**pyCMAC V11L USER GUIDE**

**Windows Setup and Processing**

**Overview**

This document covers the process required to set up a Python environment for running the CMAC V1.1 demo application “pyCMAC\_V11L”, which is used to process Landsat 8/9 image data. The environment is a conda-based “virtual” environment that can be created through Miniconda, a “slimmed down” version of the Anaconda software package and environment management suite. The setup procedure described here was performed on an HP Omen laptop PC (32 MB system RAM) running MS Windows 10 64-bit home edition. This can also be installed on a Linux-based system or WSL session running under Windows; additional details on this installation path are presented in Appendix A. Alternatively, pyCMAC\_V11L can be run in an existing conda environment that has the appropriate supporting software libraries installed.

* If conda is not currently installed on your system: in a Windows web browser window (e.g. Microsoft Edge, Mozilla Firefox, etc.), go to:
  + <http://docs.conda.io/en/latest/miniconda.html>.
  + In the “Latest Miniconda Installer Links” section, look for and click on the link “**Miniconda3 Windows 64-bit**” to download the 64-bit Windows Miniconda installer (Figure 1). Follow the instructions to complete the installation.

|  |
| --- |
| Graphical user interface, text, application, email  Description automatically generated |
| **Figure 1. Windows 64-bit Miniconda Installer Link** |

* **Optional** **Step**: In the lower left-hand corner of the host’s display screen, go to the “Start” window and left click. Select “Anaconda Prompt (miniconda3).” Select “More” from the popup menu, and select “Pin to taskbar”. This will allow easy access to a conda command line window where conda commands can be run. This step can be skipped if desired.
* Bring up a conda command window (Figure 2). If Step 2 was not performed, the first two actions in the description can be used to access the window.

|  |
| --- |
| **Shape, rectangle  Description automatically generated** |
| **Figure 2. “conda” Command Line Window. The “(base)” to the left of the command prompt indicates that the base (or system-level) conda environment is active.** |

* Check for updates to the conda application by entering **conda update conda** at the command prompt (Figure 3).
  + Depending on when conda was installed, there will likely be new and/or updated files to download and install that will bring conda to the latest version. Following the list of files, a [y/n] prompt will be shown. Entering **y** will initiate download and installation of the listed files.

|  |
| --- |
| Text  Description automatically generated |
| **Figure 3. Conda Update** |

* Create a separate “virtual” environment to run the pyCMAC\_V11L application (**it is highly recommended to do this for any python-based project to avoid potential clashes between project-specific python/supporting library installs and system-wide python/supporting library installs**). To do this, enter **conda create –name <environment\_name>** at the command prompt (Figure 4). Once the environment is created, enter **conda activate <environment\_name>** at the command prompt to change into the new environment. Remember: ALL supporting package installations and running the pyCMAC\_V11L application must be done from within this environment!

|  |
| --- |
| Text  Description automatically generated |
| **Figure 4. Create Conda “virtual” environment cmacv11\_env, and switch Into the new environment.** |

* In the “cmacv11\_env” virtual environment, install the library packages required to support pyCMAC\_V11L operation. These are:
  + lxml (an XML parser)
  + BeautifulSoup (library to simplify data extraction from tree data structures)
  + OpenCV (for specific filtering operations that pyCMAC\_V11L performs)
  + GDAL (for image IO)

Other library packages required for pyCMAC\_V11L operation (e.g. numpy), are/should be covered under the GDAL and/or OpenCV installations. The latest, “stable” versions of packages are installed by entering **conda install <library module name>** at the conda command prompt. Installation of the BeautifulSoup package is shown in Figure 5; enter **conda install beautifulsoup4** at the command prompt. The lxml, OpenCV and GDAL packages are installed with the same command (just change the name of the modules to **“lxml”**, “**opencv**” and “**gdal**”, respectively). If necessary, specific versions of packages can be installed by entering **conda install <module\_name>=version;** for additional information on version-specific package installation, please refer to <https://docs.anaconda.com/anaconda/user-guide/tasks/install-packages/>.

Depending on the order in which packages are installed, conda may flag some previously installed packages for “downgrade” (i.e. installation of an earlier version) in order to allow compatible installation of a new package. It shouldn’t happen very often.

|  |
| --- |
|  |
| **Figure 5. Install the Beautiful Soup (beautifulsoup4) module in the cmacv11\_env virtual environment. Installation of the lxml, OpenCV and GDAL modules uses the same “conda install” command. Note installation of the Python interpreter version 3.11.2; this is done because no Python installation was specified when the environment was first created.** |

* Test the installation of each package by trying to “import” it. This can be done for each package using the following steps:
  + Enter **python** at the command prompt. This launches the Python interpreter installed in the virtual environment.
  + To test “**BeautifulSoup”** installation, enter **import bs4** at the Python command prompt. If no error messages appear, installation of BeautifulSoup was successful.
  + To test “**lxml”** installation, enter **import lxml** at the Python command prompt. Again, no error messages indicate a successful installation.
  + To test “**opencv”** installation, enter **import cv2** at the Python command prompt.
  + To test **“gdal”** installation, enter **from osgeo import gdal** at the Python command prompt.
  + End the Python interpreter session by entering **quit()** at the Python command prompt.
  + If any package fails to import properly, try to remove it by entering **conda uninstall <package\_name>** at the Windows command prompt, then **conda install <package\_name>**. It’s possible some previously installed packages may be downgraded during the uninstall process. When package re-installation is completed, retest it using the appropriate import command after restarting the Python interpreter.

**Running the pyCMAC\_V11L Application**

* “Install” the pyCMAC\_V11L source files and configuration file to the local filesystem. For ease of use, it is recommended to keep the configuration file in the same location as the source files.
* In a conda command window prompt, enter **conda activate <name of pyCMAC\_V11L environment>.** The environment should switch from the “base” system-level environment to the local virtual environment (Figure 6).

|  |
| --- |
| Text  Description automatically generated |
| **Figure 6. Switch to the virtual environment set up to run pyCMAC\_V11L.** |

* In Notepad++ or other Windows text editor, change the configuration file **</path/to/pyCMAC\_V11L config file>/pyCMAC\_V11L\_cfg**.**json** entries for **<toaRootDrive>** and **<cmacRootDrive>** to point to the core locations in your filesystem where the input and output data files are to be placed. An example configuration file used to test the procedures described in this document is shown in Figure 7.
  + The <**additionalPathInfo**> entry allows additional specification of the final input location. **The (sub)path specified here must already exist for the input TOA image data**. Note that this path information will also be used to create the CMAC output path if it doesn’t already exist.
  + The **<productID>** entry should be changed to the product ID of the Landsat 8/9 image to be processed.
  + The <**nodataValue**> entry represents the value of a ‘nodata’ pixel. To better accommodate the 16-bit unsigned integer range of current L8/9 image data, this value is set to a default value of 0. Note this could impact image display.
  + The <**atmiGridcellDim**> entry sets the size of the square gridcells in which the Atm-I levels are estimated. The smaller the value, the “finer” the granularity/resolution of the resulting Atm-I image. For Landsat 8/9 images, the default value of 10 results in a gridcell size of 10x10 pixels, or 300m x 300m in “physical” area on the Earth’s surface. Values less than 10 are typically not recommended for use, as there may be an insufficient number of pixel samples to generate the image statistics required for Atm-I estimation.
  + The <**medianFilterRadius**> and <**gaussianFilterRadius**> entries set the size of median and Gaussian smoothing filters that can be applied to the “raw” Atm-I image**.** The associated <**medianFilterOn**> and <**gaussianFilterOn**> entries control whether these filtering steps are enabled (1) or disabled (0). By default, these filtering steps are enabled using the specified filter radius values of 3 and 7, respectively.
  + The <**atmiCapLevel**> entry sets a maximum “cap” on the correctable Atm-I level (i.e. Atm-I levels higher than this level are set to this level). The <**atmiCapOn**> entry controls whether capping is enabled (1) or disabled (0).
  + The <**option1Folder**> and <**option2Folder**> entries allow specification of up to two additional levels to the final CMAC output folder location. This enables better segregation of outputs due to changes in parameter values, etc.
  + **Do not change the <fileExt> entry**. Landsat 8/9 image data files are currently in GeoTIFF format, hence the “tif” extension.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |
| **Figure 7. Notepad++ session editing pyCMAC\_V11L configuration file (any Windows-based text editor should work as well). The <toaRootDrive> and <cmacRootDrive> entries identify the top-level locations in the filesystem. pyCMAC\_V11L uses the <additionalPathInfo> and <productID> entries to further establish final input and output file locations. The <productID> entry is the name of the image data folder, as obtained from online download.** |

* At the command prompt, enter **python </path/to/pyCMAC\_V11L Source Files>/pyCMAC\_V11L.py** **</path/to/pyCMAC\_V11L config file>/pyCMAC\_V11L\_cfg.json** (with spaces after “python” and the main python script filename)**.** As shown in Figure 8, if the configuration information is specified correctly, the program should launch. The program has been successfully launched when processing status messages are scrolled on-screen. Upon successful completion of the program, total processing time in seconds is displayed and a processing report is created in the CMAC output folder. This report contains summary information on the observed Atm-I levels across the image and band-specific counts of pixels exhibiting correction error (e.g. correcting to a scaled negative surface reflectance value, a scaled surface reflectance value less than 1 DN, or a scaled surface reflectance value greater than 10000 DN). In this particular context, “scaled reflectance” refers to a DN pixel value equal to the surface reflectance (a unitless number between 0 and 1) multiplied by 10000.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |
| **Figure 8. Running pyCMAC\_V11L in an existing Anaconda-based conda environment. Here </path/to/pyCMAC\_V11L/pyCMAC\_V11L\_Source Files> refers to the pyCMAC\_V11L folder in the local filesystem. As recommended, </path/to/pyCMAC\_V11L/pyCMAC\_V11L config file> refers to the same folder location. Since the configuration was set up properly, the program has started execution, and processing status messages are displayed on-screen.** |

**APPENDIX A**

**Linux / WSL Linux Setup and Processing**

This appendix presents a procedure to create a virtual Python environment in a “native” Linux system or WSL session running on MS Windows. As with the Windows installation, a conda virtual environment based on Miniconda 3 is considered. If running in a native Linux or WSL command window previously set up on your system, the first 4 steps should be skipped.

* Bring up a cmd prompt window by right-clicking on the window in the lower-left corner of the screen, selecting the “Run” menu option, and entering **cmd** in the Open text box (Figure )

|  |
| --- |
| Graphical user interface, text, application  Description automatically generated |
| **Figure 9. Bring up a Windows command line window.** |

Right-click and select Run in the menu.

* If WSL is not currently installed and running on your system, enter **wsl –install -d Ubuntu** at the Windows command prompt**.** All the required Windows and Ubuntu components will be downloaded and installed. Any distribution can be specified; for the purposes of this installation procedure, Ubuntu was chosen as the Linux distribution. If a message upon completion of the installation requests a system reboot, go ahead and reboot; this will be needed to establish all the changes made to the system (e.g. new processes to run at startup, etc.).
  + If WSL has already been installed and set up (including having an assigned Linux username and password), simply enter **wsl** at the Windows command prompt. The next step can be skipped.
* Once the WSL main installation and Linux setup has completed, you will be asked to enter a Linux username and password. Do so; it’s recommended that users have a separate account to access the system as needed.
* After setting up your username and password, check to see if any Linux components need to be updated. Do this by entering **sudo apt update** at the Linux command prompt, entering your Linux password, then entering **sudo apt upgrade** at the command prompt. The **sudo** part of the update and upgrade commands provides temporary root-level credentials without having to log in as root; this allows sensitive administrative tasks to be done by “trusted” non-root users. Once the list of component updates has been generated, a yes/no prompt will appear requesting permission to download and install the updates. Enter **y** (or yes) at the command prompt to enable updating.
* At the command prompt, enter **cd ~** to switch to your home Linux directory (Figure 10).

|  |
| --- |
| Text  Description automatically generated |
| **Figure 10. Switch to your Linux home directory. As with the Windows installation described earlier, (base) indicates a system-level conda environment is active. The following “pwd” command displays the current (or “working”) directory.** |

* In your home directory, get the Linux Miniconda installer “Miniconda3 Linux 64-bit” from <https://docs.conda.io/en/latest/minconda.html>. If running Linux in a WSL session, you can do this by downloading the file in Windows and using Windows’ File Manager to copy the file from “Downloads” to your Linux home directory (accessed by clicking on the Linux penguin icon at the bottom of the File Manager filesystem display pane, clicking on the “Ubuntu” folder, clicking on the “home” folder, and finally clicking on your particular user home directory). Alternatively, you can download the file directly to your Linux home directory by entering the command **wget** [**https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh**](https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh)at the Linux command prompt (Figure 11). If running on a “native” Linux host, you’ll likely have to use **wget** retrieval.

|  |
| --- |
| Graphical user interface, text  Description automatically generated |
| **Figure 11. Retrieving Linux Miniconda installer in a native Linux or WSL session using ‘wget’ command.** |

* At the Linux command prompt, enter **bash Miniconda3-latest-Linux-x86\_64.sh** to run the installation script. Additional details on installing Miniconda on Linux systems can be found in the web page accessed by the link contained in the ancillary text file **“link\_to\_linux\_miniconda\_installation\_procedure.txt”**, located in the PyCMAC\_V11L main folder.
* Once conda has been installed, check to see if it needs updating by entering **conda update conda** at the Linux command prompt. This step is identical to the step for upgrading the Windows version of conda.
* Enter **conda create –name cmacv11\_env** at the miniconda command prompt to create a virtual environment. This step is identical to the step creating the virtual environment in the Windows setup procedure. When created switch to this environment by entering **conda activate cmacv11\_env**.
* Install the **“beautifulsoup4”**, **“lxml”**, and **“gdal”** packages as described earlier by entering **conda install <package\_name>** at the command prompt. Test each installation following the same procedure as described earlier (i.e. start a python session by entering **python** at the command prompt, then importing the module with the import command as described).
  + Installation of the **“opencv”** package is slightly different than with the corresponding Windows installation. The use of just **conda install opencv-python** resulted in a “complete” installation; however, running the import test **import cv2** in the Python interpreter failed due to a missing graphics-related component package. Entering **conda install opencv-python-headless** resulted in a successful install and import test.
* In a Linux text editor (e.g. vi/vim, cmacs, etc.), prepare the configuration file **<path/to/pyCMAC\_V11L\_config\_file>/pyCMAC\_V11L\_cfg.json** as described earlier. Make sure the **<toaRootDrive>** and **<cmacRootDrive>** entries point to locations in the Linux filesystem (e.g. /home, etc).
* Run pyCMAC\_V11L by entering **python <path/to/pyCMAC\_V11L\_Source\_Files>/pyCMAC\_V11L.py <path/to/pyCMAC\_V11L\_config\_file>/pyCMAC\_V11L\_cfg.json**