User Manual

NEU4BRIX Forest Biomass processor

Issue / Revision: 0/0

**Date:** 26/06/2024

**IFAC reference:** NEU4BRIX\_User\_Manual\_v01

Prepared by

Emanuele Santi

Senior Researcher

IFAC STAFF



Table of Contents

1. Scope of this document 3

2. Reference Documents 3

3. Contents of the delivery 3

4. Hardware and software requirements 5

Hardware 5

Software 5

5. How to install 5

6. Configuration options 5

7. Algorithm Inputs 7

8. Algorithm outputs 9

9. How to run the code 9

Training 11

Testing 12

Validation 12

10. End of execution. 14

List of Figures

Figure 1. Input and output directory structure 4

Figure 2. configurable paths and filenames. 5

Figure 3. BIOMASS\_ANN\_RF.py in Spyder. 10

Figure 4. summary of the main configuration settings at the very beginning of execution 11

Figure 5. training execution and results (RF case). 11

Figure 6. example of console outputs during testing 12

Figure 7. example of validation outputs 13

Figure 8. End of execution message 14

Document Revision Status

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Last Edited Date | Revision/ Release Number | Status | Edited By | Pages / Paragraphs Affected | Change Ref |
| 17/03/2025 | 0.1 |  | Emanuele Santi | First draft version |  |

## Scope of this document

This document provides the basic information to install and run the NEU4BRIX processor delivered by IFAC for estimating forest AGB from airborne P-band SAR data collected in the AfriSAR campaign. The activity has been carried out in the framework of Biomass Retrieval Intercomparison Exercise (BRIX I and II) and NEU4BRIX follow on.

**⚠️ The code has been defined to work with the calibrated and coregistered files derived from Afrisar acquisitions made available by ESA in the framework of BRIX. Sample files are delivered along with the code.**

## Reference Documents

|  |  |  |  |
| --- | --- | --- | --- |
| AD# | Title | Doc # | Date |
| 1 | Santi, E.; Paloscia, S.; Pettinato, S.; Cuozzo, G.; Padovano, A.; Notarnicola, C.; Albinet, C. Machine-Learning Applications for the Retrieval of Forest Biomass from Airborne P-Band SAR Data. Remote Sens. 2020, 12, 804. | [doi: 10.3390/rs12050804](https://doi.org/10.3390/rs12050804) | 2020 |
| 2 | M. Simard, N. Pinto, J. B. Fisher, and A. Baccini, “Mapping forest canopy height globally with spaceborne lidar”, *J. Geophys. Res.*, 116, G04021, 2011. | doi:10.1029/2011JG001708 | 2011 |

## Contents of the delivery

* **BIOMASS\_ANN\_RF.py main code** in Python that applies the ML algorithms (Artificial Neural Networks – ANN and Random Forests – RF), by performing training, testing and validation.
* **Suite of** **functions** needed to run the code in the “function” subfolder.
* **ANN and RF files** containing the structure of pretrained algorithms to be used for testing**.**

**⚠️ these files can be updated by repeating the training (see section 9)**

* **Calibrated and coregistered input images,** one for each site (La Lope, Mabounie, Mondah and Rabi) obtained by calibrating, coregistering and mosaicking the images provided by ESA in the framework of BRIX.
* **ROI data** containing the coordinates of the polygons in which the in-situ measurements have been carried out**.**
* **writeROIlist.py code** that associates the polygons with the AGB measurements for each test area in a format that can be used by the code**.**
* **Training.csv sample training file** containing backscattering, incidence and auxiliary tree height [RD #2].

An example of folder structure in which the data are stored is shown in Figure 1. Paths and filenames can be configured in the parameter definition section at the top of the code (see section 6).

Immagine che contiene testo, schermata, Carattere, design

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 1. Input and output directory structure

The **Input data** folder (**DataIN** in the example) contains IMAGES, ROI and TRAIN subfoldes.

* **IMAGES** subfolder contains one subfolder for each area (La Lope, Mabounie, Mondah and Rabi), each subfolder contains the input backscattering and auxiliary data (Incidence angle and Tree height from Simard – see RD 02).
* **ROI** subfolder contains all the shapefiles provided by ESA and the ROIlist.npy files containing the list of shapefiles and associated AGB values for each area in the 4 subfolders.
* **TRAIN subfolder** contains the training file (csv format) for repeating the training of ML model if needed (see section 9)

The **DataOUT folder** contains one subfolder for each area in which the results (output AGB maps) are saved.

The **ML folder** has ANN and RF subfolders containing “ANN.net” and “RF.rff” files with saved configuration of ANN and RF, respectively. The configuration is overwritten in case the training is repeated.

## Hardware and software requirements

### **Hardware**

the code has been successfully tested on a desktop machine with the characteristics listed below. The possibility to run the code in different hardware and OS configuration must be verified.

* CPU: Intel® Core ™ 19-10900 CPU @2.80 GHz
* RAM 128 Gb
* HDD 4Tb

### Software

* OS: Windows 10
* Standard Administrator rights.
* Development environment: Python 3.8 (Anaconda 3.0). The packages to install/import are specified at the top of the code and functions, they include:
* Gdal
* glob
* joblib
* neurolab (for ANN development)
* numpy
* os
* pathlib
* ScyPy
* sklearn.ensemble (for RF development)
* sys
* time

## How to install

Simply unzip the delivered NN4BRIX.zip archive in the working directory. The input images – too large to be included in the zip – must be placed in the input directory configured at the top of the main code (see section 6).

## Configuration options

The configuration options are listed at the top of the main code BIOMASS\_ANN\_RF.py, these options include flags, paths and filenames, general parameters, and ANN and RF specific parameters. The configurable paths and filenames are at lines 51 – 68 of the code, as shown in Figure 2, ANN, RF and general parameters are defined at lines 71 - 85.

Immagine che contiene testo, Carattere, schermata

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 2. configurable paths and filenames.

The configurable parameters are listed in Table I.

Table I. List of configurable parameters

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Description** | **values** |
| **Flags** | | |
| trainflag | To enable training | “true”, “false” |
| testflag | To enable test (application of the trained ML to images contained in *indir* folder) | “true”, “false” |
| MLmethod | To select the ML method between ANN and RF | “ANN”, “RF” |
| **Paths and filenames** | | |
| Indir | To select the main path for input data | Any subdirectory of the working directory |
| Traindir | To set the training data path (must be a subdirectory of *indir*) | Any subdirectory of *Indir* |
| Outdir | To select the main path for output data | Any subdirectory of the working directory |
| ANNdir | path to save/load the ANN | Any subdirectory of ML directory |
| RFdir | path to save/load the RF | Any subdirectory of ML directory |
| trainfile | Name of the training file name in *traindir* | Any file name with *csv* extension |
| ANNfiguretrainfile | Name of the png file showing the training result for ANN (if trainflag enabled and ML method set to ANN) | Any file name with *png* extension |
| RFfiguretrainfile | Name of the png file showing the training result for RF (if trainflag enabled and ML method set to RF) | Any file name with *png* extension |
| ANNfile | Filename to save/load the ANN | Any file name with *net* extension |
| RFfile | Filename to save/load the RF | Any file name with *rff* extension |
| normfile | file containing data normalization parameters (for training ANN) | Any file name with *npy* extension |
| ROIfile | To set the ROI data file | Any file name with *npy* extension |
| **General parameters** | | |
| BIOMASSlow | lower limit of biomass value (t/ha) for data filtering and plots | Depends on the range in training, default=0 |
| BIOMASShigh | higher limit of biomass value (t/ha) for data filtering and plots | Depends on the range in training, default=600 |
| trainperc | % of data in training file to be used for training | Training file will be composed by *trainperc* of data sequentially picked: e.g. *trainperc*=20 (%) means that one sample every 5 is used for training |
| **ANN parameters** | | |
| nepochs | number of ANN training iterations | Default=500 |
| valgoal | target error to stop training | Default=0.1 |
| nshow | training error is printed on screen every *nshow* iteration | Default=10 |
| nneu | number of neurons in hidden layers (the proposed ANN has 2 hidden layers with *nneu* neurons each) | Default=10 |
| **RF parameters** | | |
| n\_est | number of estimators | Default=500 |
| rnd\_state | Random state | Default 0 |

## Algorithm Inputs

The inputs required by the processor to generate the AGB maps are listed in Table II.

Table II. Input list

|  |  |  |
| --- | --- | --- |
| **Data description** | **filename** | **Data format** |
| **For Training - Required only if trainflag is enabled** | | |
| Backscattering at the available polarization plus LIA, Hlidar (see RD02) and target AGB from in-situ | Training.csv (configurable) | Csv file with 7 columns containing in the (strict) order: HH,HV,VH,VV,LIA,HL,BIOMASS |
| **For testing (generating the AGB Maps) – required only if testflag is enabled** | | |
| Calibrated, geocoded and mosaicked backscattering in VV polarization. | GRD\_afrisar\_dlr\_VV.tif (configurable) | Geocoded Tiff containing the acquired backscattering in VV polarization |
| Calibrated, geocoded and mosaicked backscattering in VH polarization. | GRD\_afrisar\_dlr\_VH.tif (configurable) | Same as above for VH polarization |
| Calibrated, geocoded and mosaicked backscattering in HV polarization. | GRD\_afrisar\_dlr\_HV.tif (configurable) | Same as above for HV polarization |
| Calibrated, geocoded and mosaicked backscattering in HH polarization. | GRD\_afrisar\_dlr\_HH.tif (configurable) | Same as above for HH polarization |
| Geocoded and mosaicked Local Incidence angle (LIA). | GRD\_afrisar\_dlr\_LIA.tif (configurable) | The LIA for VV is considered |
| geocoded and mosaicked tree height | GRD\_afrisar\_dlr\_HLidar.tif (configurable) | Tree height map (see RD 02) |
| **For validation** | | |
| ROI list file | ROIlist.npy (configurable) | association between ROIs within the image frame (only one hectare rois have been considered) and AGB |

## Algorithm outputs

The outputs generated by the processor are listed in Table III.

Table III Output list

|  |  |  |
| --- | --- | --- |
| **Data description** | **filename** | **Data format** |
| **Training outputs** | | |
| architecture of trained ANN | ANN.net (configurable) | Neurolab compatible format |
| Architecture of trained RF | RF.rff (configurable) | sklearn.ensemble compatible format |
| **Testing outputs** | | |
| AGB map | In the form of Filename+MLflag+”BIOMASS”.tif (e.g. GRD\_afrisar\_dlr\_ANN\_BIOMASS.tif -configurable) | 1st band of the Geocoded Tiff generated by the ML algorithm and saved I the specified target path |
| MASK | Same file as above | 2nd band of the ML algorithm output, containing a mask for not valid outputs or not retrievable (not forests) targets |
| AGB preview | Same as above but with “.png” extension | Png preview of the output AGB map |
| **For validation** | | |
| Validation scatterplot | Png file | Scatterplot estimated vs in-situ AGB from ROIs with main statistics on display |

## How to run the code

To start the execution, open the main code BIOMASS\_ANN\_RF.py in Python console (e.g. Spyder), configure the parameters in the tables above, if necessary, verify that the required libraries are installed and start the execution.

**⚠️The code has been provided with the configuration parameters already set for working on the images that have been delivered along with the software.**

A screenshot of the Spyder console is shown in Figure 3:

Immagine che contiene testo, schermata, schermo, software

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 3. BIOMASS\_ANN\_RF.py in Spyder.

At the very beginning of execution, the main settings are summarized at prompt (see Figure 4) and some checks for input data are performed. Checks include the presence of input images in the specified folder, the presence of ANN or RF configuration files in the specified folder if training is disabled (i.e. only test and/or validation are configured), and the presence of training file in the specified directory if training is enabled.

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 4. summary of the main configuration settings at the very beginning of execution

### Training

If trainflag is enabled, training of the ML method selected by *MLmethod* flag begins. This process takes in general longer for ANN than RF, but it depends on the parameter setting, e.g. number of neurons in hidden layers for ANN or the size of training set. When the training is completed, the test over the entire training file is shown as in figure and saved in the selected ML directory along with the architecture of the trained ML algorithm.

Immagine che contiene testo, schermata, Diagramma, diagramma

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 5. training execution and results (RF case).

Before starting the training, the availability of the training file in the selected folder is checked. The file must be a CSV with data arranged by columns with the mandatory order: HH, HV, VH, VV, LIA, HLidar, BIOMASS (from in-situ). The first 6 columns are taken as inputs, the last column is taken as reference (target) for training.

**⚠️ when running the training, any existing ML configuration file whit the same filename and in the same path is overwritten, we suggest specifying different filenames when repeating the training or having a backup of these files.**

### Testing

If testflag is enabled, the code runs the testing, i.e. it applies the trained ML method to the input images stored in the selected input path. The images must be one for each polarization, identified by \_VV, \_VH, \_HV and \_HH flag in the filename immediately before the .tif extension, plus \_LIA for local incidence and \_HLidar for tree height.

The initial part of the filename must be the same for all the tif files referring to the same scene, as in the sample files delivered along with the code.

All the polarization and auxiliary images referring to the same scene must have the same dimensions, and they must be coregistered: missing files or errors in format and naming convention will cause error messages and stop execution. The process is iterated on all the images contained in the top-level input directory specified by *indir.*

First, the \_VV files are listed, then the filenames for the other polarizations and auxiliary data are created and the data are loaded and shaped to be ingested by the ML algorithm. Finally, the inversion is carried out and the output AGB maps generated by the algorithms are saved in the destination path. An example of console outputs during testing is shown in Figure 6.

Immagine che contiene testo, schermata, Carattere, viola

Il contenuto generato dall'IA potrebbe non essere corretto.

Figure 6. example of console outputs during testing

**⚠️ Depending on the size of input images, the process may take a while to complete. The complexity of ML algorithm architecture also affects the computation time.**

### Validation

In the last step of processing, the AGB estimated by the ML algorithm for each of the available ROIs are extracted from the output images and compared with the in-situ AGB measurements. The results in the form of scatterplots estimated vs. target AGB are saved in the output folder as png images. This part of processing also includes the generation of AGB map previews that are also saved as png files. An example of console outputs during validation execution is shown in Figure 7.

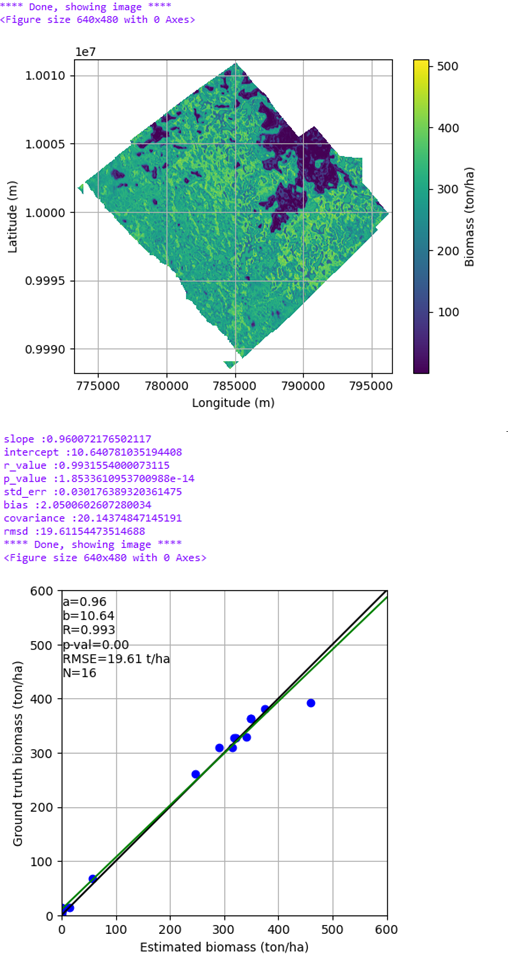


Figure 7. example of validation outputs

## End of execution

After completing the loop on all the images in the input path, execution ends, a message is shown at prompt along with the total processing time elapsed (see Figure 8).



Figure 8. End of execution message