Automatic Detection of Archaeological Features – ADAF ver. 0.0.1 Manual

By Nejc Čož*, Žiga Kokalj*, Ana Kostovska**, and Dragi Kocev**
*Research Centre of the Slovenian Academy of Sciences and Arts
**Bias Variance Labs d.o.o.
Contact: *nejc.coz@zrc-sazu.si*

When using the toolbox, please cite:

tba

Version

Version: 0.0.1, December 2023

For changes, see version history at the bottom.

General information

The tool for Automatic Detection of Archaeological Features (ADAF) has been developed to provide user-friendly software that uses machine learning models (in particular convolutional neural networks) to enable the automatic detection of archaeological features from airborne laser scanning (ALS) data. The software requires minimal interaction and no prior user knowledge of machine learning techniques, greatly improving its accessibility to the archaeological community. The underlying machine learning models have been trained on an extensive archive of ALS datasets in Ireland, labelled by experts with three types of archaeological features (enclosures, ringforts, barrows). The core components of the tool are the Relief Visualisation Toolbox (RVT) and the Artificial Intelligence Toolbox for Earth Observation (AiTLAS), both of which are actively used in the field of aerial archaeology. RVT is indispensable for processing input data (for training and inference) by converting digital elevation models into machine learning-friendly visualisations, while AiTLAS provides access to the machine learning models.

Online resource

Tba

Installation

The installation is currently only supported on Windows 64-bit machines. The application is compatible with machines equipped with CUDA-enabled graphics cards, but will also work on a standard CPU where GPU processing is not possible.

To install ADAF manually, you have to install Anaconda, create a virtual environment and install the requirements with pip following the next steps.

1. Install Miniconda. You can skip this step if Anaconda is already installed on your machine. To install it, execute the *Miniconda3-py38_23.10.0-1-Windows-x86_64.exe* located in the *installation* folder.

2. Run Anaconda Prompt (press Win key ( ) and type "anaconda prompt").
3. In the Anaconda Prompt, navigate to the **installation** folder by running command:
   ```
   cd <path-to-installation-folder>
   ```
4. Create and activate a conda environment. Run commands:
   ```
   conda create -n aitlas python=3.8
   conda activate aitlas
   ```
5. FOR CUDA ONLY – install CUDA compatible PyTorch version by running:
   ```
   torch_cuda_installation.bat
   ```
6. Install the packages using pip:
   ```
   pip install GDAL-3.4.3-cp38-cp38-win_amd64.whl
   pip install aitlas-0.0.1-py3-none-any.whl
   ```
7. Enable the use of the aitlas virtual environment in Jupyter notebooks by running:
   ```
   python -m ipykernel install --name aitlas
   ```
8. Close the Anaconda Prompt window. The installation is now complete.
9. Start ADAF by double-clicking the **ADAF.lnk** shortcut in the main ADAF folder.
10. Copy/paste the shortcut to any desired location (e.g. c:\Users\<user>\Desktop) to start ADAF from there.

Reinstalling or uninstalling the environment

1. Open Anaconda Prompt (press Win key ( ) and type "anaconda prompt")
2. Remove aitlas virtual environment by executing the following commands:
   ```
   conda env remove -n aitlas
   jupyter kernelspec uninstall aitlas
   ```
3. Delete the main ADAF folder. ADAF is now uninstalled.

To reinstall follow the installation steps from the start.

Running ADAF

Starting ADAF

Start ADAF by double clicking the shortcut called **ADAF.lnk**

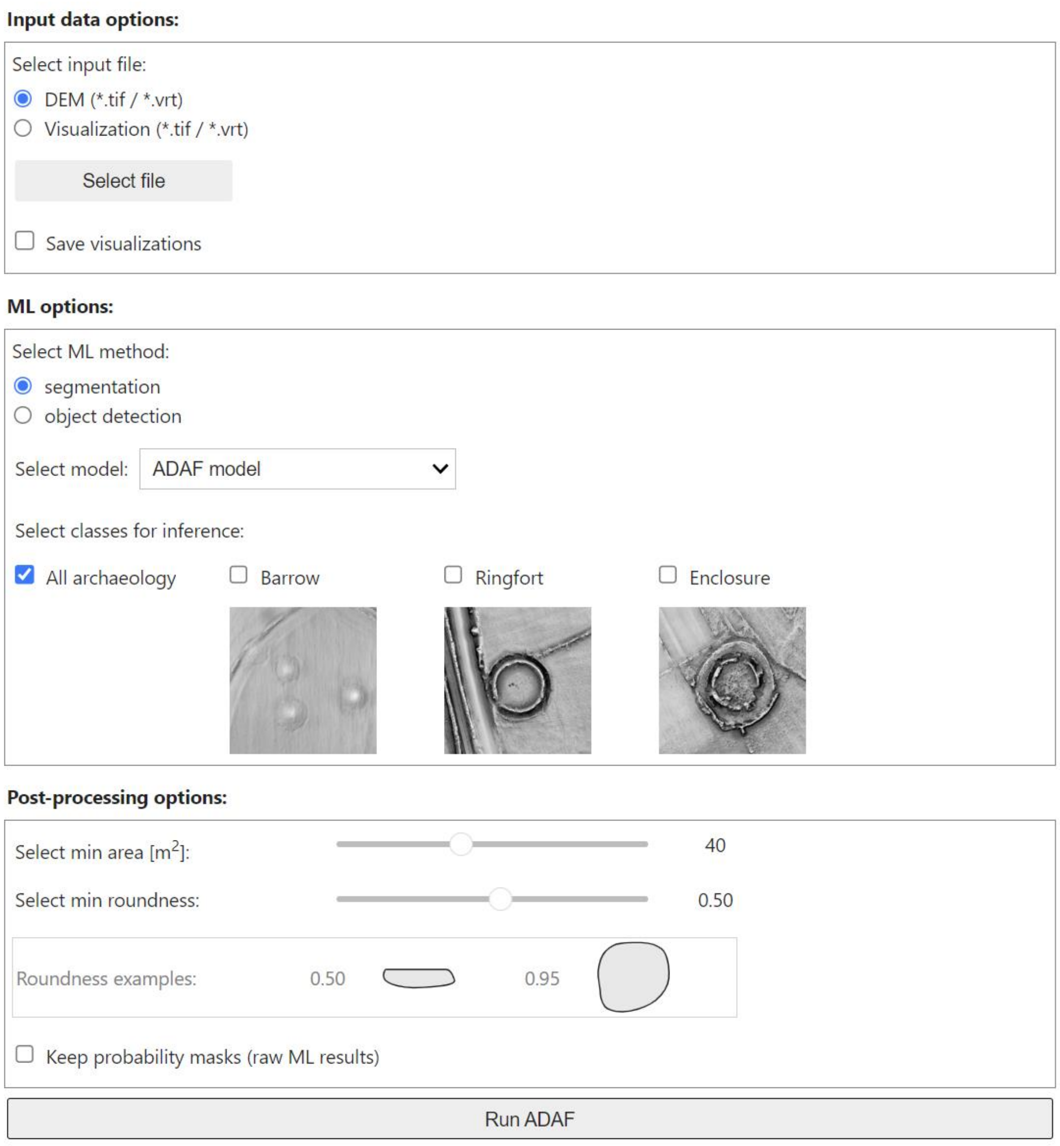
Starting ADAF manually*

** Users who are familiar with Python and/or Jupyter Notebooks can use ADAF notebooks used in their preferred way.*

- Open Anaconda Prompt (press Win key (⊞) and type “anaconda prompt”)
- Activate the aitlas environment
  ```
  conda activate aitlas
  ```
- Navigate to the location of ADAF
  ```
  cd <path-to-adaf-folder>
  ```
- Run command to open Jupyter notebook:
  ```
  jupyter notebook ADAF_main.ipynb
  ```

GUI

The graphical user interface (Figure 1) leverages interactive Jupyter Notebook widgets, allowing users to intuitively set parameters for running the provided ML models.



Input data options:

Select input file:

(●) DEM (*.tif / *.vrt)

( ) Visualization (*.tif / *.vrt)

Select file

☐ Save visualizations

ML options:

Select ML method:

(●) segmentation

( ) object detection

Select model: ADAF model

Select classes for inference:

☑ All archaeology ☐ Barrow ☐ Ringfort ☐ Enclosure

Post-processing options:

Select min area [m^2]: 40

Select min roundness: 0.50

Roundness examples: 0.50 0.95

☐ Keep probability masks (raw ML results)

Run ADAF

Figure 1: *ADAF graphical user interface.*

Input data options

Select input file

DEM – the user can enter any DEM in a geoTIFF (*.tif) or a GDAL virtual format (*.vrt) format. The app creates the visualisation and splits the image into smaller tiles. The size of the tiles is pre-set and has been selected to optimize processing performance.

Visualisation – the user can enter the visualisation that has already been created. This speeds up processing as the step of creating the visualisation is skipped. It is up to the user to ensure that the correct visualisation type is specified. For the built-in ADAF ML models, the required visualisation is the normalized SLRM (can be calculated externally with the RVT tool). For the parameters for creating the SLRM visualization, see section ADAF models.

Input file(s)

Select the input file(s) by clicking on the "Select file" button. The files are selected through a dialog window.

The app will allow to select any file in *.tif or *.vrt format. The best results will be achieved with input rasters of similar quality to the datasets used in training. The model is trained on digital feature models (DFM, i.e. DEM of terrain and buildings) with spatial resolution of 0.5m

Save visualisation

If checked, the visualisation files will be saved in the output folder as a virtual mosaic (*.vrt).

Machine learning options

Select ML method – segmentation or object detection

Select model – The two options that can be selected from the dropdown menu are *ADAF model* and *Custom model*.

ADAF model uses the default ML models that were trained specifically for the task of detecting the selected archaeological features in Ireland and the UK. The user has an option to run the detection for the following archaeological classes:

- Barrows
- Ringfort
- Enclosure
- All archaeology (all three from above are treated as a single class)

Custom model (not yet operational) will allow the user to run detection using a user-provided ML model. The ML model is loaded into the ADAF by providing the path to the *.tar file.

Post processing options

Select minimum area

Area [m^2] – remove detections smaller than the threshold value.

Select minimum roundness

Roundness [-] – archaeological features that ADAF has been trained to detect have a regular circle shaped footprint. Removing irregular shapes from the detected footprints can help to reduce the number of false positives.

The results can be filtered by minimum Roundness index, which is calculated as:

$roundness = 4\pi \cdot area / (convex\ perimeter)^2$

Roundness is expressed mathematically by dividing the area by the square of the perimeter of the polygon and normalising by multiplying the value by 4π. Shape index of a perfect circle is 1 and for a very elongated polygon it is close to zero.

Save raw predictions files

Keep temporary files that are created during processing (normally they are deleted after processing):

- Probability maps in GeoTIF format for *semantic segmentation* – a raster file where every pixel is labelled by probability score predicted by the ML model.

- Bounding box text files for *object detection* – a text file containing relative coordinates of the detected bounding boxes with probability score and metadata for georeferencing.

The following is a template for an individual result as outputted by object detection model:

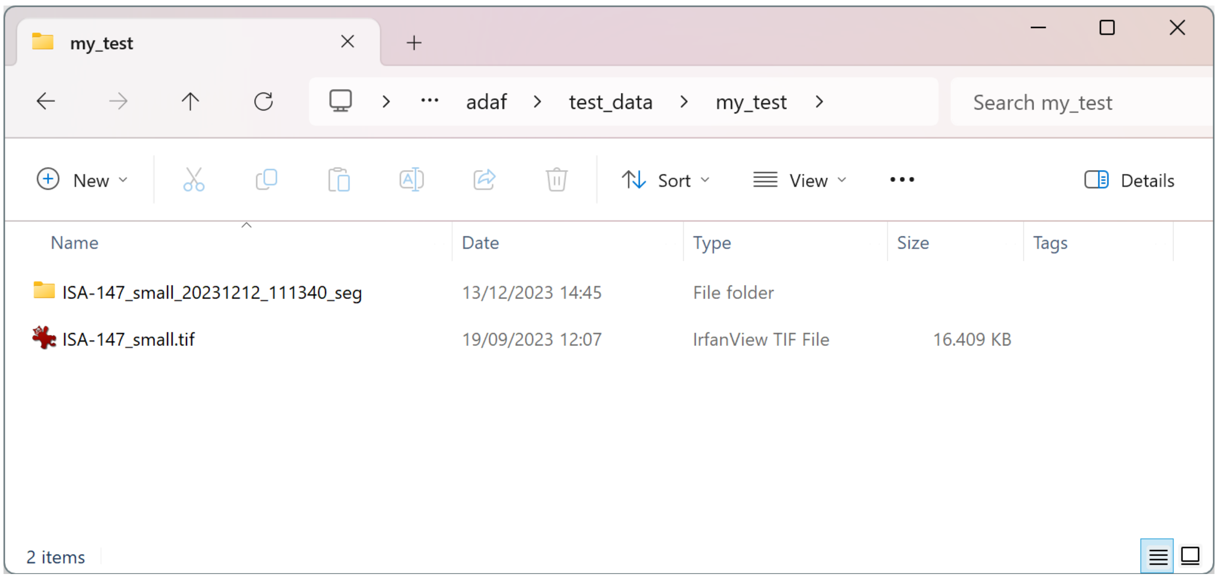
$x_1\, y_1\, x_2\, y_2\, x_3\, y_3\, x_4\, y_4$ <label> <probability score> <ESPG code> <resolution> <x min> <y max>

Where x_n y_n are pixel (relative) coordinates, label is the detected archaeological class, probability score is ML output and is self-explanatory, and the rest (EPSG, resolution, x min and y max) are image metadata that are needed to transform relative coordinates into georeferenced polygons.

Output

For each run, ADAF saves the results in a separate folder located in the same parent directory as the input raster file (**Figure 2**). The name of the output folder is a combination of the name of the input file, the timestamp and the ML model type (seg for semantic segmentation, obj for object detection).

The results folder always contains a log file and ML results in vector format (*.gpkg), which can be opened with most GIS tools. If more than one label has been selected for processing, the results are all saved in the same vector file with the "label" attribute to distinguish between the different detected classes. If the user has selected to save intermediate results, these will also be saved here.



Name	Date	Type	Size	Tags
ISA-147_small_20231212_111340_seg	13/12/2023 14:45	File folder		
ISA-147_small.tif	19/09/2023 12:07	IrfanView TIF File	16.409 KB	

Figure 2: ADAF output

ADAF models

The detection in ADAF employs 8 different ML models that were trained on an extensive dataset of ALS data with annotations for archaeological sites in Ireland, which include barrows, ringforts and enclosures.

We divided these models into two categories: four for object detection and four for semantic segmentation. For each type of archaeological site - barrows, enclosures, ringforts - there is a separate model. There is also a model that detects all archaeology and treats the three classes as one.

For the semantic segmentation models we use the HRNet architecture, which is available in the AiTLAS toolbox. These models are trained on data of quality levels 1 (excellent) and 2 (some triangulation; enclosing feature identifiable) as specified by the domain experts, and on patches containing only the specific objects for which the models are trained. The other two quality levels are 3 (complete or almost complete triangulation) and 4 (monument barely visible). For example, if we train a model for barrows, the training data only contains data about barrows with quality levels 1 and 2. For validation and testing, we extend the data to include annotation qualities 1, 2 and 4 and use patches with different object types. The validation data for semantic segmentation incorporates feedback from domain experts based on analysing the predictions of our previous models.

For the object detection models, we use the FasterRCNN architecture from the AiTLAS toolbox. These models are trained on quality level 1 and 2 data, as specified by the domain experts, and on patches containing only the specific objects for which the models were trained. For example, if we train a model for barrows, the training data only contains data about barrows with quality levels 1 and 2. For validation and testing, we broaden the data to include annotation qualities 1, 2 and 4 and use patches containing only the specific objects for which the models were trained. The validation data for object detection does not contain any feedback from domain experts.

Finally, in both learning scenarios (object detection and semantic segmentation), all our data is sourced from the SLRM visualization, using a patch size of 512 px. We perform data transformations, including random horizontal and vertical flips and rotations. The learning rate is set to 10^-4 for optimal results.

SLRM visualization

While the primary input data is DEM, the ML models were trained on simple local relief model (SLRM) visualizations. The SLRM images were created using the SLRM visualization function available in the RVT toolbox. The parameter for radius for trend assessment was set to 20 pixels for images with a spatial resolution of 0.5 m and normalized with a linear histogram stretch between -0.5 and 0.5 and normalised to values between 0 and 1.

Radius for trend assessment	10 m (20 px for 0.5m image)
Min/max normalisation	between -0.5 and 0.5

The same settings are used in ADAF when preparing the input data for detection with ML models.

Processing without a GUI

Follow the template in the ADAF_notebook.ipynb notebook.

License agreement

This software is distributed without any warranty and without even the implied warranty of merchantability or fitness for a particular purpose.

ADAF is open source. The code is available at GitHub (*tba*), with an Apache License 2.0.

Known issues

n/a

Version history

Version 0.0.1, December 2023

- Initial draft of the manual

References

Tba

~~~~~~

Copyright © 2023 Research Centre of the Slovenian Academy of Sciences and Arts (ZRC SAZU) and Bias Variance Labs.

12/12/2023
~~~~~~