```
!pip install rasterio numpy opencv-python-headless
```

```
Collecting rasterio
  Downloading rasterio-1.4.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.1 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.11/dist-packages (4.11.0.86)
Collecting affine (from rasterio)
  Downloading affine-2.4.0-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: attrs in /usr/local/lib/python3.11/dist-packages (from rasterio) (25.2.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from rasterio) (2025.1.31)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.11/dist-packages (from rasterio) (8.1.8)
Collecting cligj>=0.5 (from rasterio)
  Downloading cligj-0.7.2-py3-none-any.whl.metadata (5.0 kB)
Collecting click-plugins (from rasterio)
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl.metadata (6.4 kB)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.11/dist-packages (from rasterio) (3.2.1)
Downloading rasterio-1.4.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (22.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22.2/22.2 MB 23.7 MB/s eta 0:00:00
Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Downloading affine-2.4.0-py3-none-any.whl (15 kB)
Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Installing collected packages: cligj, click-plugins, affine, rasterio
Successfully installed affine-2.4.0 click-plugins-1.1.1 cligj-0.7.2 rasterio-1.4.3
```

```
import os
import rasterio
import numpy as np
import cv2

# Path to folder containing GeoTIFF images
input_folder = "/content/drive/MyDrive/GEE_AQI_Images"
output_folder = "path_to_output_numpy"

# Ensure output folder exists
os.makedirs(output_folder, exist_ok=True)

# Resize parameters
TARGET_SIZE = (128, 128)  # Can be (256, 256) if needed

# Function to normalize an image (0-1 range)
def normalize_image(image):
    min_val = np.min(image)
    max_val = np.max(image)
    return (image - min_val) / (max_val - min_val + 1e-8)  # Avoid division by zero

# Process all GeoTIFF images
for filename in os.listdir(input_folder):
    if filename.endswith(".tif") or filename.endswith(".tiff"):
        file_path = os.path.join(input_folder, filename)

        # Open GeoTIFF
        with rasterio.open(file_path) as dataset:
            # Read all bands into a NumPy array
            image = dataset.read()  # Shape: (bands, height, width)

            # Ensure the image has 4 bands (NO2, CO, O3, PM2.5)
            if image.shape[0] < 4:
                print(f"Skipping {filename} (not enough bands)")
                continue

            # Select the first 4 bands (assuming order: NO2, CO, O3, PM2.5)
            image = image[:4, :, :]

            # Normalize each band
            image = np.array([normalize_image(image[i]) for i in range(4)])

            # Resize each band to TARGET_SIZE
            resized_image = np.array([cv2.resize(image[i], TARGET_SIZE, interpolation=cv2.INTER_CUBIC) for i in range(4)])

            # Save as NumPy array
            output_file = os.path.join(output_folder, filename.replace(".tif", ".npy"))
            np.save(output_file, resized_image)
            print(f"Processed: {filename} → {output_file}")

print("✅ All images processed and saved as NumPy arrays!")
```

```
Processed: Delhi_AQI.tif → path_to_output_numpy/Delhi_AQI.npy
Processed: Mumbai_AQI.tif → path_to_output_numpy/Mumbai_AQI.npy
Processed: Bengaluru_AQI.tif → path_to_output_numpy/Bengaluru_AQI.npy
Processed: Kolkata_AQI.tif → path_to_output_numpy/Kolkata_AQI.npy
Processed: Hyderabad_AQI.tif → path_to_output_numpy/Hyderabad_AQI.npy
Processed: Chennai_AQI.tif → path_to_output_numpy/Chennai_AQI.npy
```

```
Processed: Ahmedabad_AQI.tif → path_to_output_numpy/Ahmedabad_AQI.npy
Processed: Indore_AQI.tif → path_to_output_numpy/Indore_AQI.npy
Processed: Pune_AQI.tif → path_to_output_numpy/Pune_AQI.npy
Processed: Jaipur_AQI.tif → path_to_output_numpy/Jaipur_AQI.npy
✅ All images processed and saved as NumPy arrays!
```

To remove missing values(NaN)

```
import os
import numpy as np

# Define input and output folder
input_folder = "/content/path_to_output_numpy"
output_folder = "path_to_clean_folder"

# Ensure output folder exists
os.makedirs(output_folder, exist_ok=True)

# Process all `.npy` files in the folder
for filename in os.listdir(input_folder):
    if filename.endswith(".npy"):
        file_path = os.path.join(input_folder, filename)

        # Load the file
        data = np.load(file_path)

        # Replace NaNs with 0 (or use np.nanmean(data) for mean imputation)
        cleaned_data = np.nan_to_num(data, nan=0.0)

        # Save the cleaned file
        output_file = os.path.join(output_folder, filename.replace(".npy", "_clean.npy"))
        np.save(output_file, cleaned_data)
        print(f"✅ Cleaned and saved: {output_file}")

print("🚀 All files processed successfully!")
```

```
✅ Cleaned and saved: path_to_clean_folder/Delhi_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Chennai_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Kolkata_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Mumbai_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Jaipur_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Indore_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Pune_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Bengaluru_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Hyderabad_AQI_clean.npy
✅ Cleaned and saved: path_to_clean_folder/Ahmedabad_AQI_clean.npy
🚀 All files processed successfully!
```