Blackbird Analyzer scripts

This repository contains a set of Python-based scripts for Blackbird images analysis based on the original MATLAB scripts. The OpenCV library is used on some of the scripts found in this repository.

The trained models must be converted to ONNX in order to be able to be used with the scripts in this repository.

> Contact Dani Martinez for the model conversion.

> **NOTE:** The scripts were developed and tested on Python 3.11 for Windows 10/11. For Linux users, change "onnxruntime-directml" to "onnxruntime-gpu" in the **requirements.txt** file for CUDA acceleration.

Installation steps

Follow the following steps to install all required software required to correctly run the scripts in this repository.

1 - Install Python

Download Python 3.11 from the official webpage: https://www.python.org/downloads/windows/ .
Or click here to download it directly.

1. Start the installation by running the downloaded Python installer.
2. Check the "Add python.exe to PATH".
3. Click on "Install Now".

> **NOTE:** If you previously installed another Python version, you might need to uninstall it first or manually modify the PATH environmental variable to point to the correct Python version.

2 - Set up a Python virtual environment

A Python virtual environment is convenient in order to create isolated workspaces where you want to install specific Python packages that might need to fulfill specific version dependencies. This is very convenient to avoid conflicts between packages that require different versions of the same dependency. Therefore, it is recommende to create a Python virtual environment to use these scripts. In a nutshell, a Python virtual environment is a folder where all the packages are installed. Therefore, if something "breaks" this environment, the user can easily delete it by simply deleting the folder.

- Click on Windows search bar and start typing "powershell", then click on the "Windows Powershell" icon to start the command terminal.
- When the terminal opens, create a Python virtual environment by running the following command:
  `python -m venv C:\path\to\new\virtual\environment_name`
  - For example: `python -m venv C:\blackbird_env`
- Once created, activate the environment by running the following command:
  `C:\path\to\new\virtual\environment_name\Scripts\Activate.ps1`
  - For example: `C:\blackbird_env\Scripts\Activate.ps1`
- Once activated, the environment name will appear in parenthesis at the beginning of the prompt.

> **NOTE:** If you got an error when activating the environment, you might need to give execution permissions by calling this (only required in first time):

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

> **NOTE:** The created virtual enviroment will need to be activated every time you need to use the scripts from this repository.

3 - Install dependencies

The scripts found in this repository requires to install very specific Python packages as dependencies. These dependencies are listed in the "requirements.txt" file in the repository folder. The users can use this file to install them all at once by calling the following command: `pip install -r requirements.txt`

First of all, make sure to change your current directory in the Powershell terminal to the folder where the scripts are located by using the **"cd"** command.

- For example (Windows): `cd C:\blackbird-analyzer`

> **IMPORTANT:** Make sure you have activated the Python virtual environment. You should see the virtual environment name in parenthesis at the begining of the prompt bedore calling this command.

```
(blackbird_env) PS C:\blackbird-analyzer> pip install -r requirements.txt
```

Usage

Analyzer

The Blackbird analyzer script is a Python carbon-copy version or the original MATLAB "PM_analyzer.m" script with some added improvements on performance. Make sure to have the correct Python virtual environment activated before calling the script. The path to the CNN models used for the analysis must be passed explicitly when calling the script:`python analyzer.py <C:\path\to\model.onnx>`

For example:

```
(blackbird_env) PS C:\blackbird-analyzer> python analyzer.py models\PMnet.onnx
```

When the graphical user interface (GUI) is shown, the powershell terminal will remain blocked until the window is closed.

> **IMPORTANT:** In order to run the models on the GPU (DirectML), you need to have Windows 11 or Windows 10 with the latest updates installed in your system.

Thresholder

The thresholder script does not have a graphical interface and it is easily called by command line.

```
usage: thresholder.py [-h] [-o <OUT_XLSX>] [-lo <LOW_TH>] [-hi <LOW_TH>]
<RES_PATH>

Blackbird Results Thresholder

positional arguments:
  <RES_PATH>            Path to experiment results as a *.msgpack file.

options:
  -h, --help            show this help message and exit
  -o <OUT_XLSX>, --out <OUT_XLSX>
                        Path where the Excel file will be placed and its filename
(incl. .xlsx extension).
  -lo <LOW_TH>, --low <LOW_TH>
                        Specify lower threshold between 0 and 1
  -hi <HIGH_TH>, --high <HIGH_TH>
                        Specify higher threshold between 0 and 1
```

Example usage:

```
(blackbird_env) PS C:\blackbird-analyzer> python thresholder.py
D:\stacked\test_experiment\results.msgpack -lo 0.2 -hi 0.8 -o .\output.xlsx
```

This new tresholder script outputs different data compared to the original MATLAB script. For each timepoint and sample ID, there are three given values:

- Absolute number of sub-images labeled as infected (INF)
- Absolute number of sub-images labeled as clear (CLR)
- Absolute number of total analyzed sub-images (ALL)

> **NOTE:** Users are reponsible for calculating any Infection ratios and other statistical metrics by using this data.

Annex

This section contains technical information about the development of the methods found in this repository. **Users don't need to read this section.**

Exporting MATLAB models to ONNX

Download the ONNX Converter add-on for MATLAB. Run the **dag2onnx.m** script on MATLAB for the model conversion. Then, check and validate its integrity with Netron.

```
exportONNXNetwork(net2,"VitisNet.onnx",'BatchSize',1)
```

Compiling C++ code with Pybind11

NOTE: OpenCV C++ libs must be compiled dynamic/shared! Add opencv_world4xx.dll in the "cpp_functions" folder.

Start vscode from Developer Command Prompt (x64 Native Tools Command Prompt for VS 2022) Go to project dir and run:

```
code .
```

Edit CMakeLists.txt file and set correct paths to "find_package". To find out PyBind11 include paths, run:

```
python -m pybind11 --includes
```

Use CMAKE

```
mkdir build
cd build
cmake ..
cmake --build . --config Release --target leaf_masking
cmake --install .
```

Observations

- ONNXRuntime-CUDA package is not supported on Windows. Instead, install the ONNXRuntime-DirectML package for GPU inference.