→ THE ESA EARTH OBSERVATION Φ-WEEK

EO Open Science and FutureEO

12–16 November 2018 | ESA–ESRIN | Frascati (Rome), Italy

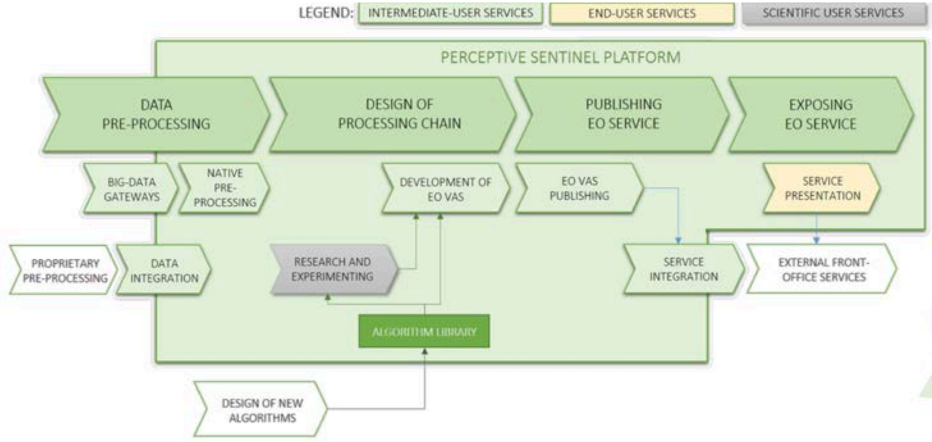
Perceptive Sentinel

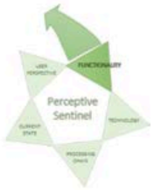
Dr. Anže Zupanc

16/11/2018

European Union's Horizon 2020 Research and Innovation Programme under the Grant Agreement 776115

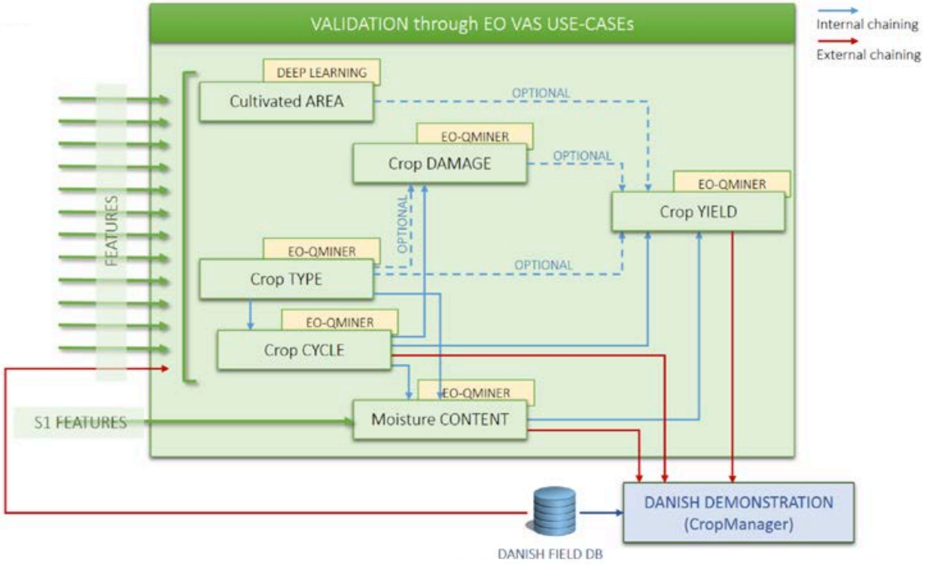
Perceptive Sentinel Platform





- **Intermediate Earth observation (EO)** service providing
  - ***modelling*** and ***publishing*** capabilities for
  - ***design***, ***exposure*** and ***exploitation*** of EO-processing chains for
  - ***forecasting***, ***monitoring*** and ***historical analysis*** based on
  - ***multi-temporal*** and ***multi-spectral*** EO and non-EO data ***modelling***

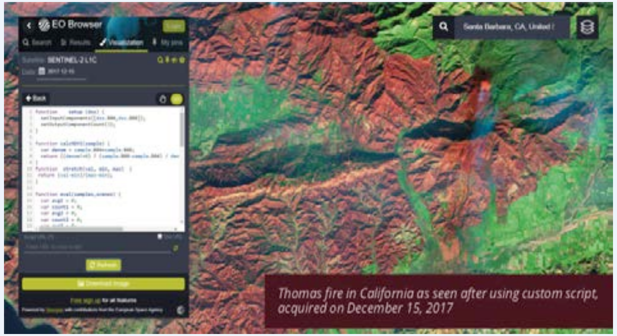
- Partners
  - Sinergise, Slovenia
  - GeoVille, Austria
  - Magellium, France
  - Jožef Stefan Institute, Slovenia
  - Agricultural Institute Slovenia, Slovenia
  - Seges, Denmark



- Demonstration Use-cases
  - Cultivated Area
  - Crop Type
  - Crop Damage
  - Cultivated Area
  - Crop Type
  - Crop Damage

Building on top of existing tools and services

Sentinel Hub Services



APIs for advanced feature integration

helping experts and software developers build new Earth observation services using our REST interfaces and open-source libraries.

EXPLORE THE WORLD WITH SENTINEL HUB

Sentinel Playground

Explore satellite imagery in an easy-to use web aplication. Select from a variety of products or create your own.
http://sentinel-hub.com/explore/sentinel-playground

EO Browser

Query satellite data, inspect results in various colour composites in full resolution, and download for offline analysis.
http://sentinel-hub.com/explore/eobrowser

DEVELOP WITH SENTINEL HUB

Easy integration with desktop GIS and web applications

such as ArcGIS, QGIS, MapBox, Carto, Google Maps, Leaflet, OpenLayers...

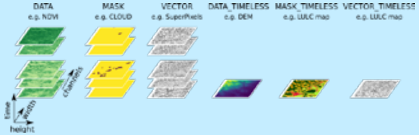
Standard **web services** – WMS, WMTS, WCS and WFS

fully configurable and customizable to your needs. You can easily integrate the services into existing products or use them to develop new applications. http://sentinel-hub.com/develop/capabilities/wms

- WMS/WCS Requests
- Standard and Custom Layers
- Multi-temporal Requests
- Cloud Coverage Filtering
- Download to / Read from Disk

sentinelhub-py – Sentinel Hub Python SDK

EOPatch



EOTask

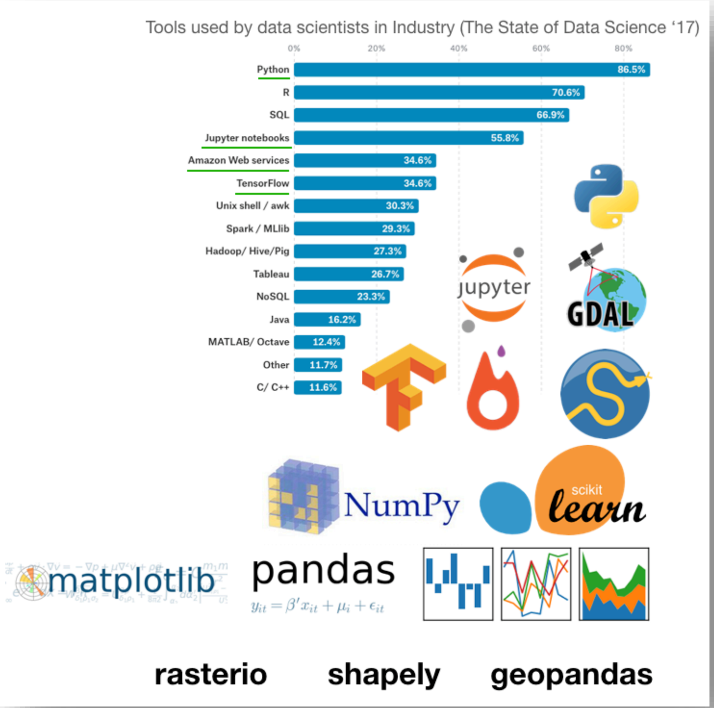
EOWorkflow build complex value-extraction pipelines

eo-learn extract value from satellite imagery

eo-learn library

eo-learn

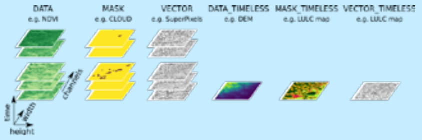
- *collection of modular Python sub-packages that allow processing of spatio-temporal data to prototype, build, and automate large scale EO workflows*
- *operates on Area of Interest (AOI) of any size*
- *acts as a bridge between EO and Python ecosystem for data science and machine learning*
- it's open source (MIT License)



eo-learn enables experts and non-experts alike to explore, discover, and share value from vast amounts of satellite imagery

eo-learn building blocks

EOPatch



- a common data-object for *spatio-temporal* EO and non-EO data, and their derivatives (numpy arrays, shapely polygons)
- sensor agnostic (optical, radar, ...)

EOTask

- a single, well-defined action being performed on existing **EOPatch**(es)
- each **EOTask** takes an **EOPatch** as an input and returns a modified **EOPatch**
- little to no overhead in order to implement a new task

EOWorkflow build complex value-extraction pipelines

- a collection of **EOTask**s that together represent an *EO-value-adding-processing* chain

EO-value-extracting processing chain development – Learn by doing it

- Start with a simple pipeline and build on top of it
  - Why aim for >97% accuracy, if 80% is already acceptable?
  - Why use L2A, if acceptable results are possible with L1C?

- Identify weakest links in the pipeline
  - Why improve task X, if precision is driven by task Y?

- Make qualitative instead of quantitative conclusions
  - Systematically monitor how various improvements of (or parts of) the pipeline impact the evaluation metric

Goal of eo-learn is to help you answer above questions and in the same time let you develop and/or use the model of your choice (domain knowledge, rule based, *classical* ML, DNN, AI, ...)

Example: Land Use Land Cover Classification with eo-learn

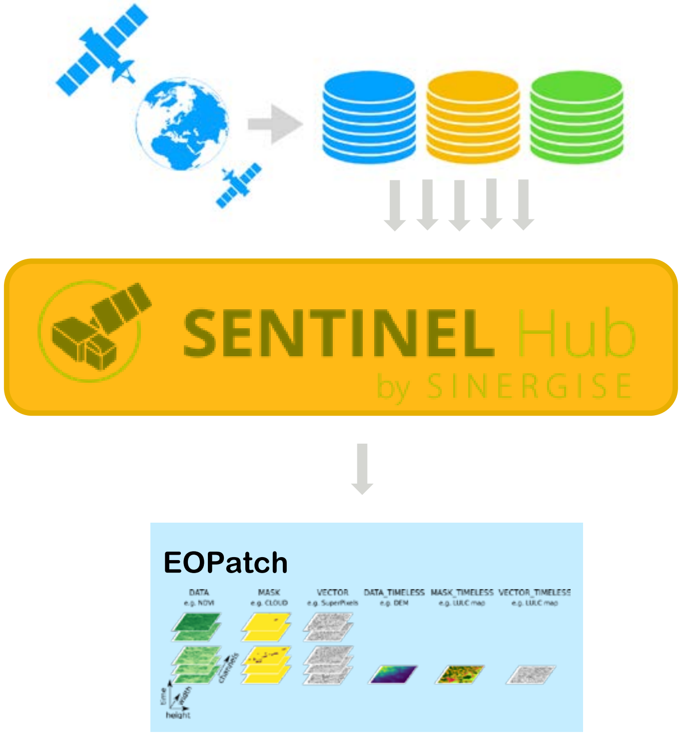
1-year Sentinel-2 L1C time-series → 10 class Land Use Land Cover map

End-to-end Land Use Land Cover Classification Jupyter notebook available in eo-learn's GitHub repository (notebook with description)

Example: Land Use Land Cover Classification with eo-learn

Processing pipeline in a nutshell:

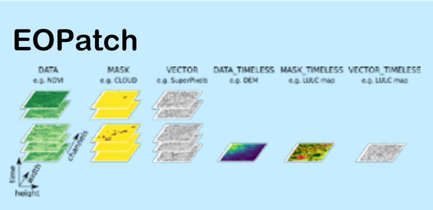
1. Define and get input data
    - 1-year S2 L1C t-series



Example: Land Use Land Cover Classification with eo-learn

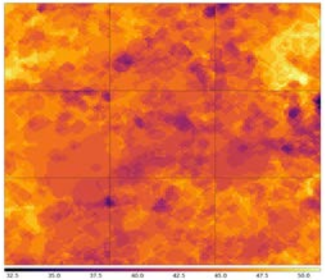
Processing pipeline in a nutshell:

1. Define and get input data
   - 1-year S2 L1C t-series
2. Perform cloud masking and filtering
   - s2cloudless
3. Calculate additional features, like NDVI, NDWI
4. Add ground truth data
   - vector or raster

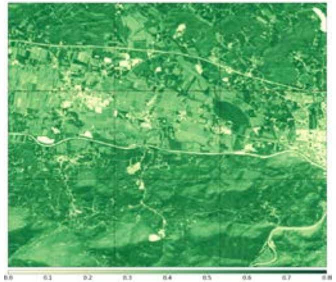


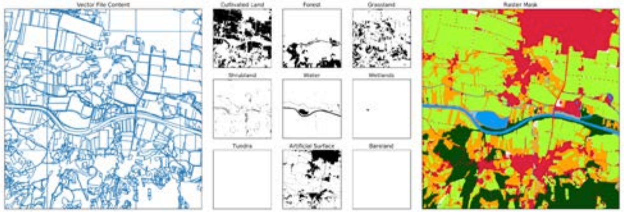
RGB at specific date

valid observations in 2017



mean NDVI[valid] in 2017

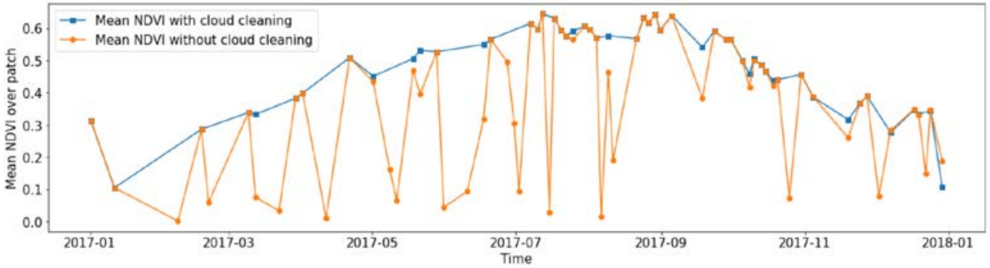


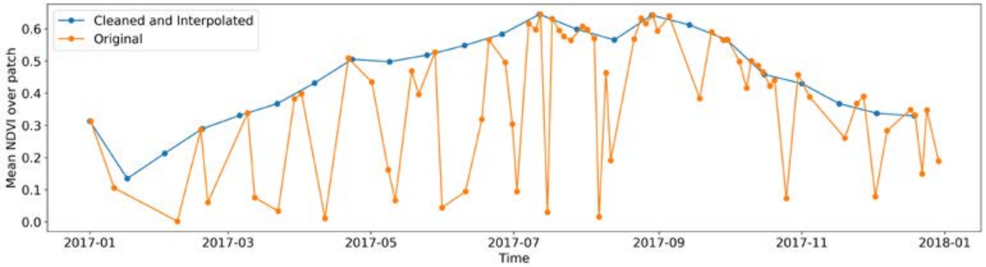


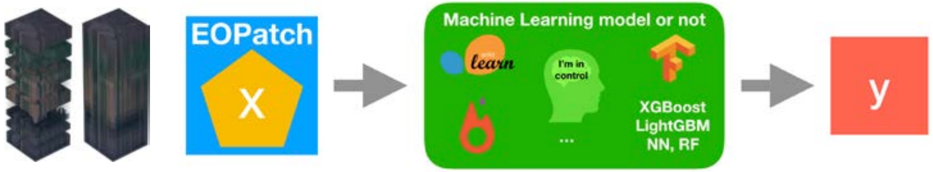
Example: Land Use Land Cover Classification with eo-learn

Processing pipeline in a nutshell:

1. Define and get input data
   - 1-year S2 L1C t-series
2. Perform cloud masking and filtering
   - s2cloudless
3. Calculate additional features, like NDVI, NDWI
4. Add ground truth data
   - vector or raster
5. Interpolate and temporally resample
   - get fixed number of features for every pixel
6. Spatially sample to create a train/val datasets
7. Train the model
   - use any model, library, outside eo-learn



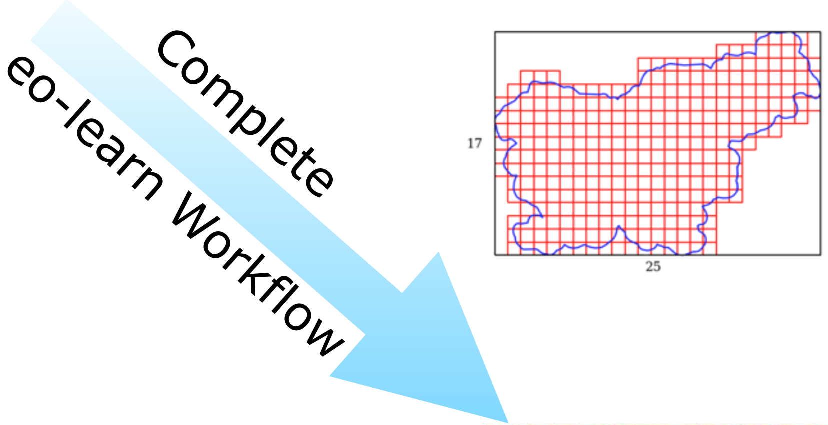


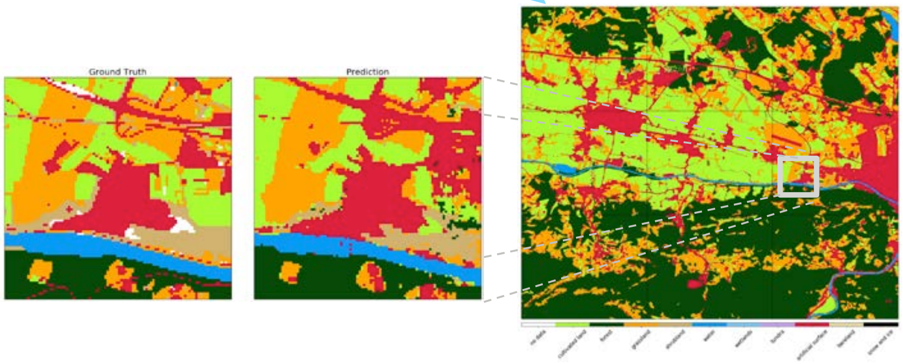


Example: Land Use Land Cover Classification with eo-learn

Processing pipeline in a nutshell:

1. Define and get input data
   - 1-year S2 L1C t-series
2. Perform cloud masking and filtering
   - s2cloudless
3. Calculate additional features, like NDVI, NDWI
4. Add ground truth data
   - vector or raster
5. Interpolate and temporally resample
   - get fixed number of features for every pixel
6. Spatially sample to create a train/val datasets
7. Train the model
   - use any model, library, outside eo-learn
8. Include the model and make predictions





Example: Land Use Land Cover Classification with eo-learn

```python
custom_script = 'return [B02, B03, B04, B08, B11, B12];'
add_data = S2L1CWCSInput(
    layer='BANDS-S2-L1C',
    feature=(FeatureType.DATA, 'BANDS'), # save under name 'BANDS'
    custom_url_params={CustomUrlParam.EVALSCRIPT: custom_script}, # custom url for 6 specific bands
    resx='10m', # resolution x
    resy='10m', # resolution y
    maxcc=0.8, # maximum allowed cloud cover of original ESA tiles
)

# TASK FOR CLOUD INFO
# cloud detection is performed at 160m resolution
# and the resulting cloud probability map and mask
# are scaled to EOPatch's resolution
cloud_classifier = get_s2_pixel_cloud_detector(average_over=2, dilation_size=1, all_bands=False)
add_clm = AddCloudMaskTask(cloud_classifier, 'BANDS-S2CLOUDLESS', cm_size_y='160m', cm_size_x='160m',
                           cmask_feature='CLM', # cloud mask name
                           cprobs_feature='CLP' # cloud prob. map name
                          )

# TASKS FOR CALCULATING NEW FEATURES
# NDVI: (B08 - B04)/(B08 + B04)
# NDWI: (B03 - B08)/(B03 + B08)
# NORM: sqrt(B02^2 + B03^2 + B04^2 + B08^2 + B11^2 + B12^2)
ndvi = NormalizedDifferenceIndex('NDVI', 'BANDS/3', 'BANDS/2')
ndwi = NormalizedDifferenceIndex('NDWI', 'BANDS/1', 'BANDS/3')
norm = EuclideanNorm('NORM','BANDS')

# TASK FOR VALID MASK
# validate pixels using SentinelHub's cloud detection mask and region of acquisition
add_sh_valmask = AddValidDataMaskTask(SentinelHubValidData(),
                                      'IS_VALID' # name of output mask
                                     )

# TASK FOR COUNTING VALID PIXELS
# count number of valid observations per pixel using valid data mask
count_val_sh = CountValid('IS_VALID', # name of existing mask
                          'VALID_COUNT' # name of output scalar
                         )

# TASK FOR SAVING TO OUTPUT (if needed)
path_out = './eopatches_small/' if use_smaller_patches else './eopatches_large/'
if not os.path.isdir(path_out) and save_choice:
    os.makedirs(path_out)
save = SaveToDisk(path_out, overwrite_permission=OverwritePermission.OVERWRITE_PATCH)
```

```python
# TASK TO LOAD EXISTING EOPATCHES
load_from_file = LoadFromDisk(path_out, lazy_loading = True)
load_from_memory = LoadFromMemory()

# TASK FOR CONCATENATION
concatenate = ConcatenateData('FEATURES', ['BANDS', 'NDVI', 'NDWI', 'NORM'])

# TASK FOR FILTERING OUT TOO CLOUDY SCENES
# keep frames with > 80 % valid coverage
valid_data_predicate = ValidDataFractionPredicate(0.8)
filter_task = SimpleFilterTask((FeatureType.MASK, 'IS_VALID'), valid_data_predicate)

# TASK FOR LINEAR INTERPOLATION
# linear interpolation of full time-series and date resampling
resampled_range = ('2017-01-01','2017-12-31',16)
linear_interp = LinearInterpolation(
    'FEATURES', # name of field to interpolate
    mask_feature = (FeatureType.MASK,'IS_VALID'), # mask to be used in interpolation
    resample_range=resampled_range, # set the resampling range
    bounds_error=False # extrapolate with NaN's
)

# TASKS FOR MOVING STUFF
move_features = MoveFeature({
    FeatureType.MASK_TIMELESS: {'LULC'},
    FeatureType.MASK: {'IS_VALID'}
})

# TASK FOR SPATIAL SAMPLING
# Uniformly sample about 100k pixels from patches
n_samples = int(4e4) if use_smaller_patches else int(1e5) # no. of pixels to sample
ref_labels = [0,1,2,3,4,5,6,7,8,9,10] # reference labels to take into account when sampling
disk_radius = 1 # size of erosion disk, applied before sampling
spatial_sampling = PointSamplingTask(
    n_samples=n_samples,
    ref_mask_feature='LULC',
    ref_labels=ref_labels,
    sample_features=[  # tag fields to sample
        (FeatureType.DATA, 'FEATURES'),
        (FeatureType.MASK, 'IS_VALID'),
        (FeatureType.MASK_TIMELESS, 'LULC')
    ],
    disk_radius=disk_radius)
```

Runs on your laptop too!

Complete example available at https://github.com/sentinel-hub/eo-learn

Conclusions and Plans

- Perceptive Sentinel is going to be an intermediate EO service for fast, efficient and easy ***design***, ***exposure*** and ***exploitation*** of EO-processing chains based on ***multi-temporal*** and ***multi-spectral*** EO and non-EO data
- Based on **eo-learn** Python library
  - ready to be used for development of project's use-cases
  - End-to-end Land Use Land Cover classification example published (expect more)
  - open source (use it, share it, contribute to it)

European Union's Horizon 2020 Research and Innovation Programme under the Grant Agreement 776115

More info

- https://eo-learn.readthedocs.io/
- https://medium.com/sentinel-hub/
- https://github.com/sentinel-hub
- http://www.perceptivesentinel.eu

European Union's Horizon 2020 Research and Innovation Programme under the Grant Agreement 776115