G5

PROJECT USER MANUAL

Unet Convolutional Neural Network (UNET) Pipeline

Aerial Imagery Initiative

Contents

Flood Extent Extraction NoteBook User Manual

Unet Convolutional Neural Network (Unet CNN) method

Introduction

This manual describes the deployment and usage of the Flood Extent Extraction (FEE) notebooks developed for the Department Of Customer Services (DCS). This manual contains guides on how to download and execute said Notebooks within the Amazon Web Services AWS Sagemaker Environment. This manual will also cover the expected input data type and outputs that are configured.

Assumptions

Because of the nature of the FEE system the following assumptions have been made about the user as well as the ability to access the storage systems (bitbucket, AWS S3) used to host the data and executable Notebooks. While many steps are covered some details may be missing such as logging into AWS / Bitbucket, navigating AWS / Bitbucket. etc

- User has access to an AWS account
- User has access to an AWS S3 bucket containing flood imagery for analysis
- User understands how to start / stop Sagemaker Notebooks
- User understands how to open Sagemaker JupyterLab Notebook
- User has access to Bitbucket repository hosting required notebooks
- User understands cloning of Bitbucket repositories
- User is computer literate and has some programming experience (ideally python)

Overview

This manual will cover the following components in higher detail.

Image / Data inputs

Flood data to be processed is expected to be JPEG2000 format images with Near Infra-Red Red Green (NRG) colour channels in the respective Channel 0 = Near Infra-Red, Channel 1 = Red, and Channel 2 = Green. It is preferable that images are square with equal height and width dimensions. This allows for accurate rescaling of images that are then later divided into patches for processing by the Unet model. Images of size 10000x10000 and 6250x6250 have been tested successfully.

Images need to have geodetic information embedded in the images metadata, specifically pixel size, projection datum, image location in relation to datum. Often these images are exported from GIS systems that already contain the required geodetic information in the image metadata.

Environment

The FEE has been developed and tested with the AWS Sagemaker environment. The Sagemaker environment is primarily a preconfigured Conda and python environment in AWS. The execution Environment stack is as follows:

- AWS Sagemaker
- AWS Instance (Linux based host)
- Python / Conda Environment
- Executable Jupyter Notebook

FEE Notebook execution steps

The complete start to stop process for operating the FEE can be described in the following steps:

1. Starting a correctly specified Sagemaker Notebook
2. Cloning the FEE Bitbucket repository
3. Configuring the Notebook
4. Executing the Notebook
5. Examine the output
6. Shut down the Notebook instance

Perquisites

- Access to the Bitbucket repository containing the required Notebook: https://bitbucket.org/csu-spatialservices/flood-extent-extraction.git
- Authorised access to AWS and AWS Sagemaker
- Compatible JPEG2000 Images hosted on AWS S3 and access to said images from same AWS Sagemaker instance

Sagemaker Notebook

Notebook instance type

A Sagemaker Notebook will need to be started within the AWS environment. The Notebook Instance type best suited to the Unet CNN process is an Accelerated Compute instance. This is because CNN is able to utilize the CUDA capabilities to perform classification quicker.

The suggested Sagemaker Notebook instances are as follows:

ml.p2.xlarge (Accelerated Compute)
Tested on tiled flood regions of 176 tiles @ 6250 pixels x 6250 pixels

Volume size

It is recommended to configure the instance to have at least **10GB of local storage**. It is unlikely that all of this will be used however it does provide storage in case of larger datasets.

Sagemaker Notebook Example

Instance suitable for large single images.

Notebook instance name	UNetNoteBook
Notebook instance type	ml.p2.xlarge
Lifecycle configuration	None
Volume size	10gb

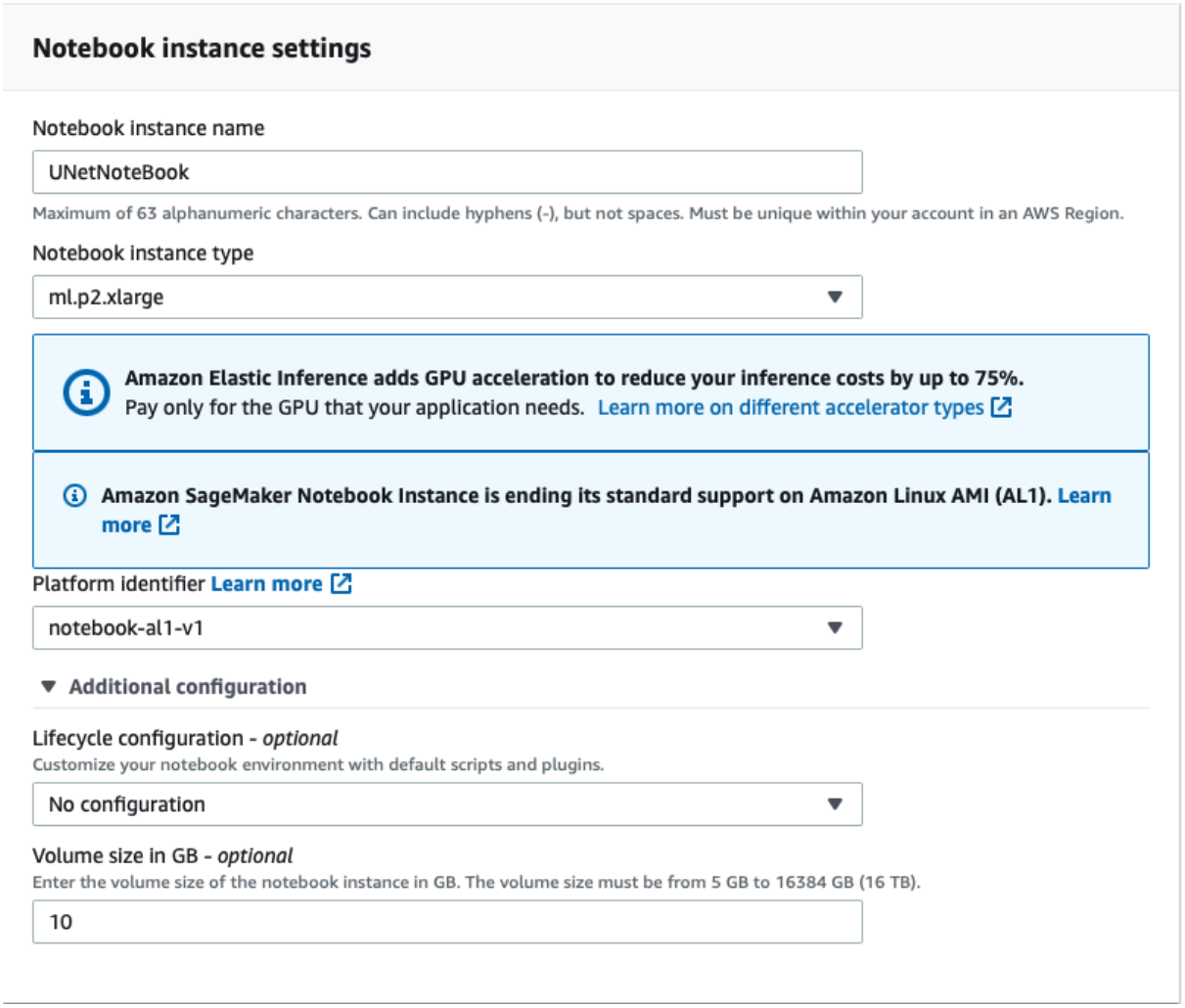

Figure 1 - Example Sagemaker instance

Clone the repository

All functional notebook code is held within the following repository within Bitbucket at the following repository URL:
https://bitbucket.org/csu-spatialservices/flood-extent-extraction.git

This repository will need to be cloned to the Sagemaker Notebook / Instance. The easiest method is to use the inbuilt Notebook Git tools access from the left menu Git icon and selecting “Clone a repository”. Once prompted enter the repository URL and fill in your Bitbucket credentials to access the repository.

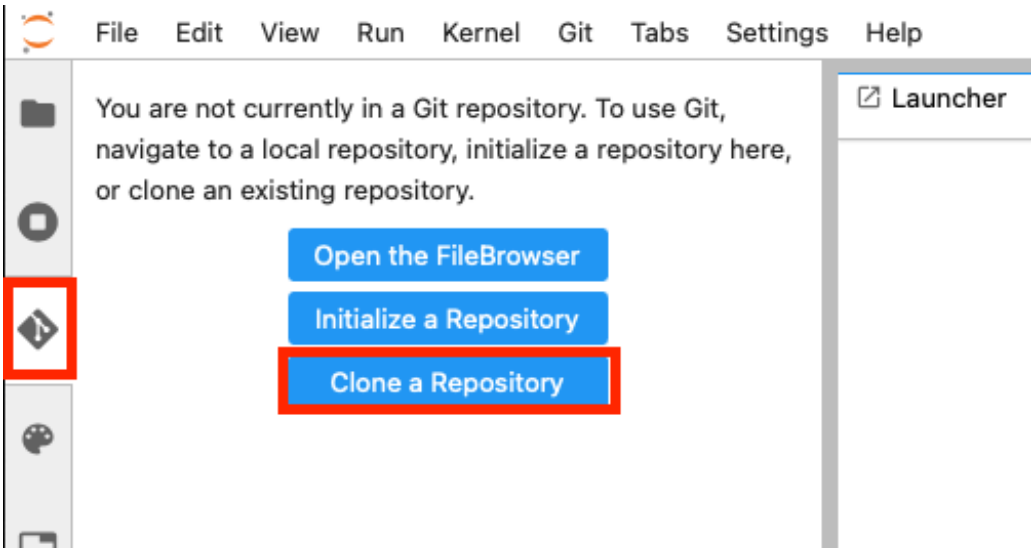

Figure 2 - Accessing Git clone button

Repository Folder Structure

The repository will have the below folder structure. Most folders are empty and are intended to be default locations for runtime data (Images = Flood images, Logs = Runtime Logs of the notebook, Out = Shapefile outputs. etc). These will be the preconfigured locations used by the notebook during execution

```
flood-extent-extraction/
├── App/
│   ├── GMM Extraction Method/
├── Documentation/
├── Images/
├── Logs/
├── Models/
│   ├── GMM/
├── Out/
```

The 3 directories of most interest will be:
App – Contains executable Notebooks. For this document we will be looking at GMM Extraction Method
Documentation – Documents such as this
Models – Machine learning models and artifacts

Notebook Setup and Execution

The Unet CNN technique notebook can be found at **flood-extent-extraction/App/Unet Extraction Method/Unet_Flood_Extent_Extraction.ipynb**. The default settings should be suitable for the execution of the notebook. However they should be checked to prevent errors during runtime.

Jupyter Kernel

Select **conda_amazonei_tensorflow2_p36** as the kernel for the Jupyter Notebook.

Python modules

The conda environment will also need additional modules for the notebook to run. This is completed by executing the last cell in **Imports and system setup.** This process can take some time (~25mins).

```
!pip install patchify
!pip install 'opencv-python>=4.5.3.56' # required as older versions fail opening some jp2 images
!conda install -y -c conda-forge gdal fiona rasterio # this will take a long time ~25mins
```

Figure 3 - Cell containing module install commands

Notebook settings

All user settings are contained in 1 cell “Notebook settings”. Below is an explanation of each setting. A brief explanation of each explanation is also contained in the Notebook.

The only setting that is likely to be changed is ***nrg_image_s3_source*** as it specifies the source S3 bucket of flood images.

- Input Image settings
    - **nrg_image_s3_source** - String: Amazon S3 source directory containing flood images destined for processing eg. s3://ss-csu-dataset/raw/Brewarrina_Flood_2021_04_15cm_NRG/
    - **nrg_image_storage_directory** - String: Storage directory where flood images will be stored for processing.
    - **image_scale** - Int: Image size to scale image to during preprocessing. patch_size needs to evenly divide into this (eg and image 1024x1024 will get 4 patches at 512x512). Default value is: 1024
    - **patch_size** - Int: Image size that the Unet model is expecting. After the image is scaled to image_scale. Preprocessing will patchify images into patch_size.
    - **ignore_nonsquare_images** Boolean: Notebook will ignore and not process non square images. Otherwise non square images will be rescaled to image_scale. Scaling non square images can cause unexpected results.
- Unet model settings
    - **unet_model** - String: Directory containing the pre trained Unet model
- Contouring / polygon settings
    - **minimum_contour_size** - Int: Minimum pixel area for a contour / polygon to be considered valid
- Logging settings
    - **log_file_prefix** - String: Prefix that will be attached to all log files.
    - **log_storage_directory** - String: Location to store log files
- Output settings
    - **raster_shp_output_prefix** - String: Prefix for output shape and raster files. Output files names will also include date and time the file was created. eg "brewarrina_20211010_07-08-01.jp2"
    - **output_directory** - String: Location to store shape files, inspection and zip file.

Execution

Once ***nrg_image_s3_source*** has been set to the source S3 bucket for the flood images the Notebook can be executed (Run -> Run All Cells).

Zip file output

After execution has successfully completed the ***output_directory*** will be zipped and saved to the parent directory of ***output_directory***. Eg "../***output_directory***"

Runtime outputs

Logging

During runtime the notebook will update the user via output below the executing cell. This output is limited and provides an overview of the Notebook state. A more detailed log is generated and output to the folder defined in ***log_storage_directory***, this is set to the *Logs* folder by default. An additional EXTRA logfile is also generated, this EXTRA log file contains logs of the Notebook and all imported modules. This can be used to get an in depth troubleshooting of the notebook.

Raster and Shapefiles

Once the notebook has been successfully executed the flood extent will be output in the following formats:

JP2 Raster - A JP2 grayscale raster file is generated with 2 pixel values:

1 = Flood area

0 = Non Flood area

These images will contain geodetic information and will render correctly in GIS software. Opening in the image in other image software may produce a blank image as the pixel values may be rendered on a scale of 0-255.

ShapeFile - Flood extent will also be output in an ESRI ShapeFile.

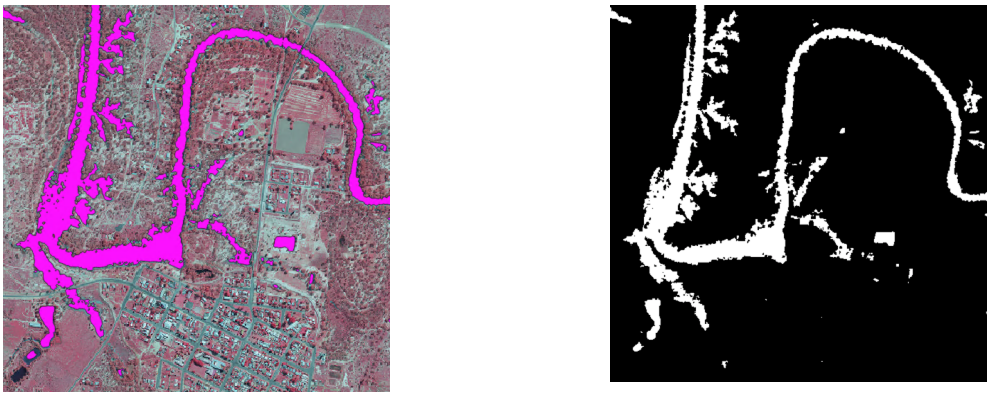

Figure 3 - Example shape file (left) and raster file (right)

Clean up

At the end of the Notebook execution the system will automatically delete flood images and zip and save the ***output_directory.*** This is to ensure multiple runs of the notebook does not cause overfilling the local disk space and to ensure no old images are accidently processed when attempting to process new images.

The zipped outputs will be saved to the parent directory of ***output_directory.***

Note: Remember to Stop the Notebook after Execution to prevent a large AWS bill