Image Segmentation Pipeline

The image segmentation pipeline has the purpose to predict a class label for each pixel of an image. To do so a neural network (Unet, https://arxiv.org/abs/1505.04597) has to be trained. This document will instruct you how to install and use the pipeline.

1. Installation

The pipeline should be installed on a computer with a relatively powerful GPU. To install it, carry out the following steps:

1. Download the ImageSegmentation Github repository and extract it. This can either be done by downloading the repository as a zip file from https: https://github.com/tschutli/ImageSegmentation and extracting it or by cloning the repository using the following command:

   git clone https://github.com/tschutli/ImageSegmentation

2. Download Anaconda with Python 3.7 from https://www.anaconda.com/distribution/ and install it.
3. Open the Anaconda Prompt Program, change to the previously downloaded ImageSegmentation folder and run the following command:

   cd /???/ImageSegmentation
   conda env create -f environment.yml

   This will install all python dependencies including the Tensorflow library with GPU support.
4. Done! These three steps should be sufficient to install everything you need to run the Image Segmentation Pipeline. Go to the Usage section to learn how to use it.

2. Usage

Before being able to use any of the python scripts, make sure that the Anaconda environment is activated. To do so, open Anaconda Prompt and change to the Image Segmentation directory. Then activate the tf_gpu environment that you installed during the installation instructions:

cd /???/ImageSegmentation/Python
conda activate tf_gpu

2.1 Preparation for Training

To train a network, you need data. Namely images and shapefiles containing the information what is visible within the images. The images should be geo referenced in the geotiff format.

2.1.1 Preparing Images

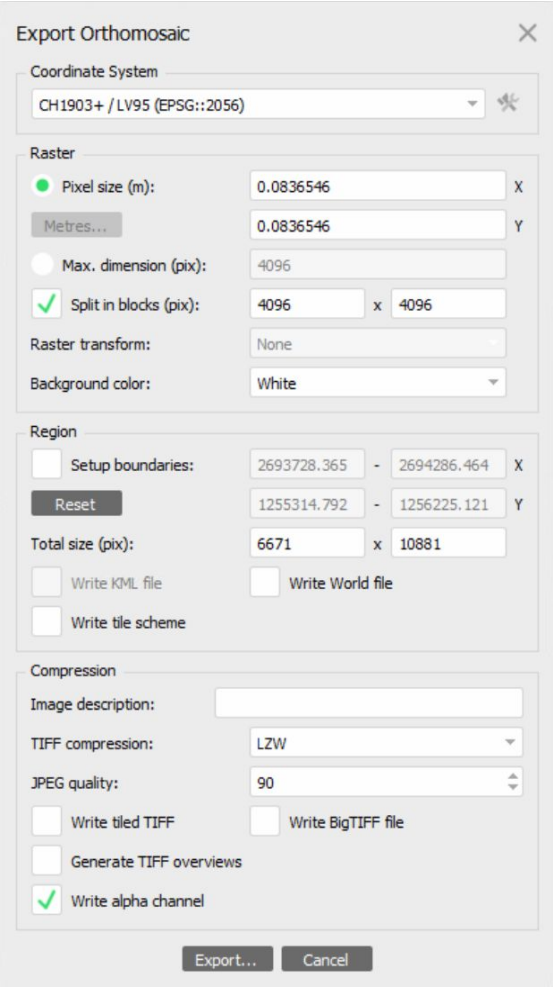
A good starting Point is an Agisoft Project. You can export the Agisoft Project as an orthomosaic.

Export the orthomosaic with the "Write alpha channel" option enabled.

I suggest to split the orthomosaic in blocks of size 4096x4096 pixels.

The coordinate system does not matter since the Image Segmentation pipeline will automatically detect the coordinate system from the images and shape files and will convert it if necessary.

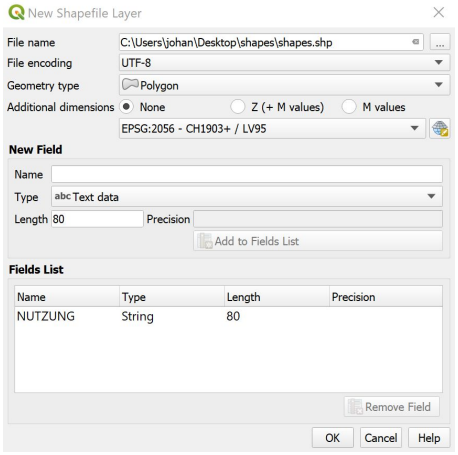
Alternatively, if you want to train on single images, these can be exported as orthophotos as well. Make sure to activate the "Write alpha channel" as well.



2.1.2 Preparing Shape Files

If you do not have shape files already, you can create them with QGIS. Carry out the following steps:

1. Create a new Project: Project -> New
2. Import the Images: Layer -> Add Layer -> Add Raster Layer
   It might help to use reduced resolution tiles. You can export reduced resolution tiles in Agisoft if you change the Pixel size in the export dialog in the previous step.
3. Create new shape files: Layer -> Create Layer -> New Shapefile Layer
   - Choose a location where the shapefiles should be saved to and name it shapes.shp.
   - Choose geometry type Polygon
   - Add at least one field to the fields list. It should be of type string. This will be the classification class.



4. To edit the shapefile, select the shapes Layer in the Layer overview and click the “Toggle Editing” button.

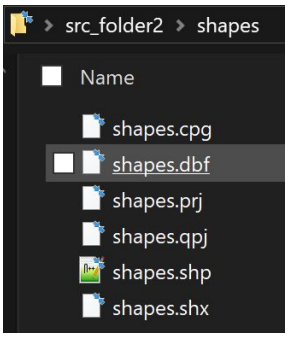
5. To add a new Polygon, click the “Add Polygon Feature” button. Then click on the Image to draw the polygon. When you finished adding vertices to the polygon, make a right click. A window will appear where you can set the attributes of your new Polygon.

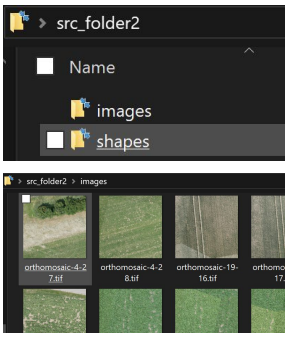
2.1.3 Pipeline Preparation

At this point, you should have a set of geo referenced Images and a shapefiles (.cpg, .dbf., .prj, .qpj, .shp, .shx). Carry out the following steps to prepare everything for training a network:

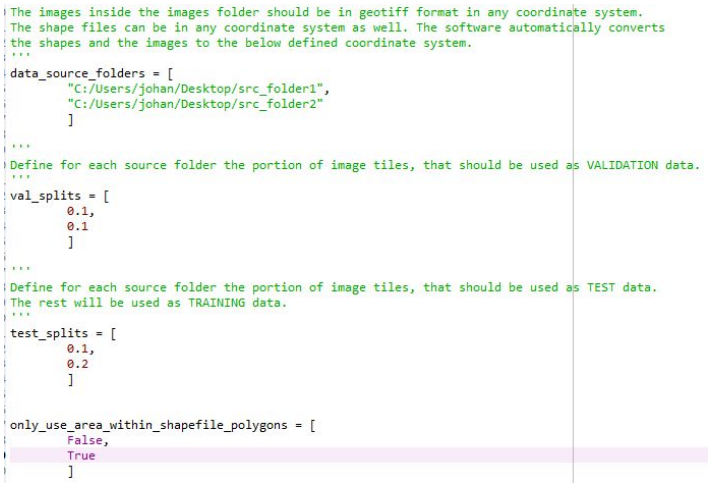
1. Create a source folder with the following subfolders: images, shapes. Place all your georeferenced images (.tif) inside the images folder. Place your shapefiles inside the shapes folder. Make sure to call them shapes.xxx as in the image on the right.

   You can also create multiple such source folders.





2. Open the constants.py file inside the ImageSegmentation/Python folder. You can see a variable called data_souce_folders. Define all your source folders there.
3. Also specify for each source folder what portion of this dataset should not be used for training, but for validation or testing. In the example configuration on the right, 10 percent of each of the two source dataset is used for validation and testing.



```
'''
The images inside the images folder should be in geotiff format in any coordinate system.
The shape files can be in any coordinate system as well. The software automatically converts
the shapes and the images to the below defined coordinate system.
'''
data_source_folders = [
        "C:/Users/johan/Desktop/src_folder1",
        "C:/Users/johan/Desktop/src_folder2"
        ]

'''
Define for each source folder the portion of image tiles, that should be used as VALIDATION data.
'''
val_splits = [
        0.1,
        0.1
        ]

'''
Define for each source folder the portion of image tiles, that should be used as TEST data.
The rest will be used as TRAINING data.
'''
test_splits = [
        0.1,
        0.2
        ]


only_use_area_within_shapefile_polygons = [
        False,
        True
        ]
```

4. Additionally, in the only_use_areas_within_shapefile_polygons variable, specify for each source folder, whether the network should only be trained on the areas within the shapefile polygons (True) or whether all areas that are not within a polygon of the shapefiles should be classified as background (False). HINT: Choose True if there are non-background regions in your images that are not covered by shapefile polygons. Otherwise choose False.
5. Also inside the constants.py file, define a working_dir. The software will use this directory to store all processed data to, including trained models and intermediate results.
6. Again in the constants.py file, define the ground_sampling_distance variable. All images are all scaled to match this ground sampling distance. It is defined in m/pixel. If you are not sure, which ground sampling distance you should use, use the “Pixel size” value in the Export Orthomosaic dialog in Agisoft. (See step 2.1.1)
7. Finally define the classification_class variable. This should be the field string in your shape file that defines the class. (In our example in 2.1.2: NUTZUNG)
8. Optionally, define a custom EPSG_TO_WORK_WITH variable. Use 2056 for the Swiss CH1903+ / LV95 coordinate system. If the shapes or images are not in the specified coordinate system, they will be converted to it.
9. Open Anaconda Prompt and run the following commands:

   cd /???/ImageSegmentation/Python
   conda activate tf_gpu
   python preprocessing.py

   This will prepare everything for training the neural network:
   - The input images are resized to the desired ground sampling distance.
   - The images are converted to the defined coordinate system.
   - They are tiled into tiles of size 256x256 pixels.
   - The mask images are created from the shapefile information.

2.2 Training

Execute the following steps to train a network.

1. Open the constants.py file inside the ImageSegmentation/Python folder. Make sure the working_dir is correct. Define the batch_size variable. More Powerful GPU's can handle larger batch sizes. This speeds up the training significantly. A Nvidia GeForce 1080 can handle a batch_size of 20.
2. In the constants.py file also define whether data augmentation options should be used (data_augmentation). If set to True, random horizontal and vertical flips as well as moderate brightness adjustments are applied to the train images.
3. Open Anaconda Prompt and run the following commands:

    cd /???/ImageSegmentation/Python
    conda activate tf_gpu
    python train.py

This will train the network for 100 epochs. If the training is not making improvements during 10 epochs it is stopped early. Once the training is finished, the trained model can be found in the working_dir as trained_model.h5.

CSV as well as Tensorboard Logs will be saved to the working_dir/logs directory. You can analyze the training progress graphically in real time by executing the following commands inside a separate conda prompt:

conda activate tf_gpu
tensorboard --logdir "/???/working_dir/logs"

Then open a web browser and type in the following address: localhost:6006.

2.3 Evaluating

Once a network is trained, you can evaluate its performance on the test set. Make sure the working_dir inside the constants.py file is the same as used during training. Then simply run the following commands inside the Anaconda Prompt:

cd /???/ImageSegmentation/Python
conda activate tf_gpu
python evaluate.py

The evaluate command will print the precision, recall and F1 score of each class, as well as a confusion matrix. All stats are also saved to the working_dir/logs/evaluate_log.txt file.

2.4 Predicting

To make predictions on new images, carry out the following steps:

1. Open the constants.py file inside the ImageSegmentation/Python folder. Define the folder_with_images_to_predict variable. It should be a path to a folder containing images. Also define the predictions_output_folder variable. It should be a path to an empty folder. The predictions will be saved to this folder.
2. Inside the constants.py file define the trained_model and label_map variables. These should be the paths to the trained model and the corresponding labelmap. These are saved into the working_dir during training.
3. Make sure that the ground sampling distance of the images inside the folder_with_images_to_predict is similar to the ground_sampling_distance used during training.
4. Run the following commands inside the Anaconda Prompt:

   cd /???/ImageSegmentation/Python
   conda activate tf_gpu
   python predict.py

5. Check the predictions_output_folder for the results.