leicester, Space Park Leicester, 92 Corporation Road, Leicester, LE4 5SP, UK
[c] Kenya Forest Service, Nairobi, Kenya

ARTICLE INFO

Keywords:
Earth observation
Machine learning
Change detection
Forest monitoring

ABSTRACT

Monitoring forest cover change from Earth observation data streams in near-real-time presents a challenge for automated change detection by way of a continuously updated big dataset. Even though deforestation is a significant global problem, forest cover changes in pairs of subsequent images happen relatively infrequently. Detecting a change can require the download and processing of tens, hundreds or even thousands of images. In geoscientific applications of Earth observation, machine learning algorithms are increasingly used. Once trained, a machine learning model can be applied to new images automatically.

This paper introduces the open-access Python 3 package Pyeo - "Python for Earth Observation". Pyeo provides a set of portable, extensible and modular Python functions for the automation of machine learning applications from Earth observation data streams, including automated search and download functionality, pre-processing and atmospheric correction, re-projection, creation of thematic base layers and machine learning classification or regression. Pyeo enables users to train their own machine learning models and then apply the models to newly downloaded imagery over their area of interest. This paper describes in detail how Pyeo works, its requirements, benefits, and a description of the libraries used. An application to the automated forest cover change detection in a region in Kenya is given. Pyeo can be used on cloud computing architectures such as Amazon Web Services, Microsoft Azure and Google Colab to provide scalable applications and processing solutions for the geosciences.

1. Introduction

In the era of 'big data' from satellite Earth observation (EO), a growing number of downstream applications require fully automated processing chains of satellite data. In the past decade, machine learning algorithms have been used increasingly to turn satellite imagery from optical or Synthetic Aperture Radar (SAR) systems into information that is relevant to the users. For example, the European Union's Copernicus initiative and its Sentinel satellite missions provide free daily images globally (Malenovskỳ et al., 2012). The unprecedented data availability is particularly important for applications to forest change detections after deforestation, logging and forest fires and the use of deforestation

* Corresponding author. University of Leicester, Institute for Environmental Futures, Centre for Landscape and Climate Research, School of Geography, Geology and the Environment, Space Park Leicester, 92 Corporation Road, Leicester, LE4 5SP, UK.
E-mail address: hb91@leicester.ac.uk (H. Balzter).
[1] Present address: Stiftelsen Flowminder, Regus, Cumberland House, Grosvenor Square, Southampton, SO15 2BG, UK.
[2] Present address: Department of Geography, School of Environment, Education and Development, University of Manchester, Oxford Rd, Manchester, M13 9 PL, UK.
[3] Present address: Laboratory of Geo-Information Science and Remote Sensing, Wageningen University & Research, Droevendaalsesteeg 3, 6708 PB Wageningen, The Netherlands.
[4] Present address: Rural Payments Agency, PO Box 69, Reading, RG1 3YD, UK.
[5] Present address: UK Centre for Ecology & Hydrology, Environment Centre Wales, Deiniol Road, Bangor, Gwynedd, LL57 2UW, UK.

https://doi.org/10.1016/j.cageo.2022.105192
Received 22 February 2022; Received in revised form 6 July 2022; Accepted 7 July 2022
Available online 9 July 2022

alerts can lead to substantial reductions in deforestation in Africa (Moffette et al., 2021). At the COP26 climate conference in Glasgow in 2021, 141 nations committed to ending deforestation by 2030 – together covering over 90% of the world's forest area. This clearly demonstrates the need for rapid access to information on forest change. Current forest detection methods process satellite images at daily or weekly time scales. This short time-scale requires an automatic method to download and process satellite images to manage workloads by human interpreters.

The Pyeo Python package (Roberts et al., 2020) provides near-real-time analyses of satellite images in less than 24 h of data acquisition. 'Near-real-time' in this context means that the software automatically queries a satellite data hub using its API and enters into the satellite image processing chain every time a new image has been acquired. The Sentinel-2 satellites provide a new free-and-open image acquisition every 5 days over all global land areas. Planet imagery is acquired daily over the whole world but it is only available in near-real-time-mode at a prize. Pyeo uses the last available cloud-free pixel before the current acquisition date to detect spectral changes in comparison with the base layer composite. The majority of satellite images obtained in this way are likely not detecting any changes, but the few images that do make a big difference to the users as they enable them to take rapid action in areas of unlicensed logging. Pyeo has been applied in Kenya (described in this paper) but also in Colombia and Mexico (Pacheco-Pascagaza et al., 2022).

Google Earth Engine provides a very widely used processing platform for satellite images, but it is limited in terms of its machine learning capability compared to the Python library Scikit-Learn, which allows far more parameter customisation than Google Earth Engine. In addition, where operational change detections from satellite imagery are required for national monitoring schemes, governments often do not want to depend on a private company such as Google for its continuity. Pyeo is implemented in Python, which was not fully supported on Google Earth Engine at the time we developed the software. Google Earth Engine also does not support the integration of the Sen2Cor atmospheric correction software, but pyeo does. While Google Earth Engine provides some functionality for machine learning, its programming interface allows only a limited range of customisation options for its random forest classification algorithm, whereas pyeo (inherited from Scikit-Learn) allows a full customisation. The performance of pyeo depends on the computing platform it runs on. We did not compare its performance to the Google Earth Engine platform because it is not possible to choose the exact same processing options for atmospheric correction from L1C to L2A products and the same machine learning options.

The development of Pyeo was motivated by the co-design of an operational deforestation alert system with stakeholders in Kenya in the REDD + Round Table, with a view to improving the national forest monitoring system (Kenya Ministry of Environment and Natural Resources, 2016). Pyeo uses machine learning for image change detection, automation of image queries and downloads and processing of new satellite image acquisitions for the delivery of information to end users. Pyeo currently enables the query, download and processing of Sentinel-2, Landsat and Planet images and it can easily incorporate other satellite data. Although several operational applications of forest monitoring from Landsat images are currently available (most notably Global Forest Watch; Hansen et al., 2013), few operational applications make use of the new high-resolution images from Sentinel-2. With the launch of the Sentinel-2 satellites, land cover change can in principle be monitored in near-real-time if the processing can be fully automated. With Sentinel-2A and 2B, the Sentinel-2 satellites provide multispectral imagery at up to 10 m spatial resolution every 5 days, an improvement over the combined Landsat 8 and 9 revisit time every 8 days at 30 m resolution.

Machine learning algorithms are increasingly being used to analyse satellite imagery (Fernández-Delgado et al., 2014). review 179 machine learning classifiers from 17 families of algorithms including Bayesian neural networks, support vector machines, random forests, generalized linear models and other methods. They evaluate these algorithms with 121 datasets and conclude that random forests (Breiman, 2001) generally performed best (Fernández-Delgado et al., 2014).

Pyeo has been designed to give a number of options of machine learning methods based on the widely-used Scikit-Learn library (Pedregosa et al., 2011). Scikit-Learn does not have any specific functionality to handle geographic data that are characterised by metadata describing the location of the dataset, its map projection and pixel size, in the case of rasters. Generic Python packages to support the processing of geographic data include RasterIO (Gillies et al., 2013), which is a higher-level abstraction of GDAL's raster functions (GDAL/OGR contributors, 2021) and Fiona for vector data processing (Gillies et al., 2013) based on GDAL functionality.

Existing software libraries for the analysis of remote sensing images include RSGISLib (Bunting et al., 2014), which contains C and Python algorithms for image segmentation, object-based classification, image-to-image registration, image filtering, zonal statistics, and general raster and vector processing functionality. At the time of developing Pyeo, RSGISLib did not provide machine learning functionality and specific functions for Sentinel-2 imagery.

The BFAST ('Breaks For Additive Season and Trend') package for R integrates the decomposition of a time series into trend, season, and residual terms for the purpose of change detection (Verbesselt et al., 2010); http://bfast.r-forge.r-project.org). There is also a spatial version of BFAST for processing time series of raster data, called 'bfastSpatial' for R (Dutrieux and DeVries, 2014); http://github.com/loicdtx/bfastSpatial). TIMESAT (Jönsson and Eklundh, 2004; http://web.nateko.lu.se/timesat) is another software package for analysing time series of satellite images. It was developed for analysing the seasonality of satellite time series data and their relationship with vegetation phenology, originally from AVHRR Normalized Difference Vegetation Index (NDVI) data but has since then been expanded to include other satellite missions. Pytorch (Ketkar, 2017) is a Python library for artificial intelligence (AI) applications to images of any kind and was developed for convolutional neural network modelling (Afzaal et al., 2021) and other AI techniques.

The Pyeo package introduced in this paper provides an open-source Python toolbox for change detection and image processing for satellite data from Sentinel-2, Landsat and Planet in its current form, although future releases may include other missions. Contributions to the software made via Github are welcome. The innovative value of this software library lies in its integration of the functionality of previous Python libraries and the Sen2Cor atmospheric correction software to create a fully automated image processing chain for change detection. This enables the near-real-time application of machine learning methods to a continuous stream of satellite remote sensing images for rapid change detection in forestry and for other applications. The software does not attempt to supersede any pre-existing libraries but it rather presents an innovative way of integrating them.

2. Methodology

Pyeo was designed to provide a library of generic Python functions that can be used to build operational satellite image processing chains to produce updatable thematic change maps for a defined area of interest (AOI). In its current version, Sentinel-2, Landsat and Planet satellite images are supported. In applications of satellite remote sensing data streams to geoscientific applications for the purpose of change detection, several challenges have to be overcome. First, the software needs to be capable of automating the querying of the data server for new image acquisitions over the area-of-interest and download new images for processing. Second, it needs to choose and apply appropriate preprocessing steps to the new image to make it radiometrically consistent with the previously processed images. Third, appropriate image pairs have to be combined (or stacked) in order to detect changes. And

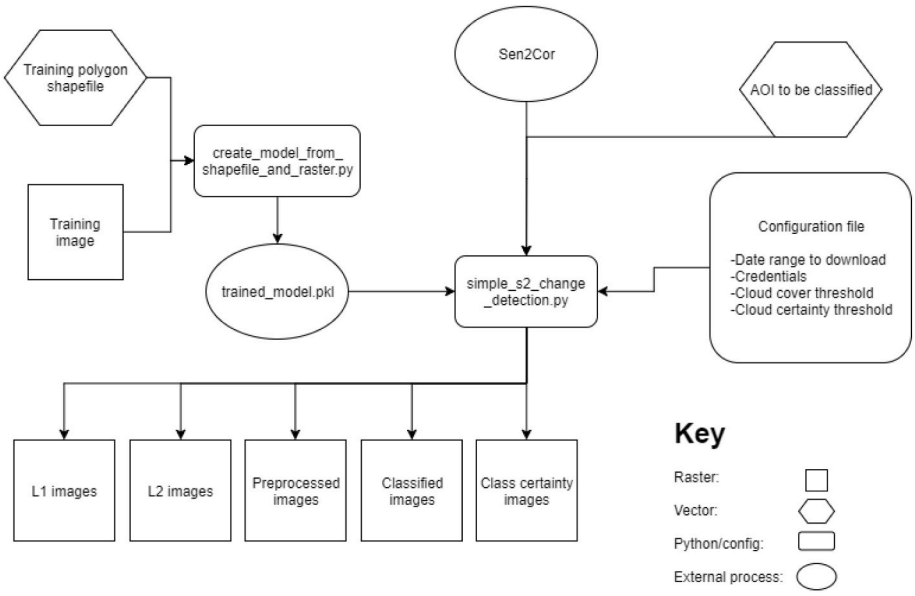

Fig. 1. Schematic example of a simple Sentinel-2 satellite image processing chain using Pyeo.

fourth, a trained machine learning model or other change detection algorithm needs to be applied to the stack of images.

There are three layers of abstraction available in Pyeo: a program that will download, stack and classify images for change detection, a set of shorter scripts that perform single steps in the downloading/preprocessing/image stacking/classification processing chain (Fig. 1), and finally a set of Python modules and functions for integrating into existing scripts, building new applications and training new machine learning models.

2.1. Software design choices

Pyeo functions have been written such that users can put together their own processing chains by copying and pasting functionalities as required. Most functions take paths to geospatial objects (raster or vector data) in the file system instead of passing data between them internally. Similarly, most geoprocessing functions take an out-path argument that writes the final product to a raster file. Though this approach causes more data read/write operations, it is more intuitive for users who are more experienced in Graphical User Interface (GUI) Geographic Information Systems (GIS) software than Python, and this permits straightforward examination of intermediate raster products at each stage of processing for the purpose of quality-checking and debugging own processing chains.

To reduce memory overhead when working with large rasters, Pyeo takes two approaches. Where possible, a raster that is opened for processing will be memory-mapped using GetVirtualMemArray() over the entire raster. This permits numpy-accelerated vectorised operations by band, row or column depending on context, removing the need to calculate offsets - the virtual mapping keeps memory use to the minimum needed for a given process. In the instance that Pyeo is being run on an OS that does not support the mmap system call (such as Windows), there is an automatic patch in place that replaces gdal.GetVirtualMemArray() with an equivalent using numpy's mmap class. Pyeo also implements the design ideal that a large object (such as a raster) should not exist past its required context, writing to the file system at the end of each major function call as described above. In the current version, Pyeo does not implement tiling at memory level; very large datasets should be tiled at the file system level prior to processing.

The software package organization is described in Appendix 1.

2.2. Data acquisition

At present, Pyeo has a fully mature module for the acquisition of Sentinel-2 data from the Copernicus Open Access Hub and the Synergise-provided AWS buckets on Sentinel Hub. There are also modules in progress for downloading data from Landsat and Planet; however, these are still in the prototype phase.

2.2.1. Sentinel 2

For all processing, Pyeo takes the product catalogue supplied by the Copernicus Open Access Hub (https://scihub.copernicus.eu/) as the canonical list of available image acquisitions. Queries are performed against this list using user-supplied credentials, and can be filtered by date range, area of interest (AOI), maximum cloud cover (according to the Hub) and processing level (L1C or L2A in the case of Sentinel-2). Once acquired, a list of images can be further filtered before being passed to a downloader.

The available downloaders are the following:

- Copernicus Open Access Hub This is the default option for downloading Sentinel-2 images. Images are downloaded in.zip format, and then automatically unzipped. Users are required by the Hub to register with a username and password before downloading, and there is a limit to no more than two concurrent downloads per username at a time. The Copernicus Open Access Hub is entirely free to access, but imagery is moved to the Long-Term Archive after a certain period of time, where data quota restrictions exist for users.
- Amazon Web Services (AWS) via Sentinel HubSentinel data are also publicly hosted on Amazon Web Services on Sentinel Hub. This storage is provided by Sinergise, and is normally updated a few hours after new products are made available. There is a small charge associated with downloading this data. To access the AWS repository, users are required to register for an Amazon Web Services account (including providing payment details) and obtain an API key for that account (see https://aws.amazon.com/s3/pricing/for pricing details). There is no limit to the concurrent downloads for the AWS bucket.

2.2.2. Landsat

All Landsat images (Wulder et al., 2019) are at present downloaded from the United States Geological Survey.

```
model = ens.ExtraTreesClassifier(criterion="gini",
 max_features=0.55, min_samples_leaf=2, min_samples_split=16,
 n_estimators=100, class_weight='balanced')
```

2.3. Data preparation

The objective of the pre-processing step is to extract the values of interest from downloaded data and convert them into a uniform internal format for comparison. For Pyeo, the internal format for raster processing is consistent with the GDAL convention; a single multiple-band array of order [bands, y, x], with associated affine geotransform, coordinate system and resolution; the latter two of these specified by the user.

The Sentinel-2 pre-processingbegins with the user specifying the spectral bands of interest and the desired spatial resolution of the raster layers.Sen2Cor atmospheric correction is applied to the L1C imagery to estimate the L2A bottom-of-atmosphere reflectance. If available, the bands of interest are retrieved from the available rasters at native spatial resolution (if the user has requested 10 m RGB values, for instance). If a band of interest is not available at the requested resolution, the closest available resolution of raster will be resampled to the finer spatial resolution, with nearest-neighbour resampling as the default method. All selected bands are then stacked into a single raster containing all selected bands from both acquisition dates. A cloud mask is then produced to exclude cloudy pixels from the classification and change detection (section 2.4.1). The stacked raster for change detection is reprojected from its native map projection (typically the appropriate UTM projection for that Sentinel-2 granule) to the desired output map projection for the chosen application. All interim products are stored as temporary geotiff files and cleaned up after processing. All resampling and reprojection is performed using GDAL. Image stacking is carried out using the Pyeo function stack_images() (Roberts et al., 2020). There are further functions for clipping raster files to specific areas-of-interest.

2.3.1. Processing steps

Cloud masking: If additional cloud-masking is required, Pyeo provides tools for producing cloud-masks from the Sentinel-2 scene classification layer (SCL), the cloud probability layer or the fmask algorithm (Zhu and Woodcock, 2014). The default cloud mask is a conservative combination of the thematic layer and fmask, with a user-definable buffering parameter. It is also recommended that users of Pyeo include a cloud class in the classification of their maps, or generate an intermediate cloud-trained model.

Image compositing: If needed, Pyeo provides functions for compositing multiple images for large area coverage, comparison and base-layer production.

Mosaicking: Pyeo allows mosaicking of multiple images if the AOI intersects with several image footprints if large area monitoring is required. This will increase computing time though, so it is recommended to adopt a tile-based processing strategy.

Band stacking: Pyeo contains functions that permit multiple raster layers to be stacked into a single file, e.g. image bands from subsequent acquisition dates. Unlike the similar function in GDAL, the Pyeo stacking routine can handle images with a different band order. In the case that images to be stacked do not overlap perfectly, the user can specify whether the intersection or union of all provided images is used; in the case of the union, the gaps are filled with a user-specified no-data value, defaulting to 0. Once preprocessed, rasters are prepared for exposure to the classification module for either feature extraction or classification.

Masked composite production: A feature designed for cloud-free compositing, this function produces a composite of last-seen-unmasked pixels from a list of images with associated masks, with an optional secondary raster of the date the last pixel was seen.

2.3.2. Feature extraction

The purpose of this processing is to produce a set of labelled training data that can be processed by Scikit-Learn and similar libraries. The feature extraction process requires two components: a Pyeo-preprocessed raster and a set of non-overlapping labelled training polygons. It also requires the user to indicate a training attribute.

The feature extraction process is as follows:

1. Reproject the polygons to match the Pyeo-processed raster.
2. Rasterize the polygons. Any pixel that corresponds to a polygon will take the value of that polygon's training attribute; every other pixel will be set to zero.
3. Stack the training label raster on top of the Pyeo-preprocessed raster, producing a raster of band ordering, e.g. [label, pixel_value_band0, pixel_value_band1, …].
4. Extract each pixel that does not have a label of zero to a table.

The table can then be passed directly to the model creation functions, cast to a dataframe or dumped to a.csv for further inspection and processing.

2.3.3. Model creation

Pyeo can natively create trained models from using a default set of parameters; a Scikit-Learn ExtraTreesClassifier with the following parameters:

This model was arrived at using the Tpot library (Le et al., 2020) for a change classification dataset; other papers have also recommended random forest based methods for image classification (e.g. Balzter et al., 2015; Breiman, 2001; Gibson et al., 2020), indicating that this is an appropriate standard model for image classification work.

Pyeo permits both the dumping of data to the Scikit-Learn accepted label-feature format and the loading of trained models. This allows other modules to create and tune classification models for use with Pyeo.

2.3.4. Classification

Models in Pyeo are transferred using the Python native joblib module (Joblib Development Team, 2020). A Pyeo-compatible model is a pickled instantiation of a class that implements at least a.predict() method that takes a GDAL-compliant array (shape=(bands, y, x)) as its only augment. If probabilities are required, it may also optionally implement a.predict_proba() method and an.n_classes_ attribute. Model outputs from sklearn follow this schema, but it would be possible for a user to implement their own classifier (based on static thresholding, for example) and pass that to Pyeo for use. Pyeo presently only implements pixel-by-pixel raster classification.

3. Algorithm and implementation

After the stacked band rasters from two satellite image acquisition dates have been processed as described in section 2, the core change detection algorithm can be run by Pyeo. Because Pyeo integrated Scikit-Learn, all available machine learning models from that library can in

principle be applied. We have used a Random Forest and an Extra Trees Classifier, which tends to be computationally more efficient than a Random Forest and is based on a similar approach. In the Extra Trees Classifier, each decision tree will be built using all available training data. To create a new node in the decision tree, the best split is determined by searching a subset of randomly selected features.

Pyeo reads in the spectral reflectance or bottom-of-atmosphere image bands from the chosen satellite mission from a 'before' and 'after' acquisition date. The bands are stacked in a single raster file and the change classes between the two dates are determined with the Extra Trees Classifier. Optionally, a probability layer can be created that gives the confidence level in the class of each pixel.

Algorithm 1 below shows a simple change detection application of Pyeo, assuming a set of training data exists, and the folders s2_l1, s2_l2, preprocessed and classified have been created.

Algorithm 1. Simple image classification with Pyeo.

```
from pyeo import raster_manipulation as ras

from pyeo import queries_and_downloads as dl

from pyeo import classification as cls


# train_model.py

cls.extract_features_to_csv("training_raster.tif",

                            "training_shape.shp",

                            "features.csv")

cls.create_model_from_signatures("features.csv", "model.pkl")




# classify_area.py

username = "scihub_user"

password = "scihub_pass"

conf = {'sent_2':{'user':username, 'pass':password}}

data = dl.check_for_s2_data_by_date("aoi.shp",

                                    "20200101",

                                    "20200201",

                                    "conf")

dl.download_s2_data(data, "s2_l1", "s2_l2", username, password)

ras.preprocess_sen2_images("s2_l2", "preprocessed", "s2_l1")

cls.classify_directory("preprocessed",

                       "model.pkl",

                       "classified",

                       apply_mask=True)
```

This is a minimal working example of Pyeo. In the case of Sentinel-2, it will download L1C and where available L2A Sentinel-2 granules over the specified AOI, preprocess them to the default 10 m bands as specified in the 'data preparation' section and classify them using the default Pyeo Extra Trees Classifier model. This script can be expanded and the interim products inspected as desired; it is also a very small modification to replace the dates with a derived range based on the current date and make a simple monitoring system.

The applicability of Pyeo in its current version is primarily aimed at operational ongoing change detection between land cover or forest types from satellite images that are acquired regularly. However, the software can be adapted to any other application that requires automated image classification with a trained machine learning model. Examples of other applications in future may include desertification, urbanization and flood extent monitoring.

The strengths of Pyeo are that it combines the machine learning functionality from Scikit-Learn with the image and raster processing

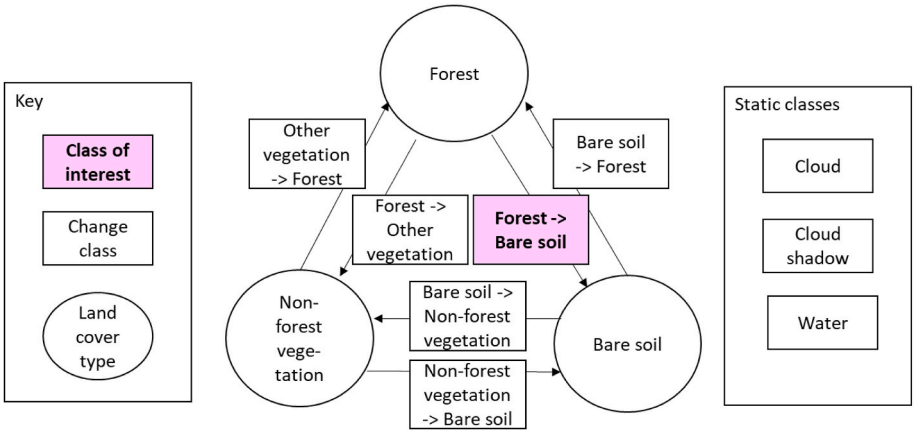

Fig. 2. Schematic describing the key definitions of the classes of interest including the forest - > bare soil class between two subsequent image acquisitions, the static classes used to mask out cloud, cloud shadow and water bodies, and the triangular diagram of land cover class transitions between forest canopies, non-forest other vegetation and bare soil. The change classes are defined as transitions between one image and the next (square boxes), while the land cover types are shown as circles.

capabilities of GDAL and atmospheric correction software such as Sen2Cor for this purpose and that it automates the updating of the change maps all the way from searching for new images down to producing the machine leaning classifications. Limitations currently include the need for visual quality assurance of the change detections, the radiometric properties of the rolling image composite, which can become very heterogeneous over time as new pixel values are replacing old ones due to seasonality of vegetation in certain ecosystems, and residual cloud contamination of images after the cloud-masking stage.

4. Application and results

4.1. Introduction and context

The cooperation between the University of Leicester and the REDD + Round Table of forestry stakeholders in Kenya began in 2014 through a Global Prospects Fund grant by the University of Leicester, followed in 2016 by the project "REDD + Monitoring Services with Satellite Earth Observation - Community Forest Monitoring Pilot" funded by the National Environmental Research Council (NERC). The UK Space Agency project "Forests 2020" funded the operationalisation of the prototype into a fully useable processing chain. The Government of Kenya wanted to establish a more robust forest monitoring system and build more capacity for using new Earth Observation data. It is estimated that Kenya's economy is losing US$68, 000, 000 annually from deforestation. Kenya has enshrined an ambitious afforestation and reforestation goal in law that aims to increase its forest cover from 6% to 10% of its land area by 2030 (Kenya Ministry of Environment and Natural Resources, 2016). Through a series of stakeholder workshops with the REDD + round table in Nairobi, we co-developed the system based on the user needs of the different organisations involved with forest monitoring, including representatives of government agencies, ministries, community forestry associations and non-governmental organisations.

In 2020, Pyeo was installed on the computing infrastructure at the Kenya Forest Service, with the aim of enabling the institution to create new monitoring applications over forest areas of concern. A capacity building event trained local staff in using, maintaining and developing the system and it now runs on the in-house computing systems at the Kenya Forest Service. Deforestation alerts delivered from this system are being used to inform the actions of forest rangers in protected forest areas.

This section describes the application of Pyeo in the Kwale Protected Forest Area, where the Sentinel-2 forest cover loss information is used in conjunction with user-submitted mobile phone alerts to inform the rangers of the Kenya Forest Service of any disturbances of the forest cover. The deforestation alerts and the Sentinel-2 forest cover loss detections are used together in a GIS database. Pyeo is used to create the models, construct the initial cloud-free base layer and carry out the automated downloads, preprocessing and classifications (Fig. 1).

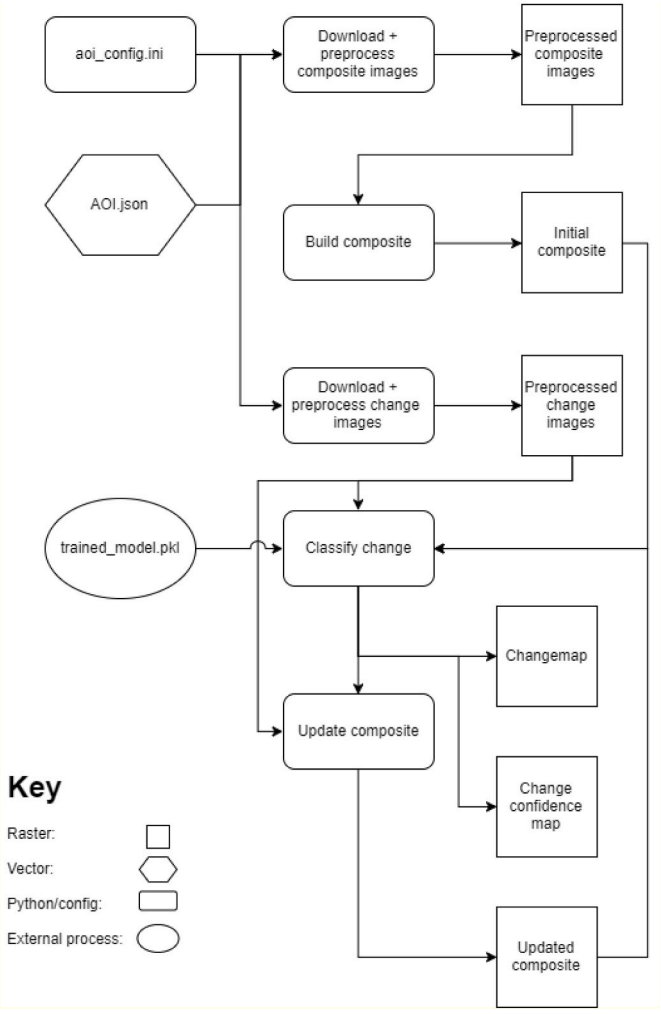

Fig. 3. Raster and vector data flow through the processing chain in the application of Pyeo for forest cover change detection at the Kenya Forest Service (KFS).

4.2. Machine learning model

The Extra Trees Classifier models used in this project were created and trained by the technical staff in the Kenya Forest Service. Unlike in

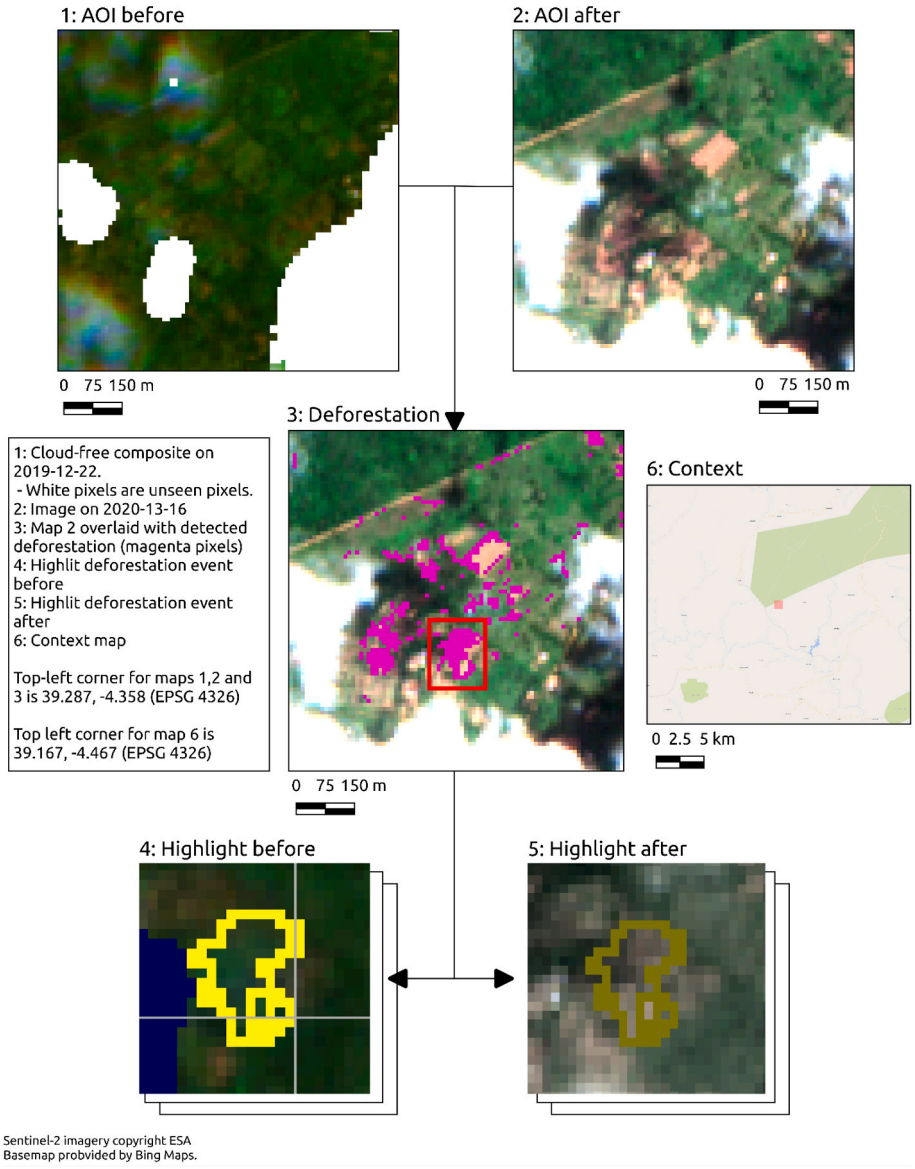

Fig. 4. Illustration of the step-by-step image processing flow of a forest change detection from Sentinel-2 imagery in the Shimba Hills Forest, Kwale County, Kenya.

many other applications, the classification identifies change classes between land cover types rather than the land cover types themselves (Fig. 2). As forest cover change is being identified, the model is trained on 8 features, which are the 4 spectral bands in red, green, blue and near-infrared from Sentinel-2 at 10 m resolution from two stacked images acquired on different dates between which the change is detected. As image choice is important, the images are downloaded, quality-checked and stacked by an interpreter at this stage. Sentinel-2 images are selected that show appropriate examples of deforestation, which is a relatively rare event despite its aggregated importance at national scale.

The model used in this project trains 12 change classes using 8 features: 9 classes are derived from the transitions between three basic land cover types, and the remaining 3 classes are cloud, cloud shadow and water. When a new area is ingested into the KFS system for further monitoring, KFS staff download appropriate images and create a new model to cover that area. For this project, the default model detailed in the modelling section was sufficient during development with KFS.

4.3. Baseline image composite creation

Images of forested areas of Kenya often have cloud cover in excess of 70%, especially in the wet season (April to June). For an effective change detection, a baseline image composite layer of cloud-free pixels is required. On instantiation of new detection areas, an appropriate date range is supplied and the function pyeo.raster_manipulation.composite_images_with_mask is used along with the automatically generated cloud mask to create a cloud-free or nearly cloud-free baseline image composite layer of the last seen cloud-free pixels at a given date. This is used as the initial raster for the change detection.

4.4. NRT change detection

Once an initial machine learning model had been created, a simple interface creates a batch script to call rolling_s2_composite.py. When it is executed, this will perform the processing chain shown in Fig. 3, downloading any recently acquired Sentinel-2 images since last change detection was carried out. Its outputs are a classified change map with the 12 classes, a corresponding probability map and an updated cloud-free image composite in which the latest cloud-free pixels are replacing the previous pixels in the composite. During this process, a forest mask based on the SLEEK land cover map for the year 2016 (Kenduiywo et al., 2020) is used to filter out any non-forest locations. Fig. 4

Fig. 5. Near-real-time forest cover loss alert for dissemination to the user, e.g. a forest ranger, containing an interactive summary display for use on any Google Earth-based platform. Each forest cover change polygon has a unique ID.

illustrates the individual processing steps with real examples. A second program then takes the Pyeo-produced change map, extracts and polygonises any deforestation area, produces images of areas-of-interest, creates a KML file (Fig. 5) and distributes this to subscribed emails.

When the Pyeo forest cover loss alerts are received by the Forest Information Centre of the Kenya Forest Service in Nairobi, an interpreter checks the quality of the alerts for any erroneous detections. These can occur because of residual cloud, cloud shadow or atmospheric haze effects in the Sentinel-2 images. Any alerts that pass the visual quality-checking are transmitted to the forest rangers in the protected forest area in Kwale for further investigation. Feedback from the rangers has shown that the inclusion of satellite-based deforestation detections adds value to their operations in the field. In particular, they are sometimes alerted to forest cover loss in areas that are hard to reach or that are not regularly being inspected. Forest cover loss polygons that pass the visual quality assurance can be used for updating the national forest inventory GIS database.

5. Conclusions

This paper describes the architecture of Pyeo, a Python package for creating satellite data processing chains for automated change detections with machine learning algorithms. It introduces an example of a simple processing chain and demonstrates Pyeo's present use in an operational environment in the Kenya Forest Service. As part of the Forests 2020 project, Pyeo has also been introduced to other countries and is presently being used to manage Sentinel-2 downloads for monitoring deforestation-free supply chains in the Ghanaian cocoa industry. Its heritage goes back to the Earth and Sea Observing System (EASOS) funded by the UK Space Agency.

Though the majority of platform-specific processing tools in Pyeo are oriented towards Sentinel-2, tools for downloading Landsat and Planet images exist in the codebase; in addition, any raster that has been appropriately preprocessed can be passed to the generic raster functions and machine learning sections. It is intended that further collaborations both inside and outside academia will increase the feature set of Pyeo to cover more platforms and further image classification techniques, permitting automated Earth observation techniques to be applied to ever widening fields.

Pyeo provides a scalable solution for change detection from regularly acquired satellite imagery. It can be run for a small area-of-interest defined by a shapefile extent, or it can be run for multiple granules or footprints, e.g. we are currently running it for over 100 Sentinel-2 satellite image granules over Mato Grosso in Brazilin theEuropean Space Agency project ForestMind, which is developing deforestation-free supply chain monitoring systems and it is the intention to incorporate the Pyeo functionality.

Authorship contribution statement

J.F. Roberts: software development and writing the first draft of the manuscript; R. Mwangi, F. Mukabi, J. Njui and K. Nzioka: implementation and operation of the software and interpretation of the change detections at the Kenya Forest Service, J.K. Ndambiri: team leader at Kenya Forest Service, conception of the application and stakeholder liaison; P. C.Bispo: satellite data analysis; F.D.B. Espirito-Santo: critical revision of the manuscript; Y. Gou: satellite data analysis; S.C.M. Johnson: critical revision of the manuscript; V. Louis: conception and application of the software; A.M. Pacheco-Pascagaza: satellite data analysis; P. Rodriguez-Veiga: satellite data analysis; K. Tansey: critical revision of the manuscript; C. Upton: leading a user study in Kenya on the uptake of the software; C. Robb: conception and

implementation of the first prototype; and H. Balzter: conception of the idea for the software and supervision of its implementation, team leader at the University of Leicester. All authors contributed to drafting and revising the manuscript.

Code availability section

Name of the code/library: Pyeo.

Contact: hb91@le.ac.uk; +44–116 252 3820.

Hardware requirements: Windows or Linux OS (Linux recommended).

Program language: Python.

Software required: Packages specified in the.yml file.

Program size: 10 MB (zipped).

The source codes are available for downloading at the link: https://github.com/clcr/pyeo.git.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Acknowledgments

This work was supported by the University of Leicester's Global Prospects Fund, the Natural Environment Research Council (NERC) Follow-on fund (NE/N017021/1) and a NERC Pathfinder grant (NE/M007839/1), the National Centre for Earth Observation (NCEO), the UK Space Agency International Partnership Fund projects Forests 2020 and EASOS and the European Space Agency project ForestMind. H.B. and P. R.V. were supported by the NERC National Centre for Earth Observation (PR140015).

Appendix 1. Pyeo software package organization

The modules in Pyeo are organised in several core scripts, including *Queries and Downloads* (functions for querying, filtering and downloading data from an API), *Raster Manipulation* (raster data processing including cloud masking using fmask, Sen2Cor and platform-specific processing functions), *Coordinate Manipulation* (functions for transforming spatial coordinates between map projections and pixel indices), File System Utilities (creation of file and folder structures, sorting lists of images, extracting data from platform-specific naming formats), *Classification* (image classification including model training with user-defined training data, array reshaping into Scikit-learn), and *Validation* (sampling points for validation to a specified confidence using the method by Olofsson et al., 2014).

Pyeo supports the collection of validation samples. The *Validation* script provides a function for producing sufficient sample points for validation to a specified confidence using the method detailed in Olofsson et al. (2014). The user provides a configuration file containing the expected user accuracy for each class, a minimum number of samples for any class that is sparse and a target standard error value.

These core modules have been used to build a number of application scripts:

Download_and_preproc_area.py downloads, atmospherically corrects, pre-processes, reprojects and stacks a collection of Sentinel-2 images for an area-of-interest.

Extract_signatures.py reads in a shapefile containing training polygons with class numbers and a matching raster file, extracts all corresponding class and radiometric pixel values into a labelled training. csv file for further analysis.

Create_model_from_signatures.py uses a.csv of pixel class labels and values to train a machine learning model, which is exported as a.pkl file for operational applications to new image data.

Composite_directory.py reads in a list of raster files that contain satellite image data acquired on specified dates and their corresponding cloud masks, and produces a cloud-free or nearly cloud-free image composite of the most recently seen pixels. It can optionally produce a layer of the acquisition dates of each pixel value in the composite. The composite can be used as a baseline to which a new image can be compared.

Simple_classification.py reads in a machine learning model from a. pkl file and a raster file, and produces a classified output image.

The EO-based geospatial element has been written with entirely open source Python and is adaptable to both desktop and cloud-based platforms. Its performance is highly platform-dependent.

References

Afzaal, H., Farooque, A.A., Schumann, A.W., Hussain, N., McKenzie-Gopsill, A., Esau, T., Abbas, F., Acharya, B., 2021. Detection of a potato disease (early blight) using artificial intelligence. Rem. Sens. 13, 411.

Balzter, H., Cole, B., Thiel, C., Schmullius, C., 2015. Mapping CORINE land cover from Sentinel-1A SAR and SRTM digital elevation model data using random forests. Rem. Sens. 7, 14876–14898. https://doi.org/10.3390/rs71114876.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32. https://doi.org/10.1023/A:1010933404324.

Bunting, P., Clewley, D., Lucas, R.M., Gillingham, S., 2014. The remote sensing and GIS software library (RSGISLib). Comput. Geosci. 62, 216–226.

Dutrieux, L., DeVries, B., 2014. bfastSpatial: Set of Utilities and Wrappers to Perform Change Detection on Satellite Image Time-Series. https://doi.org/10.5281/zenodo.49693.

Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. 15, 3133–3181.

GDAL/OGR contributors, 2021. GDAL/OGR Geospatial Data Abstraction Software Library. Open Source Geospatial Foundation.

Gibson, R., Danaher, T., Hehir, W., Collins, L., 2020. A remote sensing approach to mapping fire severity in south-eastern Australia using sentinel 2 and random forest. Rem. Sens. Environ. 240 https://doi.org/10.1016/j.rse.2020.111702.

Gillies, S., others, 2013. Rasterio: Geospatial Raster I/O for Python Programmers. Mapbox.

Hansen, M.C., Potapov, P.V., Moore, R., Hancher, M., Turubanova, S.A., Tyukavina, A., Thau, D., Stehman, S.V., Goetz, S.J., Loveland, T.R., Kommareddy, A., Egorov, A., Chini, L., Justice, C.O., Townshend, J.R.G., 2013. High-resolution global maps of 21st-century forest cover change. Science 342, 850–853. https://doi.org/10.1126/science.1244693.

Joblib Development Team, 2020. Joblib: Running Python Functions as Pipeline Jobs.

Jönsson, P., Eklundh, L., 2004. TIMESAT—a program for analyzing time-series of satellite sensor data. Comput. Geosci. 30, 833–845.

Kenduiywo, B.K., Mutua, F.N., Ngigi, T.G., Waithaka, E.H., 2020. Mapping mangrove forest using Landsat 8 to support estimation of land-based emissions in Kenya. Model. Earth Sys. Environ. 6, 1619–1632.

Kenya Ministry of Environment and Natural Resources, 2016. National Forest Programme of Kenya.

Ketkar, N., 2017. Introduction to pytorch. In: Deep Learning with Python. Springer, pp. 195–208.

Le, T.T., Fu, W., Moore, J.H., 2020. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. Bioinformatics 36, 250–256. https://doi.org/10.1093/bioinformatics/btz470.

Malenovskỳ, Z., Rott, H., Cihlar, J., Schaepman, M.E., García-Santos, G., Fernandes, R., Berger, M., 2012. Sentinels for science: potential of Sentinel-1,-2, and-3 missions for scientific observations of ocean, cryosphere, and land. Rem. Sens. Environ. 120, 91–101.

Moffette, F., Alix-Garcia, J., Shea, K., Pickens, A.H., 2021. The impact of near-real-time deforestation alerts across the tropics. Nat. Clim. Change 11, 172–178. https://doi.org/10.1038/s41558-020-00956-w.

Olofsson, P., Foody, G.M., Herold, M., Stehman, S.V., Woodcock, C.E., Wulder, M.A., 2014. Good practices for estimating area and assessing accuracy of land change. Rem. Sens. Environ. 148, 42–57.

Pacheco-Pascagaza, A.M., Gou, Y., Louis, V., Roberts, J.F., Rodríguez-Veiga, P., da Conceição Bispo, P., Espírito-Santo, F.D.B., Robb, C., Upton, C., Galindo, G., Cabrera, E., Pachón Cendales, I.P., Castillo Santiago, M.A., Carrillo Negrete, O., Meneses, C., Iñiguez, M., Balzter, H., 2022. Near real-time change detection system using sentinel-2 and machine learning: a test for Mexican and Colombian forests. Rem. Sens. 14 https://doi.org/10.3390/rs14030707.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Roberts, J., Balzter, H., Gou, Y., Louis, V., Robb, C., 2020. Pyeo: Automated Satellite Imagery Processing. Zenodo. https://doi.org/10.5281/zenodo.3689674.

Verbesselt, J., Hyndman, R., Newnham, G., Culvenor, D., 2010. Detecting trend and seasonal changes in satellite image time series. Rem. Sens. Environ. 114, 106–115. https://doi.org/10.1016/j.rse.2009.08.014.

others Wulder, M.A., Loveland, T.R., Roy, D.P., Crawford, C.J., Masek, J.G., Woodcock, C.E., Allen, R.G., Anderson, M.C., Belward, A.S., Cohen, W.B., 2019. Current status of Landsat program, science, and applications. Rem. Sens. Environ. 225, 127–147.

Zhu, Z., Woodcock, C.E., 2014. Automated cloud, cloud shadow, and snow detection in multitemporal Landsat data: an algorithm designed specifically for monitoring land cover change. Rem. Sens. Environ. 152, 217–234.