TerrAquaUAV

Abhijeet Rajput

March 2025

1 Overview of the Dataset

This document provides an analysis of the metadata extracted from a GeoTIFF file.

1.1 File Format

The raster file is stored in the GeoTIFF format, which is widely used for storing geospatial raster data along with metadata that defines its spatial reference.

1.2 Data Type

The raster contains values of type float64, which indicates that the data represents continuous numerical values. This suggests that the dataset might be related to environmental variables such as elevation, temperature, or probability distributions.

1.3 Spatial Dimensions

The dataset consists of a grid of:

- Width: 79 pixels
- Height: 73 pixels
- Total Pixels: $79 \times 73 = 5,767$ pixels

Since there is only one band, the raster contains a single layer of information.

2 Libraries Used and Model Preprocessing

2.1 Libraries Used

The script imports the following libraries:

- **rasterio**
- **numpy**
- **cv2 (OpenCV)**
- **matplotlib.pyplot**
- **tensorflow**

2.2 Data Preprocessing and Normalization

- **Handling NoData Values:** If the dataset contains missing values, they should be masked before normalization. As the nodata values are zero and there were no any missing values so there was no any need of Data cleaning.
- **Min-Max scaling:** The script normalizes the raster values using min-max scaling: This ensures the pixel values are scaled within the range [0, 1].

3 Patch-Based Processing for CNNs:

3.1 Function: `extract_patches(image, patch_size, stride)`

Purpose: This function extracts overlapping patches from an image, which can be used for training a super-resolution model.

3.1.1 Steps:

1. Take an input image of dimensions $H \times W$.
2. Define the `patch_size` and `stride`.
3. Iterate over the image with a sliding window approach:
    - Extract patches of size `patch_size` $\times$ `patch_size`.
    - Move by `stride` pixels to extract overlapping patches.
4. Store all extracted patches in an array and return them.

:

3.1.2 High-Resolution Patch Extraction

After defining the patch size = 24 and stride = 1, the function is used to extract HR patches from normalized image data:

3.1.3 Explanation of Code

- The extracted 2800 patches are stored in an array. - The function extracts all possible patches by sliding across the image with a given stride.

3.2 Function: `create_lr_hr_pairs(hr_patches, scale=2)`

Purpose: To create degraded Low-Resolution (LR) patches from High-Resolution (HR) patches for training a super-resolution model.

3.2.1 Steps:

1. Take an HR patch.
2. Downscale it using bicubic interpolation.
3. Upscale it back to the original size.
4. Return both LR and HR patches.

This function takes in HR patches, processes them to simulate degraded LR patches, and returns paired LR-HR data suitable for training super-resolution models.

4 SRCNN Developement

4.0.1 SRCNN Architecture

It is a 3 x 3 layered SRCNN architecture. The first layer consists of a Conv2d operation with 64 filters and a kernel size of 9 x 9. The second layer consists of 32 filters with a kernel size of 5 x 5, and the last layer has 1 filter with a kernel size of 5 x 5. SRCNN Architecture

Layer	**Operation**	**Kernel Size**	**Number of Filters**	**Activation Function**
Conv1	Feature Extraction	9×9	64	ReLU
Conv2	Non-Linear Mapping	5×5	32	ReLU
Conv3	Reconstruction	5×5	1	Linear

4.1 Hyperparamter Tuning Results

Configuration	**(patchsize,stride)**	**Filters**	**MAE**	**SSIM**	**PSNR**
1	(16,4)	(64,32,1)	0.0413	0.8473	25.05 db
2	(32,4)	(64,32,1)	0.0416	0.847	24.98dB
3	(32,4)	(128,64,1)	0.0411	0.8497	25.08 db
4	(32,4)	(256,128,1)	0.0415	0.8512	25.04 db
5	(24,1)	(256,128,1)	0.0280	0.9362	28.19 db
6	(24,1)	(512,256,1)	0.0254	0.9452	28.84 db

4.2 Selection of the Best Model and Final evaluation

1. Activation Function: ReLU
2. patchsize = 24 ,stride = 1 ,fiters = (512,256,1)

The selected model was trained for 200 epochs and optimized with adamw = 0.00001