GNR 607
Programming Assignment Presentation

Team Members:

Aminur Hossain (ID: 24D1384)

Amartya Ray (ID: 24D1383)

Problem Statement

Problem No 22

Objective:

Given a multiband image of N bands, compute principal components and generate the approximate version of the input image by performing inverse principal component transform using 2, 3, ..., N-1 components.

Input Data:

- **Dataset:** 1 subset image taken from a Landsat-8 satellite full scene image
- **Bands:** 7 bands (for the Mumbai scene)

  LC08_L1TP_148047_20180423_20180502_01_T1.tar.gz

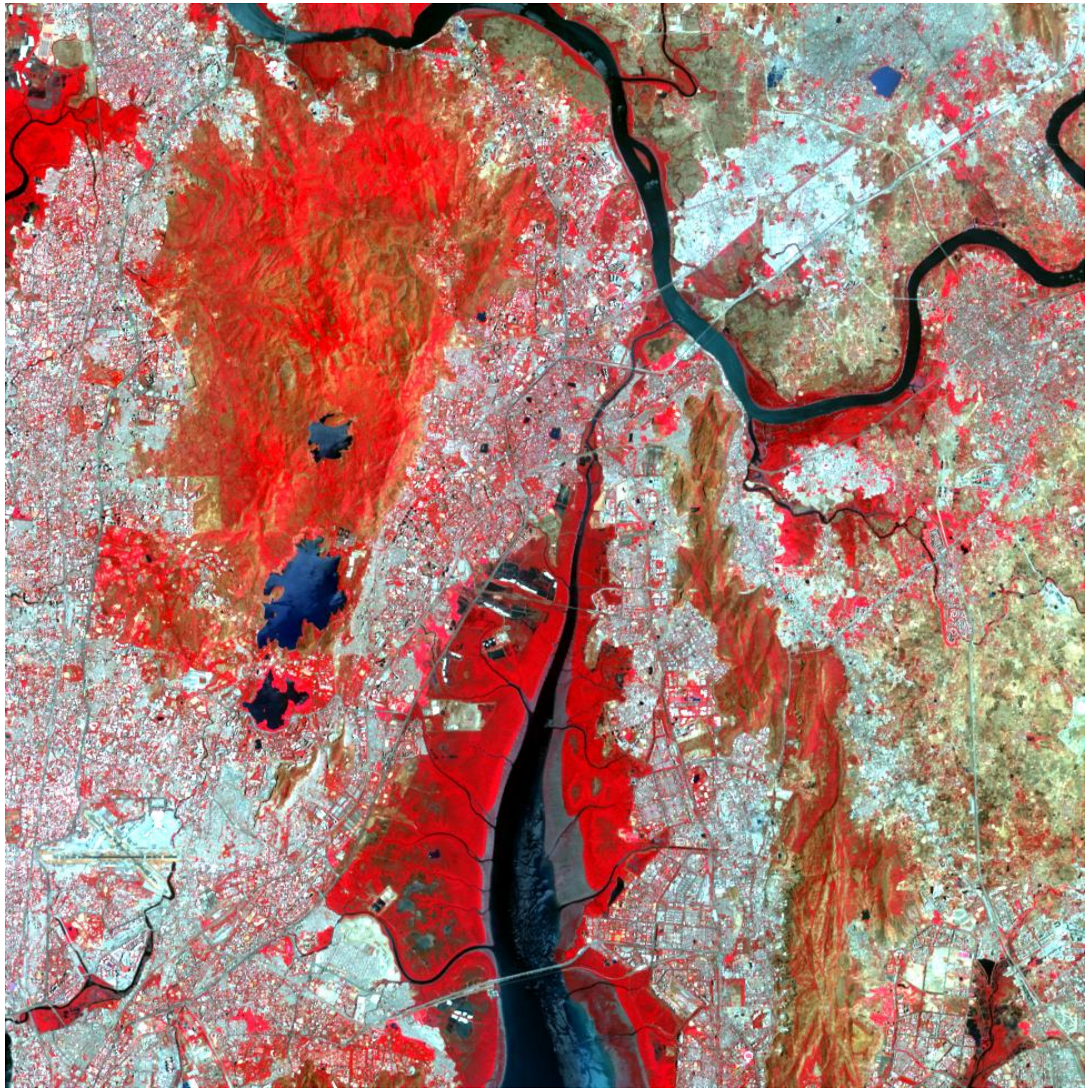
Procedure

Steps to Solve Problem

1. **Load the Multiband Image:**
   - Import the Landsat-8 image with 7 bands for the Mumbai scene.
2. **Compute Principal Components (PCA):**
   - Perform Principal Component Analysis (PCA) on the 7-band image to reduce dimensionality.
3. **Reconstruction Using 2, 3, ..., N-1 Principal Components:**
   - Reconstruct the image using inverse PCA with 2, 3, 4, 5, and 6 components to generate approximate images.
4. **Comparison:**
   - Compare the quality of reconstructed images using various quality metrics such as PSNR and SSIM to evaluate how much detail is retained with fewer components.
5. **Visualization:**
   - Display the original image alongside the reconstructed versions with different numbers of components for a visual comparison.

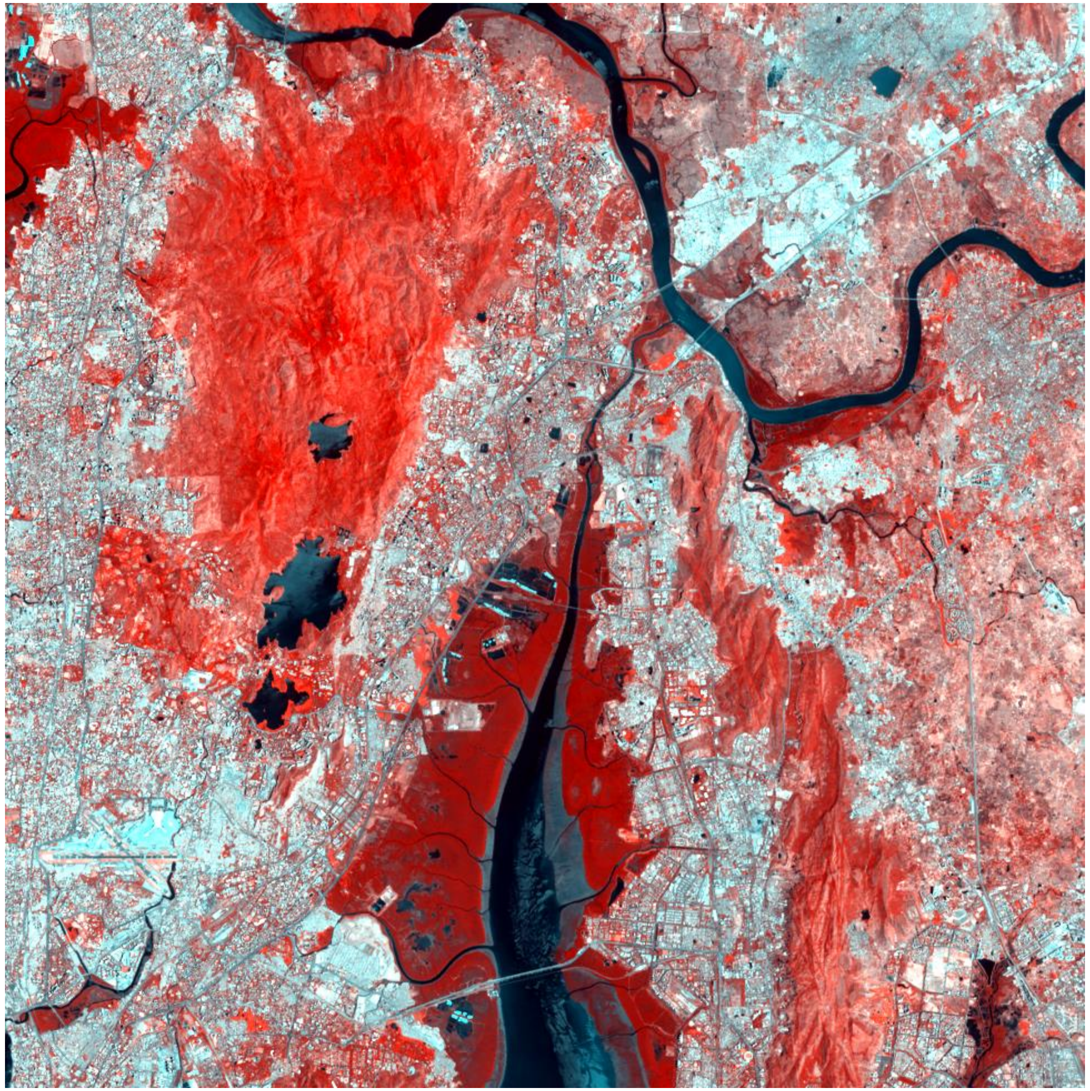
Input Images

Original RGB image

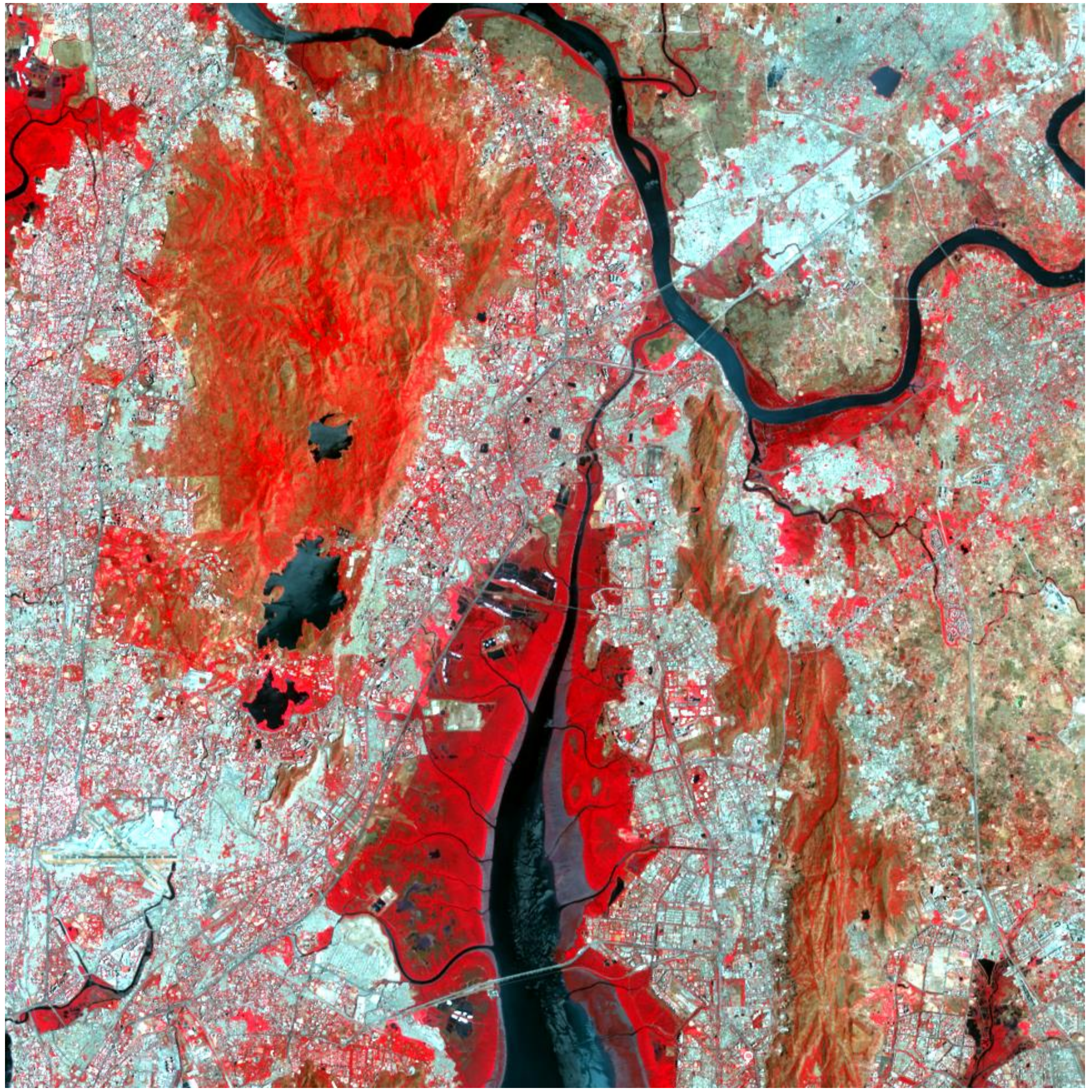


Original image in TCC

Reconstruction Results

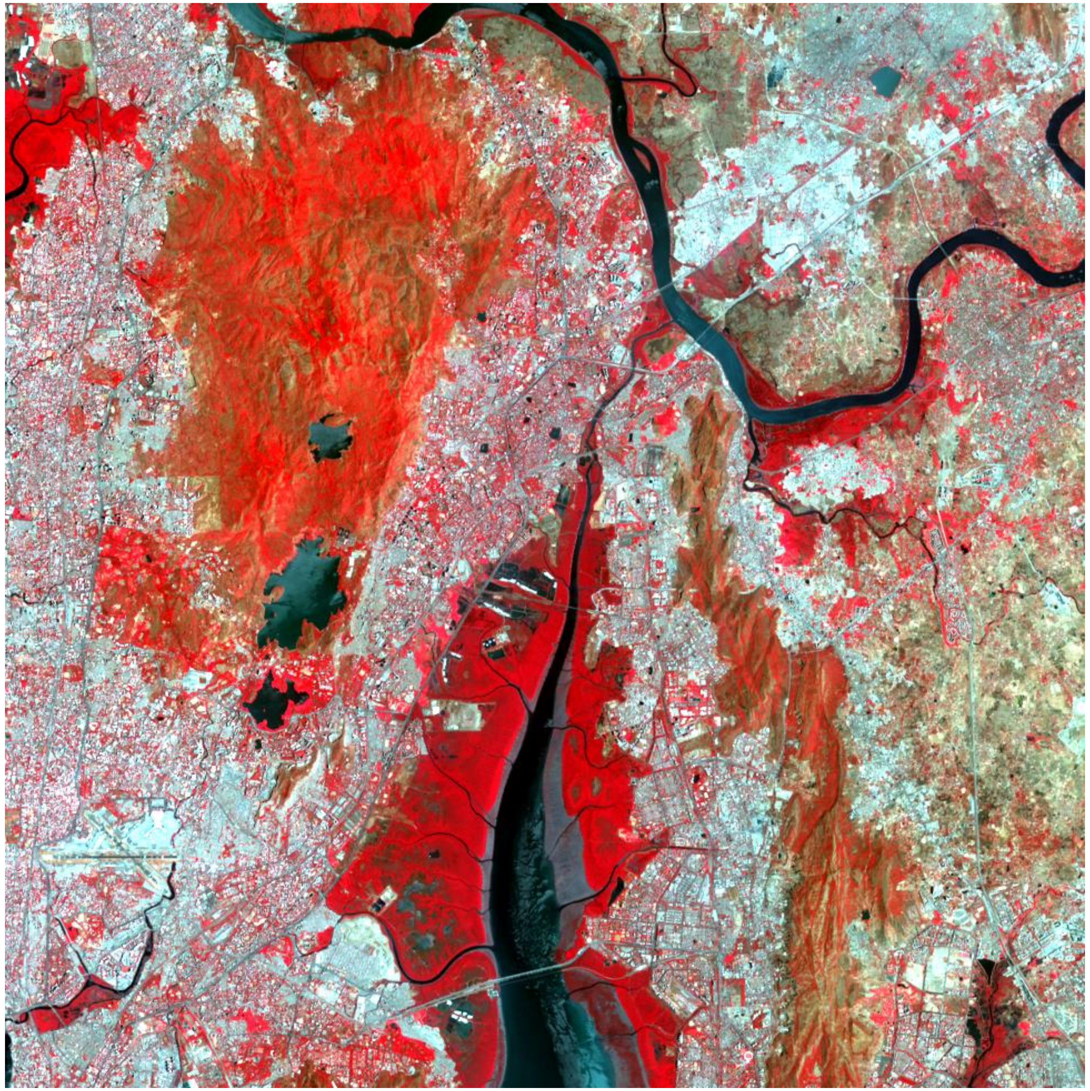


Reconstructed image with 1st 2 PCs; **PSNR – 42.37**

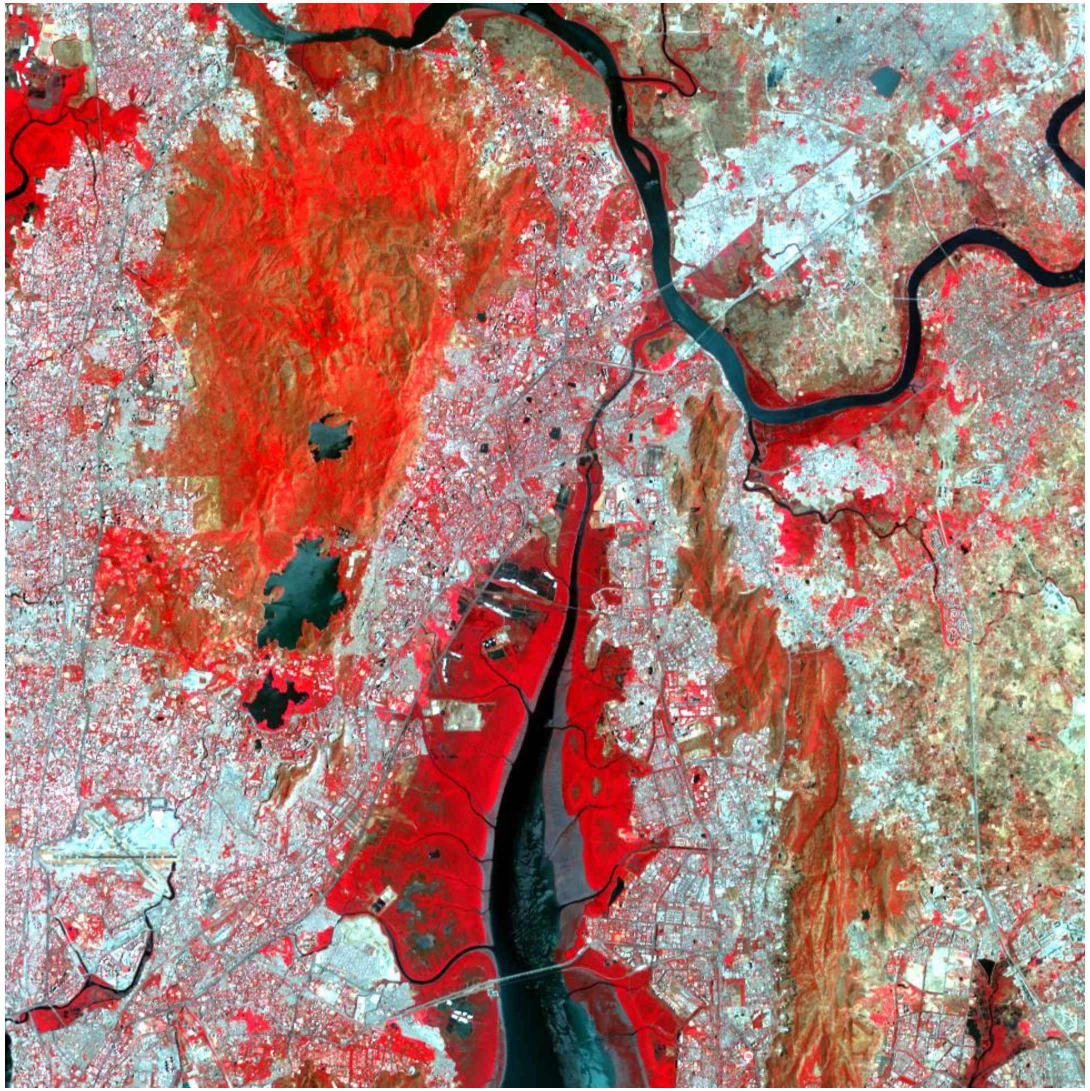


Reconstructed image with 1st 3 PCs; **PSNR – 53.18**

Reconstruction Results

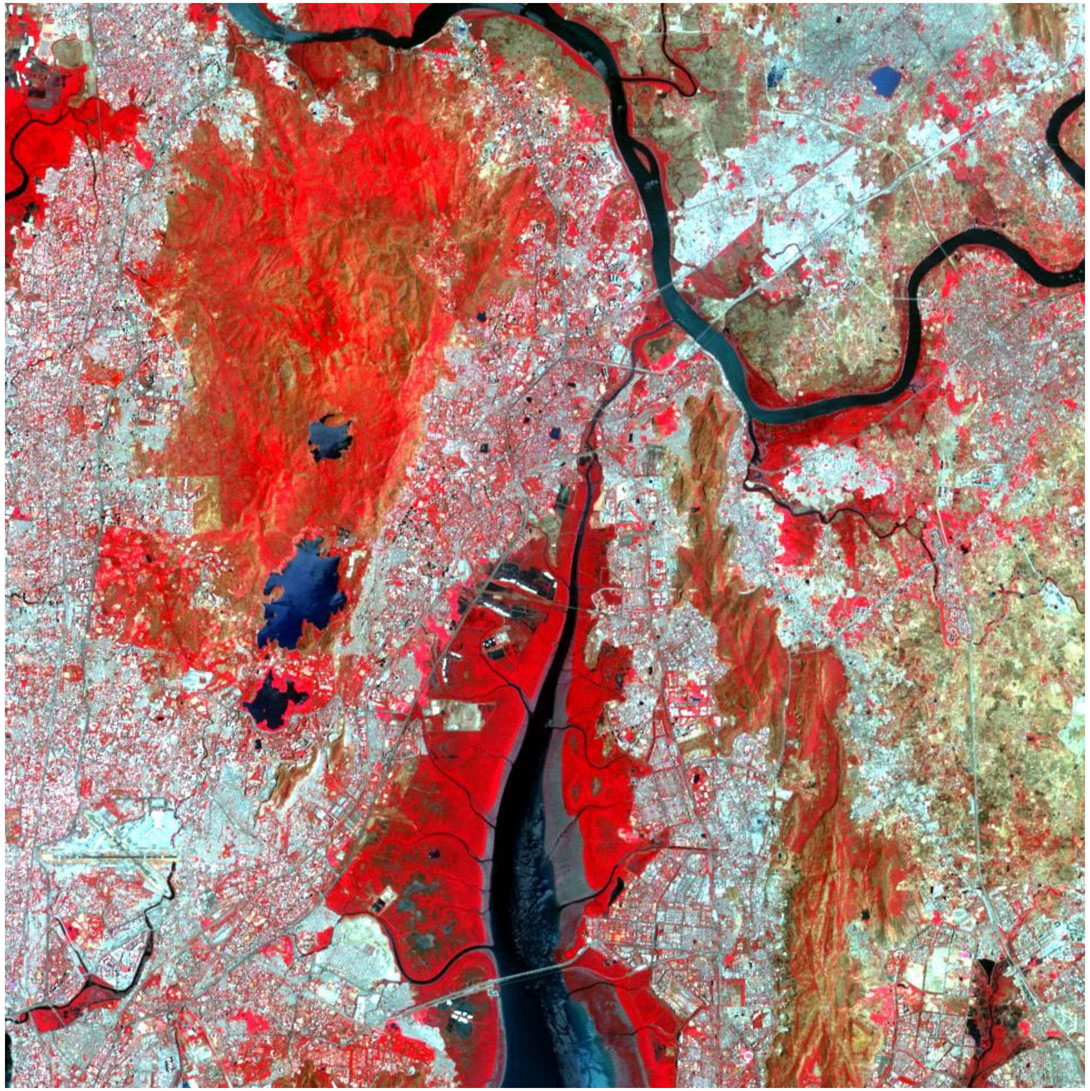


Reconstructed image with 1^{st} 4 PCs; **PSNR – 56.42**

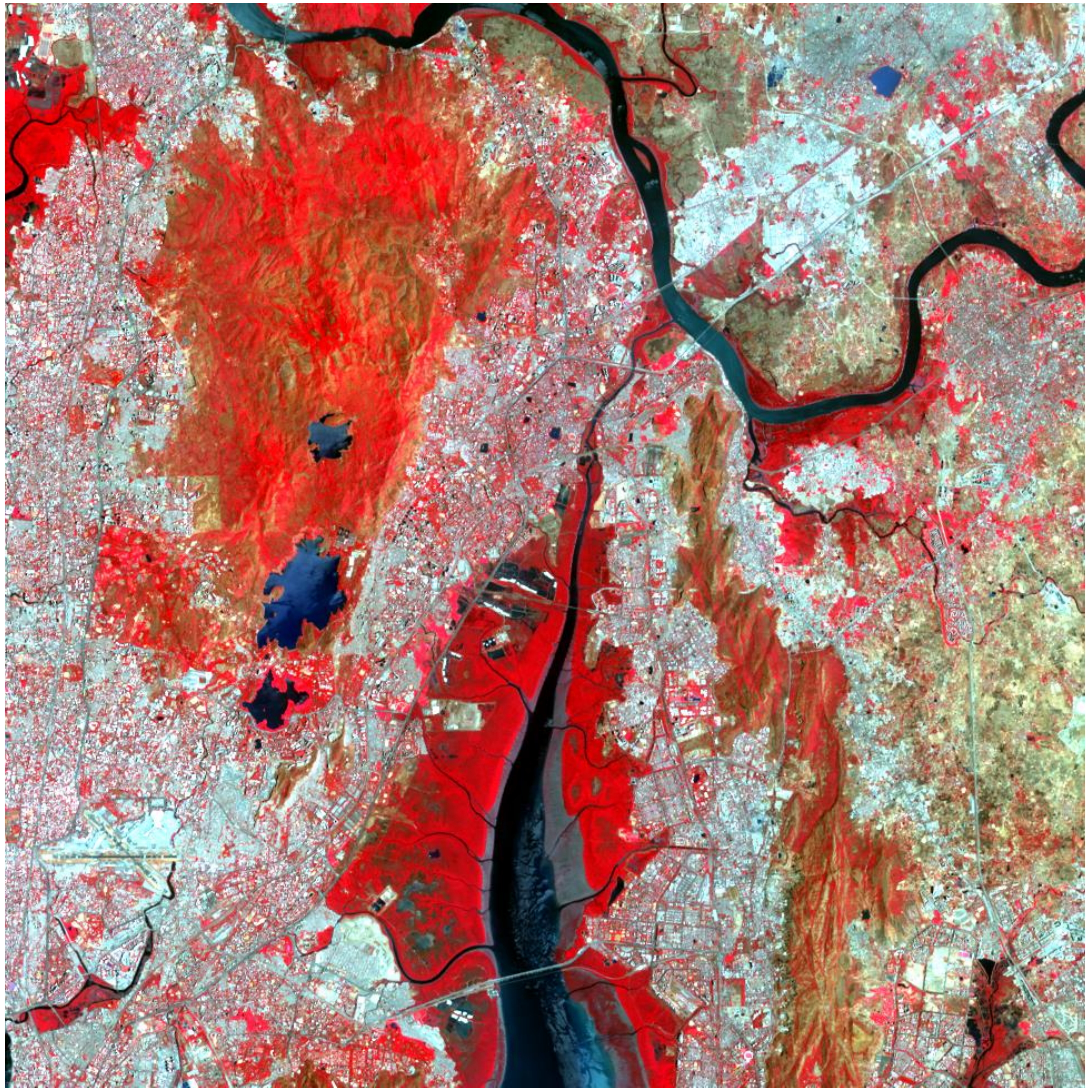


Reconstructed image with 1^{st} 5 PCs; **PSNR – 64.12**

Reconstruction Results



Reconstructed image with 1st 6 PCs; **PSNR – 73.59**



Reconstructed image with all 7 PCs; original image

Statistical Results

Number of Components	Explained Variance (%)	PSNR Value	SSIM Value
2	92.65	42.27	0.97481
3	98.75	53.18	0.997816
4	99.65	56.42	0.999007
5	99.82	64.12	0.999828
6	99.95	73.59	0.999979

PSNR – Peak Signal to Noise Ratio
SSIM – Structural Similarity Index Measure

Code Snippet

```
#importing packages
!pip install rasterio
import numpy as np
import rasterio
from rasterio.plot import show
import matplotlib.pyplot as plt
import cv2
import numpy as np
from skimage.metrics import structural_similarity as ssim

# Data preprocessing
def read_multiband_image(file_path):
    with rasterio.open(file_path) as src:
        bands = src.read()
        meta = src.meta
    return bands, meta

def write_image(file_path, image, meta):
    with rasterio.open(file_path, 'w', **meta) as dest:
        dest.write(image)
```

Code Snippet

```python
def normalize_data(bands):
    # Reshape bands to (num_bands, height * width)
    num_bands, height, width = bands.shape
    reshaped_bands = bands.reshape(num_bands, height * width)

    # Standardize each band
    mean = reshaped_bands.mean(axis=1, keepdims=True)
    std = reshaped_bands.std(axis=1, keepdims=True)
    normalized_bands = (reshaped_bands - mean) / std

    return normalized_bands.reshape(num_bands, height, width), mean, std
```

Code Snippet

```
def perform_pca(bands):
    num_bands, height, width = bands.shape
    reshaped_bands = bands.reshape(num_bands, height * width).T

    # Compute covariance matrix
    covariance_matrix = np.cov(reshaped_bands, rowvar=False)

    # Compute eigenvalues and eigenvectors
    eigenvalues, eigenvectors = np.linalg.eigh(covariance_matrix)

    # Sort eigenvalues and eigenvectors in descending order
    idx = np.argsort(eigenvalues)[::-1]
    eigenvalues = eigenvalues[idx]
    eigenvectors = eigenvectors[:, idx]

    # Project the data onto the eigenvectors
    principal_components = np.dot(reshaped_bands, eigenvectors)

    return eigenvectors, principal_components, (num_bands, height, width), eigenvalues
```

Code Snippet

```
def inverse_pca_and_reconstruct(eigenvectors, principal_components, shape, num_components,
mean, std):
    # Select the top num_components eigenvectors
    selected_eigenvectors = eigenvectors[:, :num_components]

    # Select the corresponding principal components
    selected_components = principal_components[:, :num_components]

    # Reconstruct the image
    reconstructed = np.dot(selected_components, selected_eigenvectors.T)
    reconstructed = reconstructed.T

    # De-normalize the reconstructed data
    num_bands, height, width = shape
    reconstructed = reconstructed.reshape(num_bands, height * width)
    reconstructed = (reconstructed * std) + mean
    reconstructed = reconstructed.reshape(num_bands, height, width)

    return reconstructed
```

Code Snippet

```
# Function to calculate PSNR
def calculate_psnr(original, compressed):
    mse = np.mean((original - compressed) ** 2)
    if mse == 0:  # MSE is zero means no noise is present in the signal, PSNR
is infinite
        return float('inf')
    max_pixel = 65536.0
    psnr_value = 20 * np.log10(max_pixel / np.sqrt(mse))
    return psnr_value

# Function to calculate SSIM
def calculate_ssim(original, compressed):
    ssim_value = ssim(original, compressed, multichannel=True,
data_range=65536)
    return ssim_value
```