**Steps:**

Read the images using rasterio as well as cv2

-------------------------------------------------------------------------------------------

Check the shape #Output: height width channel

--------------------------------------------------------------------------------------------

**Extra for Reading:**

**Patching:**

size\_x = (image.shape[1]//image\_patch\_size)\*image\_patch\_size

size\_y = (image.shape[0]//image\_patch\_size)\*image\_patch\_size

print("{}----{} - {}".format(image.shape , size\_x , size\_y))

#print(image.shape)

#These lines calculate the adjusted width (size\_x) and height (size\_y) by finding the closest multiple of image\_patch\_size that is less than or equal to the original width and height.

**#OutPut**

#Indicates that the original image had a shape of (644, 797, 3), and the adjusted sizes are 768 (width) and 512 (height), making them multiples of the specified image\_patch\_size.

---------------------------------------------------------------------------------------------

Check type of Image:

1

print(type(image))

<class 'numpy.ndarray'>

-----------------------------------------------------------------------------

# Convert image into image type

1

from PIL import Image #Using Pillow Library

2

import numpy as np

1

type(Image.fromarray(image))

-----PIL.Image.Image

------------------------------------------------------------------------------

**Crop According to closest multiple of image\_patch\_size**

------------------------------------------------------------------------------

**Now Overall code of patching is:**

**Also Check: https://www.youtube.com/watch?v=7IL7LKSLb9I**

image\_dataset = []

size\_x = (image.shape[1]//image\_patch\_size)\*image\_patch\_size

size\_y = (image.shape[0]//image\_patch\_size)\*image\_patch\_size

image = Image.fromarray(image)

image = image.crop((0,0,size\_x,size\_y))

#print("({} , {})".format(image.size[0] , image.size[1]))

#print(image.shape)

image=np.array(image)

patched\_images = patchify(image,(image\_patch\_size , image\_patch\_size , 3),step = image\_patch\_size)

#print(len(patched\_images))

#patched\_images.shape[0] corresponds to the height of the array (number of patches in the vertical direction).

#patched\_images.shape[1] corresponds to the width of the array (number of patches in the horizontal direction).

for i in range(patched\_images.shape[0]):

for j in range(patched\_images.shape[1]):

individual\_patched\_image = patched\_images[i,j,:,:] #The : in the indexing means "all elements along this axis."

print(individual\_patched\_image.shape)

# Normalizing Images

# using MinMax scaler convert each image pixel into 0 and 1

from sklearn.preprocessing import MinMaxScaler,StandardScaler

minmaxscaler = MinMaxScaler()

Add code in end of the above provided code:

individual\_patched\_image= minmaxscaler.fit\_transform(individual\_patched\_image.reshape(-1 , individual\_patched\_image.shape[-1])).reshape(individual\_patched\_image.shape)

individual\_patched\_image = individual\_patched\_image[0]

image\_dataset.append(individual\_patched\_image)

**Final:**

image\_dataset = np.array(image\_dataset)

**Input ready Ready**

**--------------------------------------------------------------------**

**Mask Images:  
https://www.tutorialspoint.com/how-to-mask-an-image-in-opencv-python**

[**https://youtu.be/YHA\_-QMB8\_U?si=YSH17AQZoE769KIi**](https://youtu.be/YHA_-QMB8_U?si=YSH17AQZoE769KIi)**Same steps Preprocessing as image\_dataset**

**Follow Pre-Processing notebook**

**---------------------------------------------------------------------------------------------**