Documentation on using eCognition 10.1 server in CRIB

Draft Version 0.0.2

Authors: Peter Hofmann, Serkan Girgin

Document Description

This document describes the usage of the *eCognition* 10.1 server environment as it is installed at ITC's Geospatial Computing Platform (GCP) provided by CRIB using the Command Line Engine (CLE) of *eCognition*. The paper addresses *eCognition* users who are familiar with *eCognition* and its terminology and who intend to process larger amounts of remote sensing data (together with GIS and auxiliary data) in OBIA manner with *eCognition* at the ITC GCP.

Introduction

The so-called *eCognition suite* consists of several different components which can be connected in different ways in order to setup a client-server environment (for details please refer to the software documentation and the online product description at https://geospatial.trimble.com/what-is-ecognition). Most *eCognition* users working in the remote sensing domain are familiar with *eCognition Developer* which can be understood as the development platform for OBIA rule sets, usually being operated in a stand-alone manner, e.g., on a dedicated workstation or a laptop. *eCognition Developer* itself, too, can act as a client if embedded in such an environment or it can execute *eCognition Server* processes and algorithms locally to allow developers to develop for an *eCognition Server* environment.

When using the *eCognition Server* environment in CRIB, however, the connection between the Windows-based *eCognition Developer* client and the *eCognition Server* environment is limited. That is, a direct data exchange between both is not possible, as well as sending jobs and receiving results or the current states of sent jobs. Although you can send jobs, due to the limitations of the current version of eCognition data paths are not correctly translated between the machines. Consequently, as like all applications in CRIB, *eCognition* currently can only be used in your workspace (i.e. Docker container) that you can access by logging in to the platform. Additionally, eCognition can only be used with an Intel-based computer. Thus, you should start your CRIB computing unit with an Intel-based server (Fig. 1).

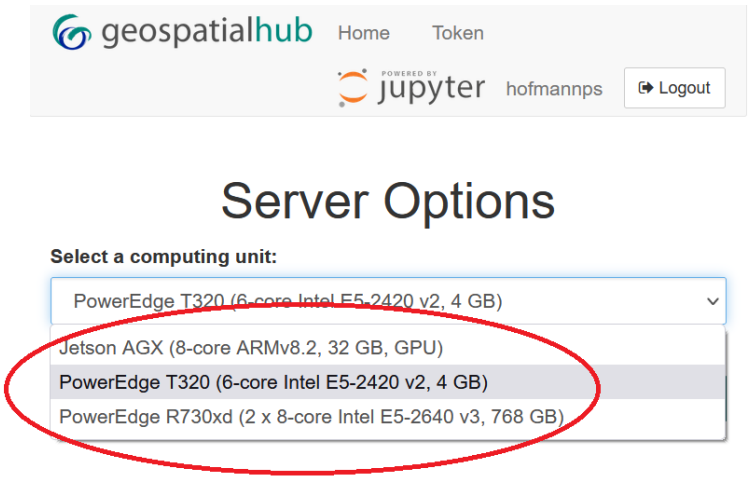

Fig. 1: Select an Intel-based server when working using eCognition on CRIB.

The CLE tools are installed in the folder `/opt/eCognition/cle`. You can use the terminal application available in the JupyterLab launcher or remote desktop to run the tools. Recently, five eCognition servers (DIAEngines) are available for processing, which can be increased to a maximum of ten in time.

eCognition tasks and results being processed on all available DIAEngines can be monitored as usual using the *eCognition JobScheduler* and its HTML user interface in the browser available on the platform (Firefox) entering the URL: http://ecognition-js:8184/ (Fig. 2). Note: the scheduler is not accessible from outside CRIB, thus the browser must be started from the remote desktop connection available in your workspace (Launcher > Remote Desktop, then select Applications > Accessories > Firefox Web Browser).

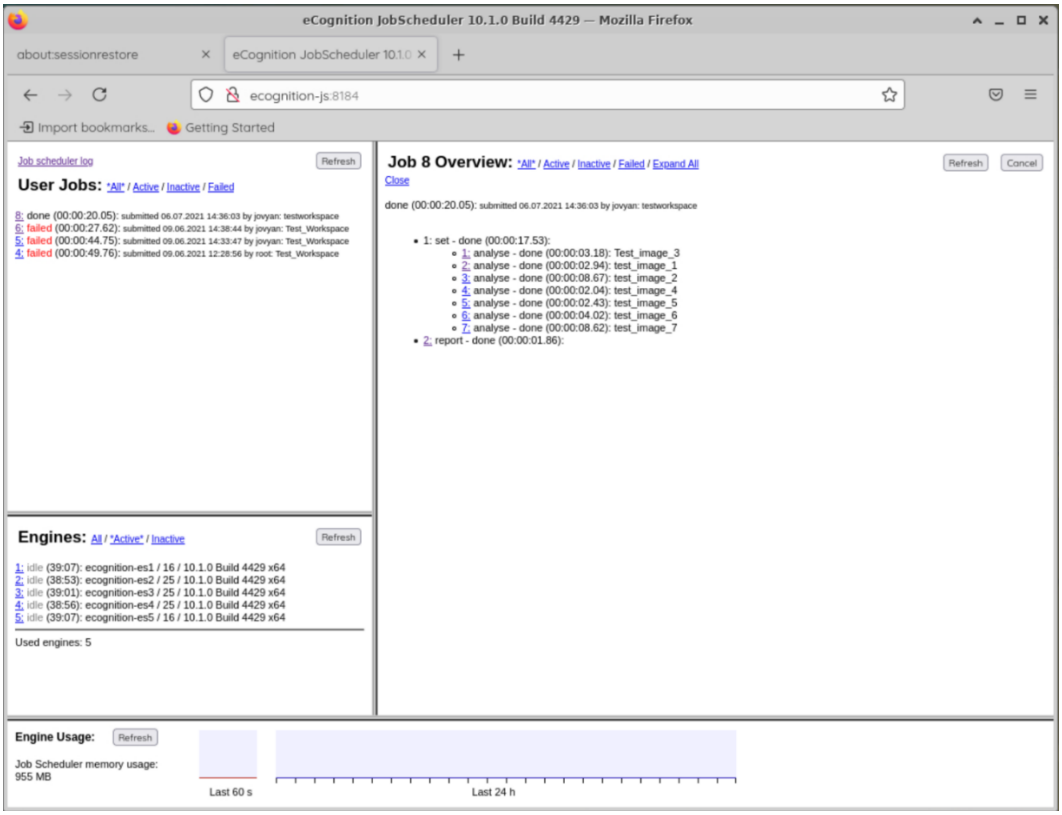

Fig. 2: eCognition Job Scheduler with available engines and active/inactive jobs displayed.

Prerequisites

The CLE is not eCognition Developer for Linux. It rather allows to process your data in different ways using a command line console in Linux (e.g. bash). This makes it possible to automate OBIA processes or to embed them in a processing chain, e.g. a 24/7 operating service.

To process data with the CLE you need to:

- Copy the data you want to process to a subfolder under `/data/shared/eCognition` on which you have full access.
- Copy the rule set(s) you wish to apply on the data into a subfolder under `/data/shared/eCognition` on which you have full access.
- Translate the rule set(s) from Windows to UNIX notation, i.e. change path names with backslash (`\`) to pathnames with slash (`/`).

You can use the upload file button available at the top of the JupyterLab file browser side bar to upload data files. To upload folders, you should create an archive (e.g. ZIP) of the folder, upload it to the platform, and then extract them. Please refer to the CRIB Support Center Knowledgebase on how to copy data from and to CRIB.

Translate the rule set from Windows notation to UNIX notation using the script `correct_paths.sh`. The script is available under `/opt/eCognition/cle`. It is self-explaining: just call the script together with the full name of the rule set to be translated, e.g.: `/opt/eCognition/cle/correct_paths.sh project.dcp`. The script will automatically create a backup file, e.g. `project.dcp.bak` and a translated *.dcp file with UNIX notation paths.

If you want to process multiple image files with the same rule set(s), organizing your data in an *eCognition Workspace* is recommendable. With the CLE you cannot process *eCognition Workspaces* you have created using a windows-based *eCognition* client. You must create them again in CRIB using the CLE. If you intend to process *eCognition Project* files (*.dpr) you should save them in your windows client separately and copy them together with the image files, the GIS files and auxiliary files used in the *eCognition Project*. Again: make sure the path names are relative and backslashes (`\`) are changed to slashes (`/`) or use the `correct_paths.sh` script. Accordingly, if you use a *Customized Import* (*.xml) file make sure the paths are accessible in CRIB and pathnames are set correctly in UNIX-manner.

Usage

The CLE offers three executable eCognition binaries. They can be executed by navigating to the folder `/opt/eCognition/cle` in the console window and entering:

`./DIACmdClient` or

`./DIACmdEngine` or

`./DIAMkWksp`

You can read more about their usage and their syntax by typing the according command without any further parameters, e.g. `./DIACmdClient` or `./DIACmdClient -help` or `./DIACmdClient -h`.

DIACmdClient

The `DIACmdClient` can only be used in conjunction with an *eCognition workspace* or a *Customized Import* (*.xml) file. That is, with multiple image files or tiles being organized accordingly. As like the windows-based *eCognition* clients it sends a job to the *eCognition JobScheduler* which then distributes the jobs to the *eCognition* servers.

DIACmdEngine

With the `DIACmdEngine` you can process image files (together with GIS and/or auxiliary files) directly. That is, they do not need to be organized in an *eCognition workspace*. Thus, you can either enter the files to be processed directly in the command line or you can create a *Customized Import* (*.xml) file referred to as `import-connector` in the syntax description. Again: make sure that path descriptions follow the UNIX standard. Note that when using `DIACmdEngine` you can set explicit output paths for the analysis results and the resulting project files(s) (*.dpr). `DIACmdEngine` processes images on the computing unit you are connected to, therefore it does not benefit from multiple (5-10) engine servers available on the platform. This may result in higher processing time.

DIAMkWksp

Use `DIAMkWksp` to create a workspace to be used by the `DIACmdClient`. `DIAMkWksp` can also be used to create templates for *eCognition* file management (i.e. data import and export). Additionally, an `import-connector` aka *Customized Import* (*.xml) can be created as a template which reflects the same file structure as the according *eCognition* workspace does. Note: in the syntax description DICOM format is mentioned. DICOM is usually not used in the remote sensing domain.

Typical workflow and examples

The typical workflow when working with *eCognition Developer* for Windows and the CRIB-based *eCognition* environment is as follows:

1. Loading image data and ancillary data (GIS data, point clouds, etc.) into *eCognition Developer* by either one of the following methods:
   1.1. Directly in the menu *File -> Load Image File …* or *File -> Import Multiple Scenes …*
   1.2. By creating an *eCognition Project File (*.dpr.): File -> New Project …*
   1.3. By creating an *eCognition Workspace (*.dpj): File -> New Workspace …*
   1.4. By using: *File -> Predefined Import …* (adds the imported scenes directly to the current workspace in the windows version)

1.5. By using *File -> Customized Import …* which can be used to define and store the way your data will be loaded in an *.xml file. The latter can be re-used in the CRIB-based *eCognition* environment, but path names must be adapted. The imported files will be automatically added to the currently open workspace in the windows version.

The (image) data can be a subset of the data you want to be processes by the *eCognition* server environment. That is, either a sample scene or a geographical subset.

2. Developing the rule set

Based on the loaded subset data you then start to develop your rule set, which you store as a *.dcp file. The rule set can contain *eCognition* server specific processes such as:

2.1. set rule set options: to control the usage of underlying hardware resources.
2.2. All workspace related processes (e.g. tiling and stitching).

Note: in order to test the rule set for an *eCognition* server environment you can simulate it in *eCognition Developer* by managing the local servers: Double-click on the left grey disk at the very lower right of the eCognition Developer window (Fig. 3):

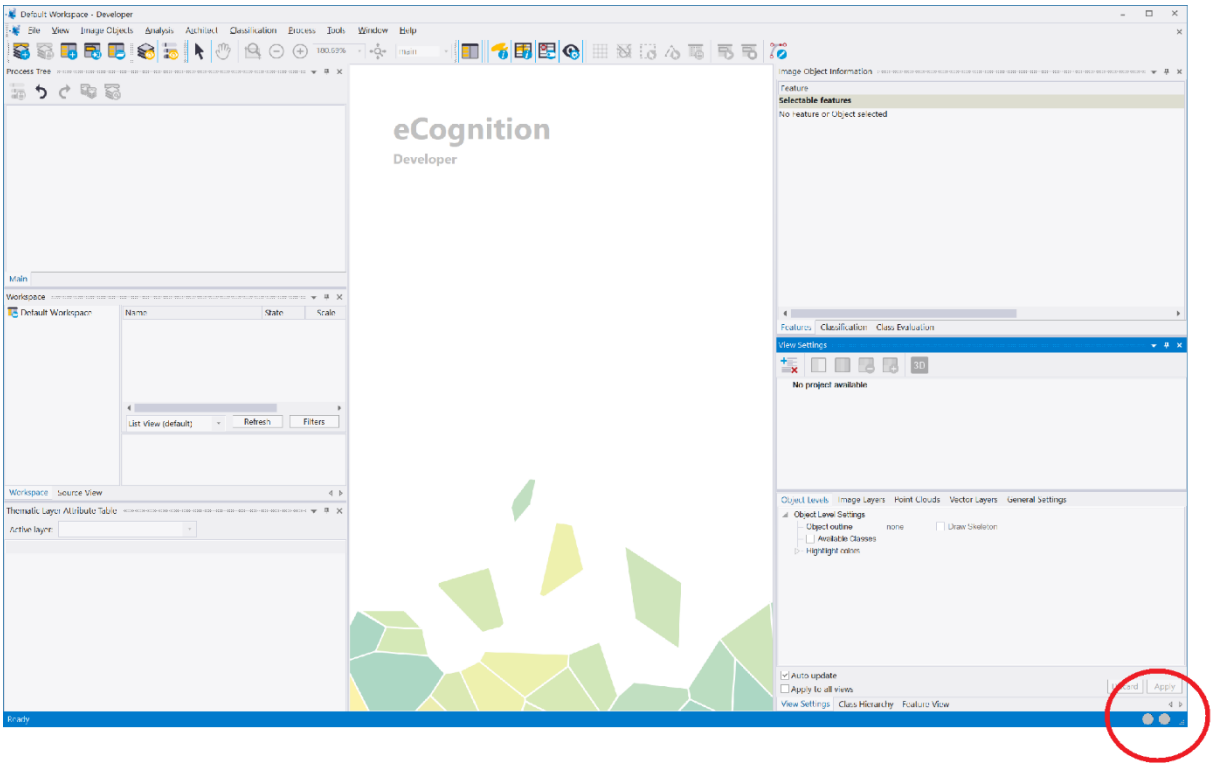

Fig. 3: Starting the dialog „Manage Local Servers" by double-clicking the left of the two grey disks.

Or select from the *Tools-menu: Tools -> Manage Local Servers …* . In the upcoming dialog "Enable" the *Number of Engines* you think you would need without extending the number of cores your CPU has (Fig. 4):

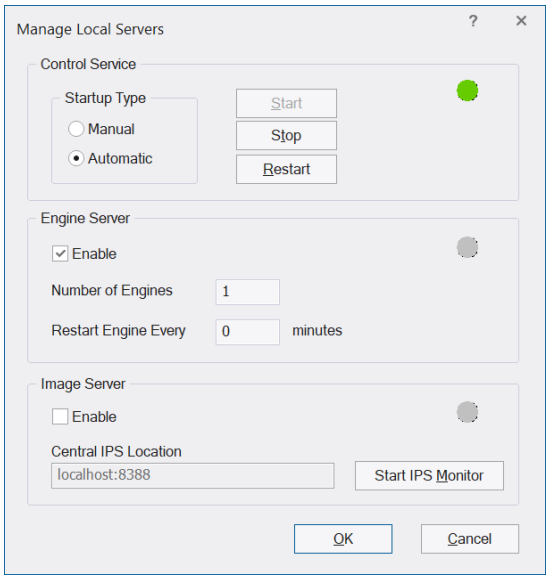

Fig. 4: Starting DIAEngines from the eCognition Developer to simulate an eCognition server environment.

After clicking "OK" the (left) disk should switch from grey to green (Fig. 3 and Fig. 4).

3. Storing analysis results

Usually, all analysis results are automatically stored in *eCognition project files* (*.dpr), which are saved in the *dpr* folder in your *eCognition workspace*. This means if you create an *eCognition workspace* in CRIB using the CLE command `DIAMkWksp`, once the rule set is executed a *.dpr file will be saved in the *dpr*-folder for each scene.

Rule sets can also export results, that is, either statistics or spatial files such as classified raster files or vector files. This is done by all the export processes the one or other way (see the *eCognition Developer manual* for details). To have the export results saved in the *workspace* created in CRIB, in your rule set you should choose as export mode "Static export item" (Fig. 5). In the CRIB *eCognition workspace* a folder named *results* (or the name you give it) will be created which contains all the results you export with the export process selected in the rule set. Note: the path can be edited when adding the export-process. Be careful when editing, since all expressions in curly brackets (`{:variable.name}`) are file name variables. Before saving and copying the rule set to CRIB you should change all backslashes (`\`) to slashes (`/`). We recommend saving two versions of the rule set: one to be executed on UNIX systems (the CRIB version) and one to be executed on Windows systems (the developer version).

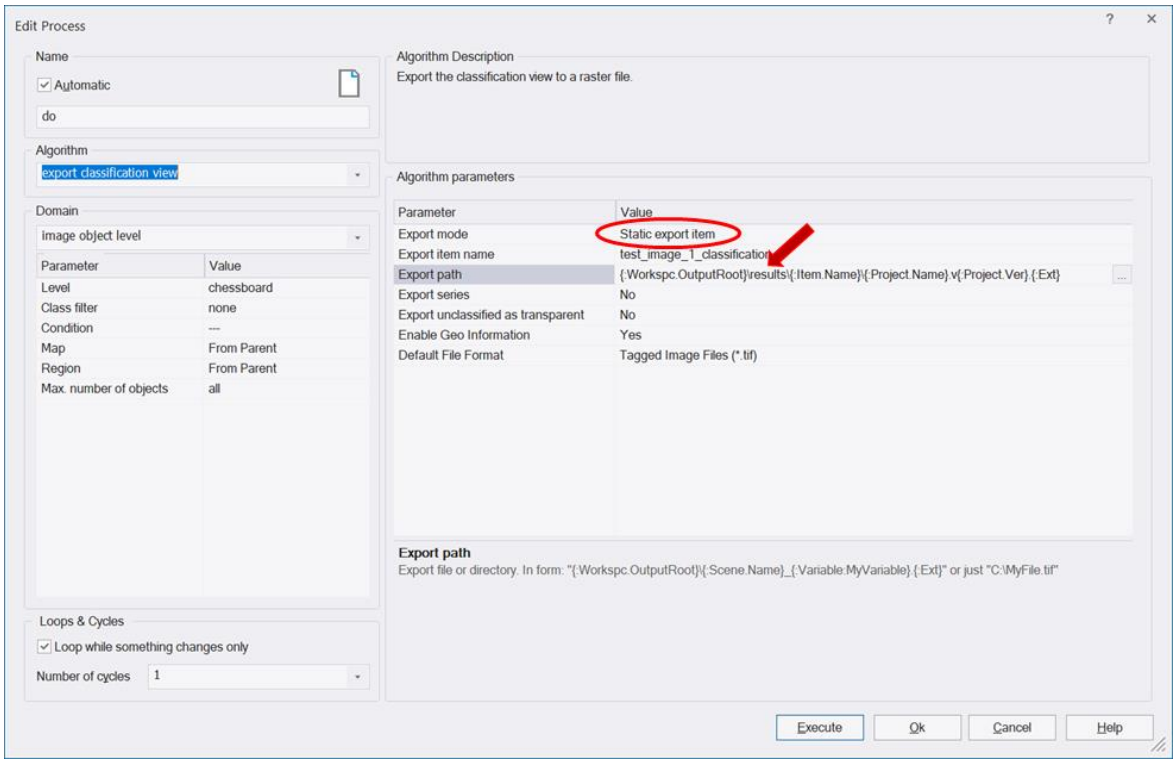

Fig. 5: Use „Static export path" as export mode in order to use relative paths for exporting your results in CRIB.

4. Executing the rule set on the data

As described above, processing larger amounts of image data (large number of images or tiled huge single images) in the CRIB-based *eCognition* client-server-environment means you need to copy the (image) data to be processed together with your developed rule set to the CRIB environment. Then create an eCognition workspace as described above using the CLE command `DIAMkWksp` and start `DIACmdClient` or `DIACmdEngine` and make sure the used path names in the rule set are in UNIX standard. Once you are fine with the results including the created *.dpr files you can copy them back to your local system to inspect the results.

5. Examples

5.1. Simple example of different colors and shapes

The following example illustrates in a very simple way the typical workflow when analyzing image data (and other data) within the eCognition environment as is installed in CRIB.

In the folder `/data/shared/eCognition` you will find in the subfolder `testdata/image data/triangles_squares_circles` several non-remote sensing images, all of them containing some geometric forms of different color. Additionally, in the subfolder `rule sets` you will find an eCognition rule set called `rule_set_test_images_geometric_forms.dcp`.

If you look into the image files you will see that they all contain different shapes with different colors (Fig. 6).

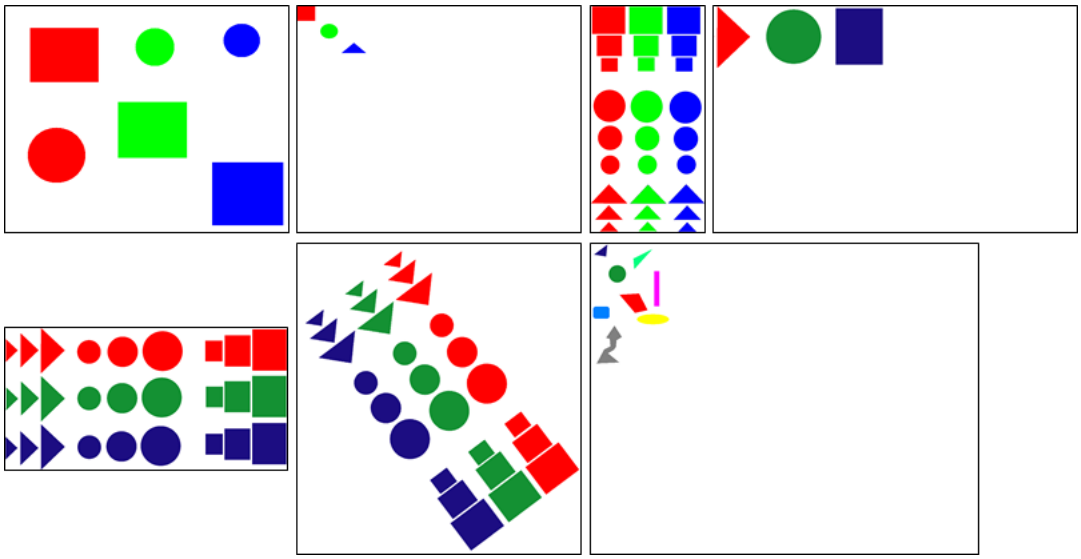

Fig. 6: Simple test images containing different shapes of different color.

The rule set for image analysis operates completely in batch mode, that is, no interactive actions are necessary to produce the results. Such rule sets could not be executed with the CRIB-based *eCognition* server installation, since *eCognition* clients and servers are not directly connected.

The rule set first does a chessboard segmentation and then merges equally colored one-pixel-objects to larger objects which have different shapes; all of them are looking like a circle, a rectangle or a triangle. The class hierarchy (Fig. 7) holds an extra class for every color and every shape, i.e.: **red**, **green**, **blue** and **circle**, **rectangle** and **triangle**. The classes are all described in fuzzy manner (read more about this in the *eCognition* user manual). Every object then is fuzzy assigned to one of the classes, whereas the class hierarchy uses multiple inheritance. That is, every object inherits from a shape-class and a color-class simultaneously, e.g.: a **red triangle** inherits from the class **red** its color assignment and from the class **triangle** its shape assignment. In some of the images – especially in image `test_image_7.bmp` some objects fulfill the criteria of several classes simultaneously but to different degrees, which means these objects are assigned to the class of which they fulfill the criteria at best (i.e. with the highest degree). Objects with classification values below 1.0 can be understood as objects being nearly a member of the respective class. E.g. the green triangle in the top-right of `test_image_7.bmp` has a classification value of 0.832 for **green triangle** and 0.658 for **green rectangle**. This is because for the parent class **green** the classification criteria are only fulfilled by 0.832 while for **triangle** they are fulfilled by 1.0 and for **rectangle** only by 0.658.

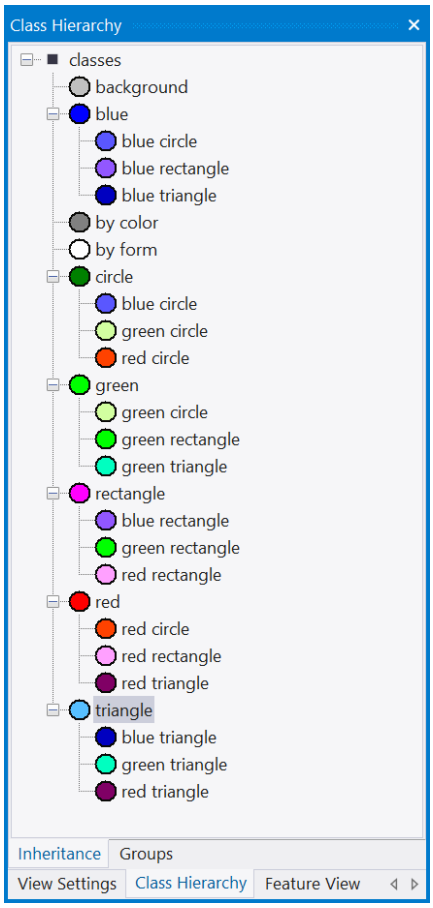

Fig. 7: Class hierarchy for the simple example with different shapes in different colors using multiple inheritance.

In the last step of the rule set the classification results are exported as a raster file per scene, whereas each pixel has the color of the class in the class hierarchy it was assigned to. Further, diverse statistics are exported as *.csv files: the number of objects per class and scene and the class assignment and the classification value (see above) per object. You can use the number of objects per class and scene to check if the process did everything properly. The classification values illustrate the certainty of the classifier for different objects or the grade each object fulfills the classification criteria per class. The latter can also be understood as a measure for the classifier's reliability or the ability the classifier can distinguish the classes. To learn more about fuzzy classifiers and the class hierarchy in *eCognition* please refer to the literature and/or read the manual.

To analyze the images using the above-described rule set you can put them into a workspace using the CLE command `DIAMkWksp`:

```
./DIAMkWksp /data/shared/eCognition/testdata/testworkspace
/data/shared/eCognition/testdata/image_data/triangles_squares_circles
```

This will create the *eCognition* workspace `testworkspace.dpj` in the folder `/data/shared/eCognition/testdata` (Fig. 8).

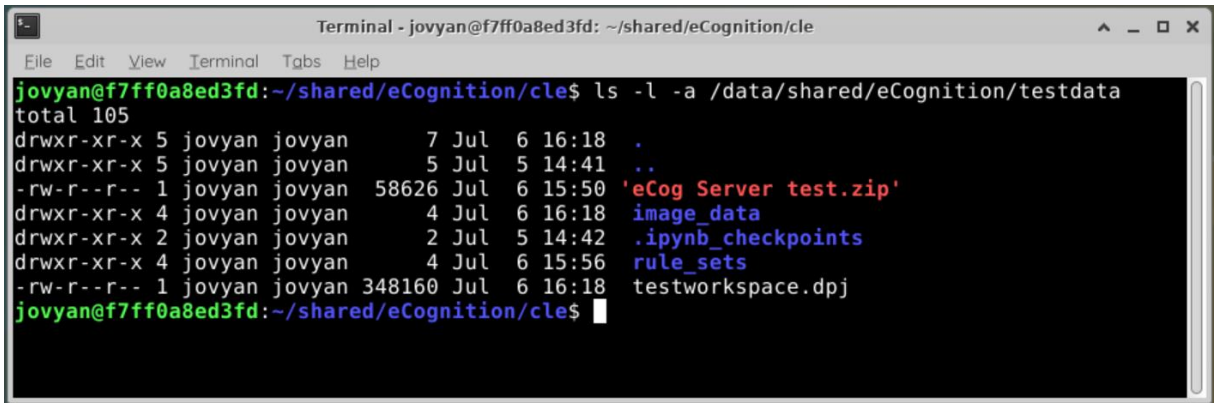

Fig. 8: Newly created workspace ‚testworkspace.dpj'

Then start the *DIACmdClient* using the freshly created workspace (*.dpj) and the rule set (*.dcp) you have created using the windows-based eCognition Developer, e.g.:

```
./DIACmdClient s /data/shared/eCognition/testdata/testworkspace.dpj
/data/shared/eCognition/testdata/rule_sets/triangles_squares_circles/rul
e_set_test_images_geometric_forms.dcp
```

And control whether everything has been executed using the job scheduler interface (Fig. 9).

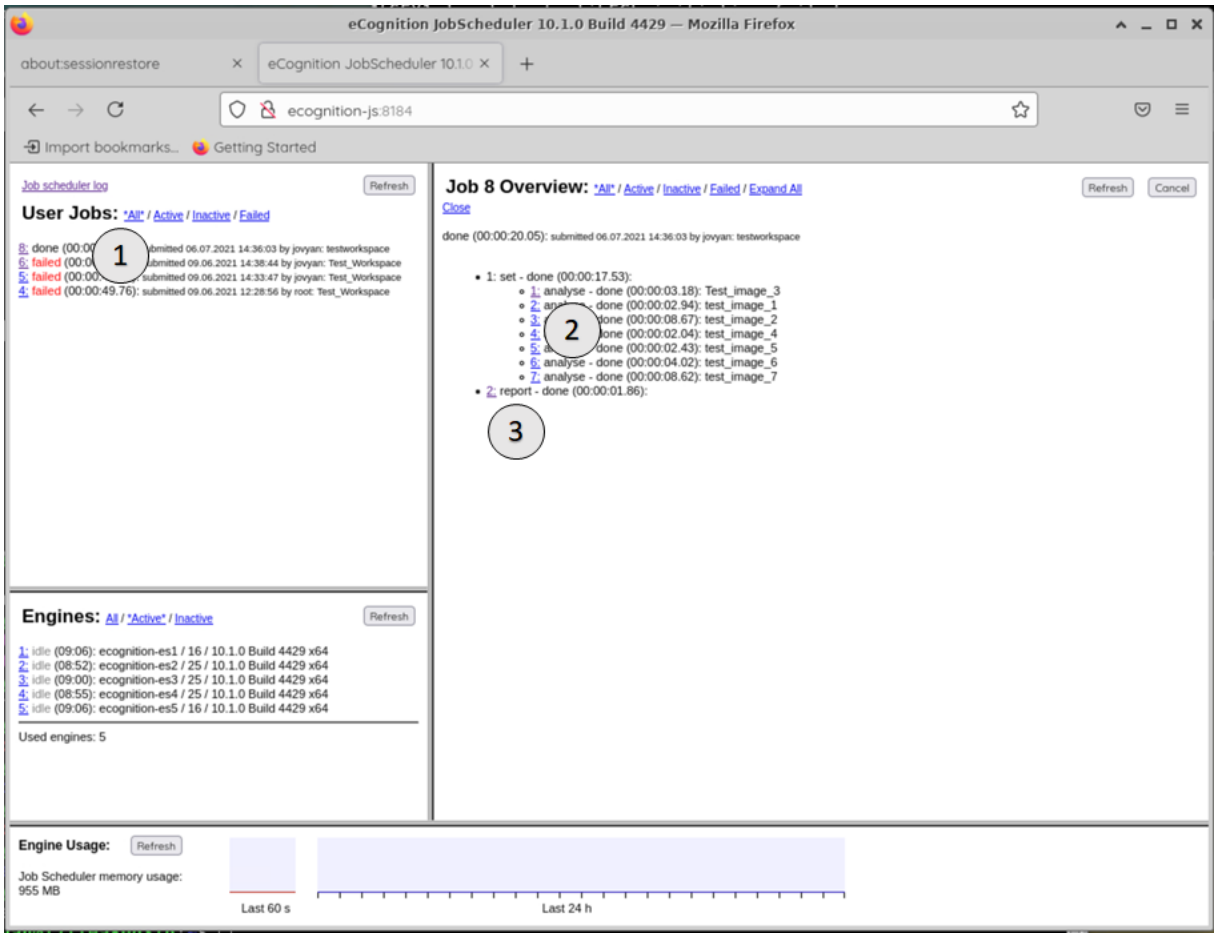

Fig. 9: HTML-based job scheduler interface to inspect which jobs have been executed successfully and where the results were stored.

Select a job which has been done (1). You can then either check the report (3) or have a closer look on the results being written per scene (2) by clicking one of the numbers.

The folders *dpr* and *results* are created automatically as like in the windows version of *eCognition* but in the folder `/data/shared/eCognition/testdata` (Fig. 10), i.e. where the workspace file (*.dpj) is located. Both hold the same files after executing the rule set as they would do in the windows version.

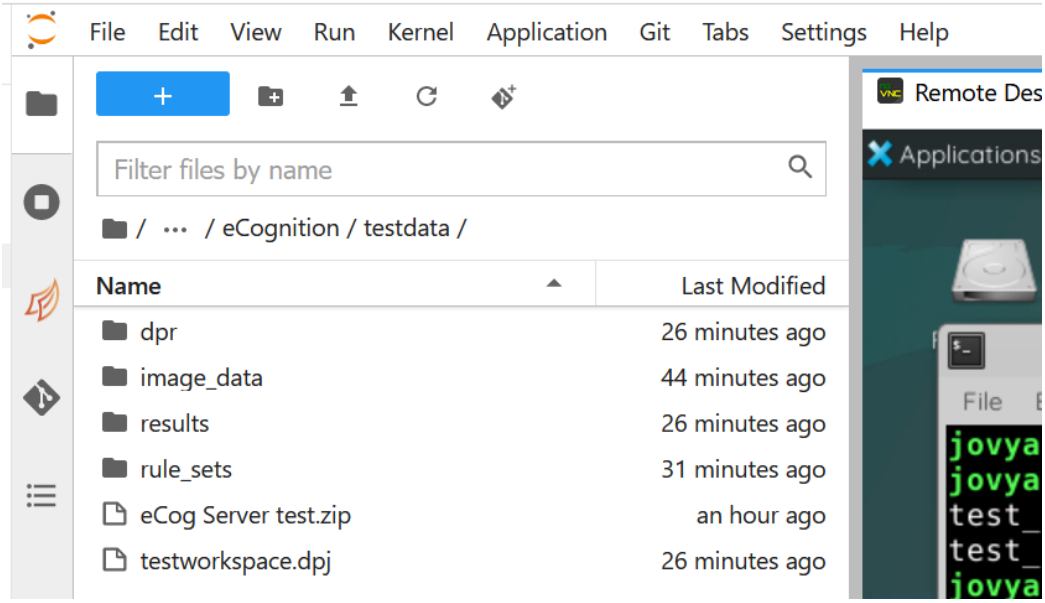

Fig. 10 Folders created automatically by the process during execution.

Now you can either use these results for further processing or zip them and copy them back to your local computer, e.g. use the generated *.dpr files to inspect them in the Windows-based *eCognition Developer*.

5.2. Supervised classification with tiling and stitching

Since no interactive actions or processes are possible in the CRIB-based *eCognition* server installation, for supervised classification methods samples need to be taken in the Windows based *eCognition* client before copying the data and rule set into the CRIB environment.

In the following a workflow is illustrated, which describes how larger images (here a subset of a Sentinel-2 scene) can be split into tiles, how these tiles can be analyzed in parallel using a supervised classifier and how the results are finally stitched for calculating global statistics.

The workflow can be subdivided into the following major steps:

1. Create tiles.
2. Segment one or more tile(s) and create samples (should be as representative as possible).
3. Copy and submit all tiles for processing to the scheduler which will distribute them to the CRIB-based eCognition servers.
4. Stitch the results and produce statistic for the whole, i.e. stitched, scene.

Since samples can only be taken in a Windows-based *eCognition* client it is necessary to take them before copying the rule set and the data into CRIB. Thus, some processes must be set to active/inactive before transferring data and rule set.

In the folder `/data/shared/eCognition/testdata/image_data/NL_S-2_2018-02-07_sub/` you will find a Sentinel-2 subset in ERDAS Imagine format (*.img): `nl_s-2_2018-02-07_sub.img`.

Create the eCognition workspace in CRIB by adding this single image:

```
./DIAMkWksp /data/shared/eCognition/testdata/NL_S-2_2018-02-07_WkSpce
/data/shared/eCognition/testdata/image_data/NL_S-2_2018-02-07_sub
```

Now copy the ZIP-file `NL_S-2_2018-02-07_sub_LULC.zip` from the CRIB-folder `/data/shared/eCognition/testdata/rule_sets/tiling_and_stitching` together with the Sentinel-2 subset to your local computer and start the *eCognition* Developer client. In the client create the same workspace as you did in the CRIB environment, that is, which contains the Sentinel-2 subset and load the rule set. The process tree should look like es illustrated in Fig. 11.

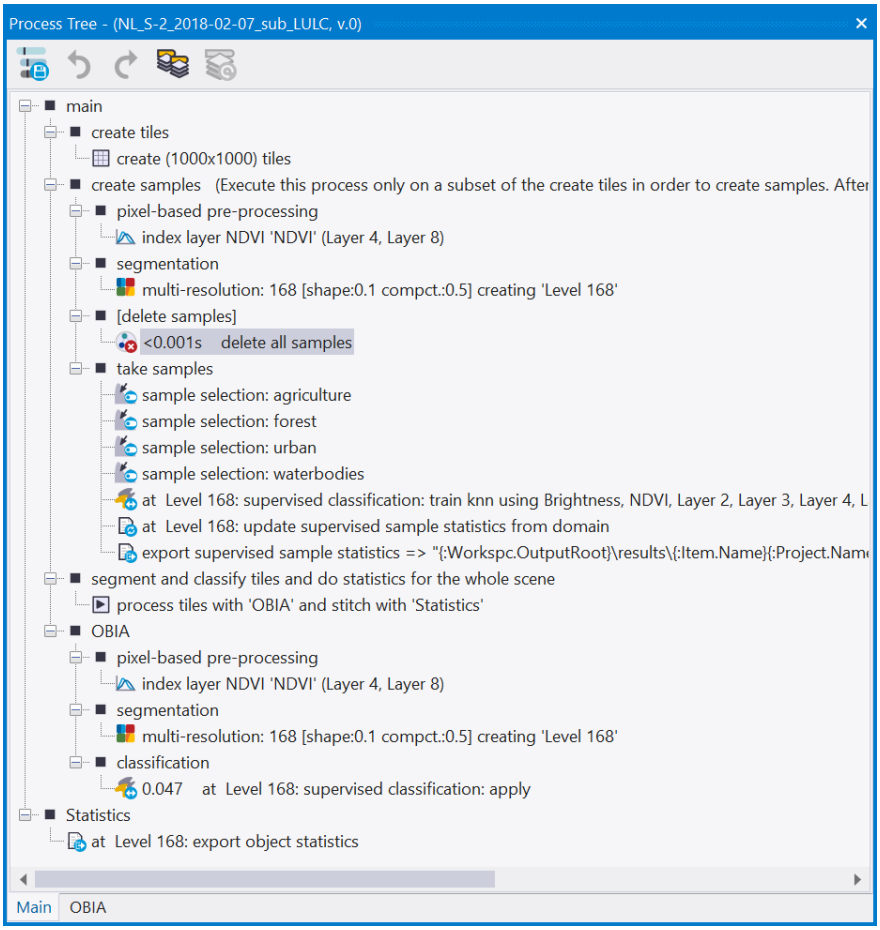

Fig. 11: Process Tree for rule set to be used for tiling and stitching od Sentinel-2 subset.

The workspace only contains the single scene (Fig. 12).

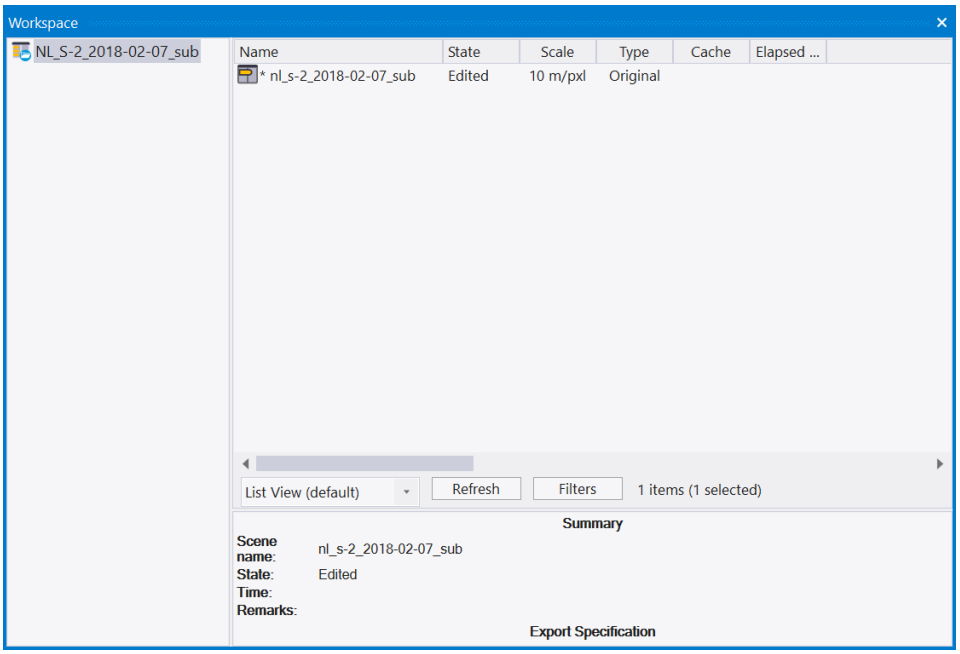

Fig. 12: Workspace after creation for Sentinel-2 subset.

Then set the bands for calculating the objects' brightness: go the menu *Classification -> Advanced Settings … -> Select Image Layers for Brightness …* and select the Layers *Layer 2, Layer3, Layer4* and *Layer 8* (= the four bands with 10m resolution). Now execute in the process tree the process 'create tiles'. The workspace now should show up all the tiles (Fig. 13).

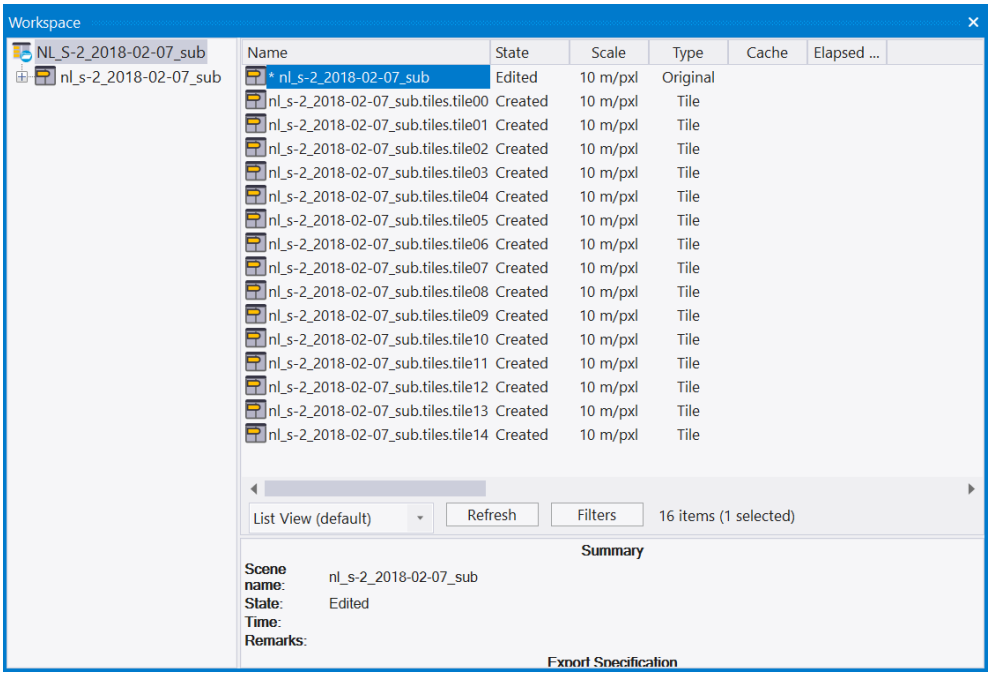

Fig. 13: Workspace after creation of tiles for Sentinel-2 subset.

You can change the size of the tiles by editing the process. You can inspect each tile by double clicking on it. Questions concerning saving the current project can be ignored.

Now select a tile of your choice (double click on it) and execute the processes ‘pixel-based pre-processing’ (creates the NDVI) for the selected tile. Then do the same for the ‘segmentation’ process (executes a Multi-Resolution Segmentation with a scale parameter of 4 * 42 = 168 based on the 10m-resolution bands of Sentinel-2.

After successful segmentation select for each class according samples. You can do this by executing the sub-processes ‘sample selection: agriculture’, ‘sample selection: forest’, ‘sample selection: urban’ and ‘sample selection: waterbodies’ individually each. Please do not execute the parent process ‘take samples’. Samples are selected by double-clicking them in the image window or using the “spray mode” (see eCognition Developer User Guide for details). The image window should show taken samples in the color of their class (Fig. 14). If you want to create samples from several tiles repeat this step accordingly for the tiles of your choice.

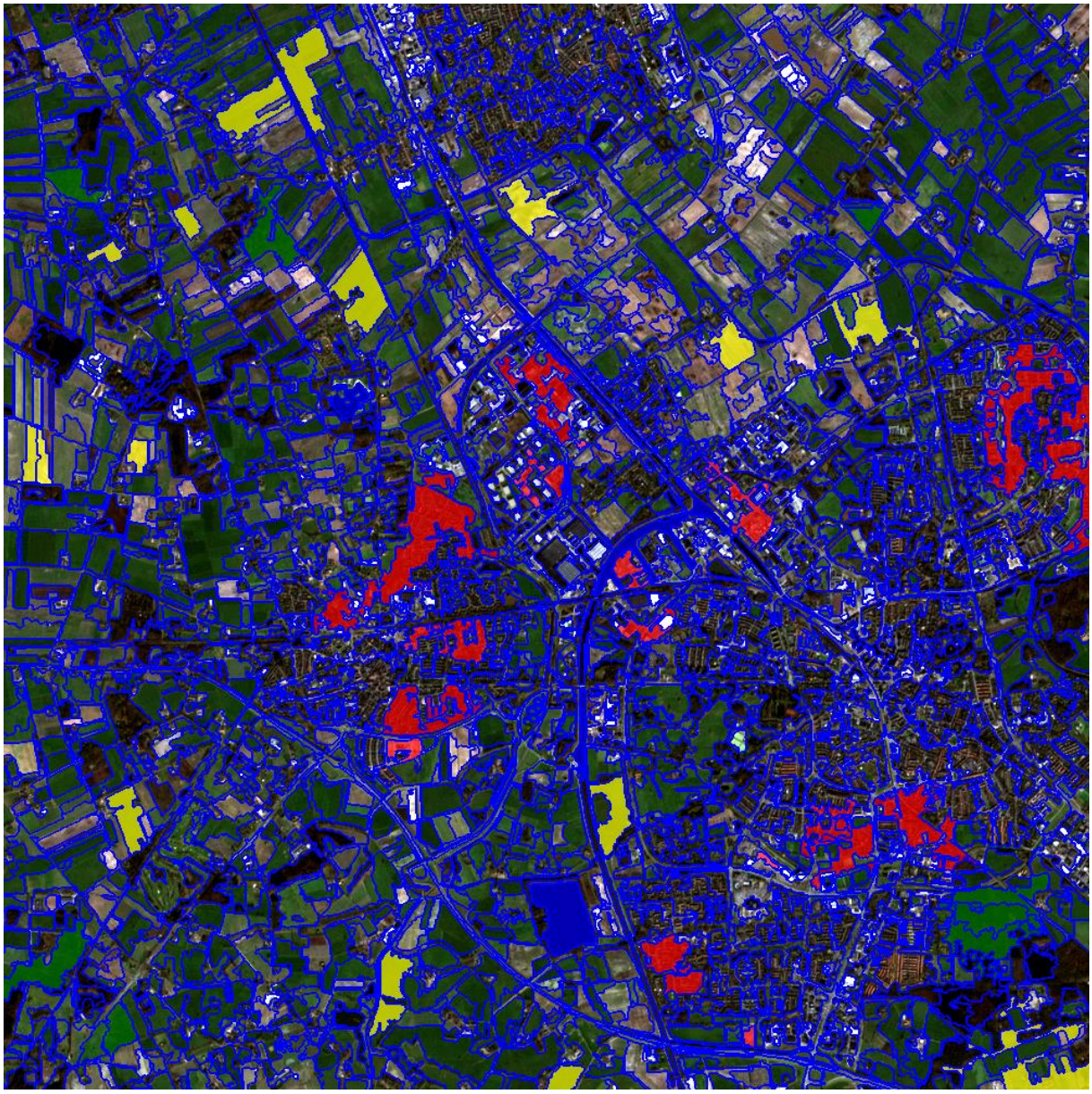

Fig. 14: Samples taken in a segmented tile.

With these samples you can train the classifier now by executing the process ‘supervised classification’. If you open the process, you will see that it is in ‘train’ mode which means the samples were taken for training and stored in the variable “kNN_samples”. The feature space for the kNN-classifier in this case is spanned by the *NDVI*, *Brightness* (calculated from the 10m resolution bands), the bands’ contribution to an object’s brightness (*Ratio Layer* n) and the standard deviation per 10m-resolution-band and object (*Standard Deviation Layer* n). Further details about this algorithm can be found in the user manual and reference book (Fig. 15).

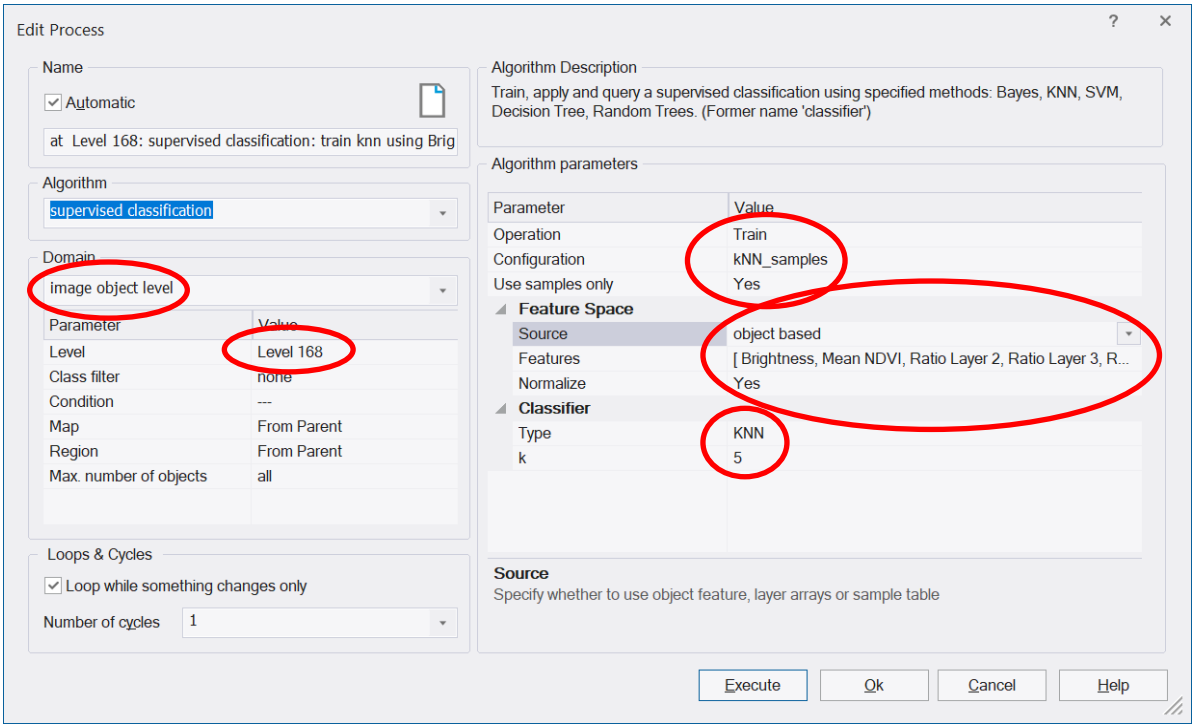

Fig. 15: Settings of the process ‘supervised classification’ for the sample-based kNN-classifier during training.

In the next two steps statistics about the token samples are calculated and stored (process ‘update supervised sample statistics from domain’ and ‘export supervised sample statistics’).

To check whether the classifier creates satisfying results, you can classify the current tile by executing the process ‘classification’ under the tree ‘OBIA’.

To process all the tiles and stitch them after processing make sure that the sub-routine ‘OBIA’ exists by checking the sub-routine tabs in the process tree window (Fig. 16). If it does not exist, please create it by right-clicking on the ‘main’ tab and select ‘Add new’. In the upcoming dialog enter the name ‘OBIA’. Then copy from the main process tree the sub-tree ‘OBIA’ into the newly created sub-routine.

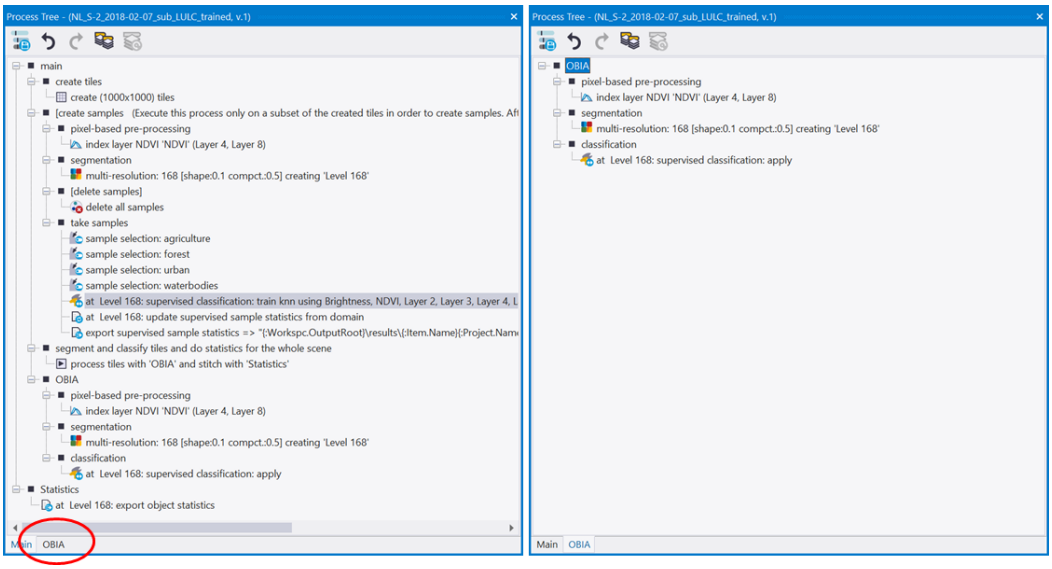

Fig. 16: Settings of the process 'supervised classification' for the sample-based kNN-classifier during training.

Before you safe the rule set make sure that all processes containing path names follow UNIX standard either explicitly (Fig. 17) or using the script `correct_paths.sh` after copying it to CRIB. In this particular case this affects the process 'Statistics' and its sub-processes.

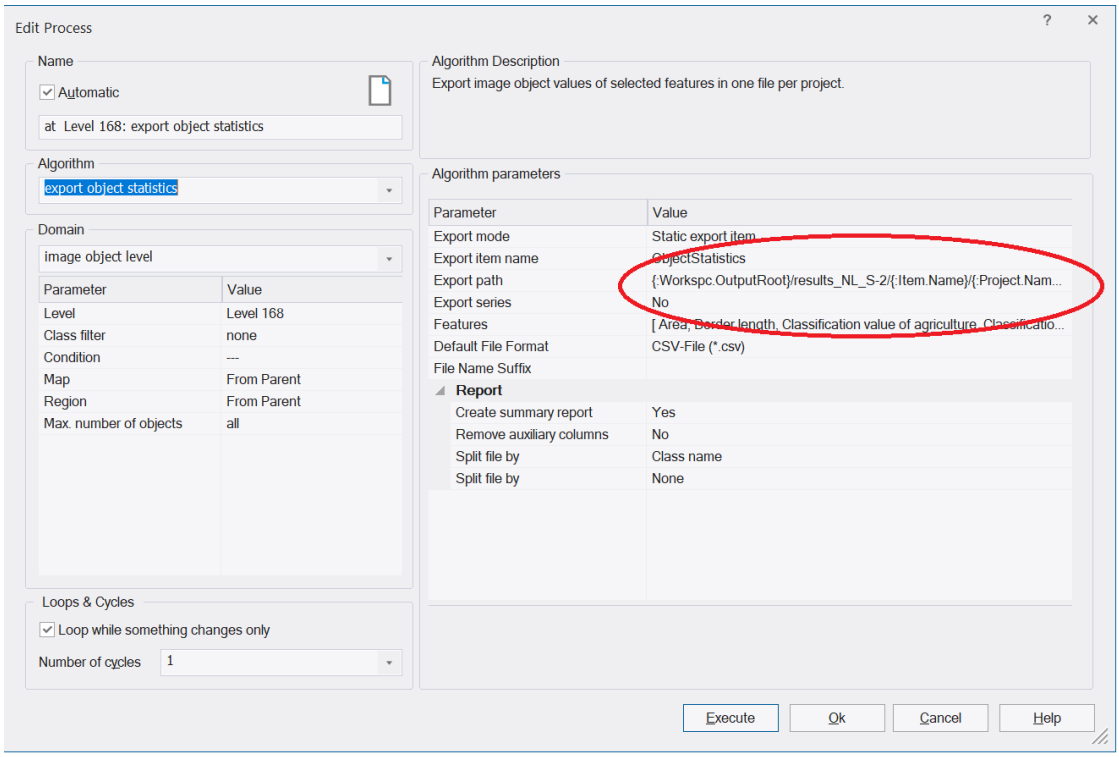

Fig. 17: Editing the export-path according to UNIX conformity.

Now safe the rule set e.g., under the name `NL_S-2_2018-02-07_sub_LULC_trained_UNIX.dcp` and copy it back to CRIB and translate it to UNIX notation if necessary. Then you should be able to apply it together with the created workspace and the eCognition server using the CLE command `DIACmdClient`. To inspect the results copy them back to your Windows computer and start eCognition. The exported

statistics, stored in the *.CSV file can be opened with any software which supports *CSV format or you can use an ASCII editor (which is a bit unhandy).