```
!pip install patchify
```

```
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Collecting patchify
      Downloading patchify-0.2.3-py3-none-any.whl (6.6 kB)
    Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.7/dist-packages (from patchify) (1.21.6)
    Installing collected packages: patchify
    Successfully installed patchify-0.2.3
```

```
import os
import cv2
from PIL import Image
import numpy as np
from patchify import patchify
from sklearn.preprocessing import MinMaxScaler, StandardScaler

from matplotlib import pyplot as plt
import random
```

```
minmaxscaler = MinMaxScaler()
```

```
!ls -lah '/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset'
```

```
    total 33K
    -rw------- 1 root root  548 Feb 13  2020  classes.json
    drwx------ 2 root root 4.0K Oct 22 00:47 'Tile 1'
    drwx------ 2 root root 4.0K Oct 22 00:47 'Tile 2'
    drwx------ 2 root root 4.0K Oct 22 00:47 'Tile 3'
    drwx------ 2 root root 4.0K Oct 22 00:47 'Tile 4'
    drwx------ 2 root root 4.0K Oct 22 00:47 'Tile 5'
    drwx------ 2 root root 4.0K Oct 22 00:47 'Tile 6'
    drwx------ 2 root root 4.0K Oct 22 00:47 'Tile 7'
    drwx------ 2 root root 4.0K Oct 22 00:47 'Tile 8'
```

```
dataset_root_folder = '/content/drive/MyDrive/Colab Notebooks/datasets/satellite/'
```

```
dataset_name = "DubaiDataset"
```

```
for path, subdirs, files in os.walk(os.path.join(dataset_root_folder, dataset_name)):
  dir_name = path.split(os.path.sep)[-1]
  #print(dir_name)
  if dir_name == 'masks': # 'images
    images = os.listdir(path)
    print(path)
    #print(images)
    for i, image_name in enumerate(images):
      if (image_name.endswith('.png')): # '.jpg
```

```
        #print(image_name)
        a = True
```

```
/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset/Tile 4/masks
/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset/Tile 7/masks
/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset/Tile 1/masks
/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset/Tile 2/masks
/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset/Tile 8/masks
/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset/Tile 3/masks
/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset/Tile 6/masks
/content/drive/MyDrive/Colab Notebooks/datasets/satellite/DubaiDataset/Tile 5/masks
```

```
image_patch_size = 256
```

```
image = cv2.imread(f'{dataset_root_folder}/{dataset_name}/Tile 2/images/image_part_001.jpg',1)
```

```
image.shape
```

```
(544, 509, 3)
```

```
image_patches = patchify(image, (image_patch_size, image_patch_size, 3), step=image_patch_size)
```

```
len(image_patches)
```

```
2
```

```
print(image_patches.shape)
```

```
(2, 1, 1, 256, 256, 3)
```

```
minmaxscaler = MinMaxScaler()
```

```
image_x = image_patches[0,0,:,:]
#MinMaxScaler
image_y = minmaxscaler.fit_transform(image_x.reshape(-1, image_x.shape[-1])).reshape(image_x.shape)
```

```
image_y[0].shape
```

```
(256, 256, 3)
```

```
print(type(image))
```

```
<class 'numpy.ndarray'>
```

```
type(Image.fromarray(image))
```

```
PIL.Image.Image
```

```
image.shape
```

```
(544, 509, 3)
```

```
(image.shape[0]//image_patch_size)*image_patch_size
```

```
512
```

```
image_dataset = []
mask_dataset = []

for image_type in ['images' , 'masks']:
  if image_type == 'images':
    image_extension = 'jpg'
  elif image_type == 'masks':
     image_extension = 'png'
  for tile_id in range(1,8):
    for image_id in range(1,20):
      image = cv2.imread(f'{dataset_root_folder}/{dataset_name}/Tile {tile_id}/{image_type}/image_part_00{image_id}.{image_extension}',1)
      if image is not None:
        if image_type == 'masks':
          image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        #print(image.shape)
        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))
        image = Image.fromarray(image)
        image = image.crop((0,0, size_x, size_y))
        #print("({},  {})".format(image.size[0],image.size[1]))
        image = np.array(image)
        patched_images = patchify(image, (image_patch_size, image_patch_size, 3), step=image_patch_size)
        #print(len(patched_images))
        for i in range(patched_images.shape[0]):
          for j in range(patched_images.shape[1]):
            if image_type == 'images':
              individual_patched_image = patched_images[i,j,:,:]
              #print(individual_patched_image.shape)
              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]
              #print(individual_patched_image.shape)
              image_dataset.append(individual_patched_image)
            elif image_type == 'masks':
              individual_patched_mask = patched_images[i,j,:,:]
              individual_patched_mask = individual_patched_mask[0]
              mask_dataset.append(individual_patched_mask)
```

```
print(len(image_dataset))
print(len(mask_dataset))
```

```
945
945
```

```
image_dataset = np.array(image_dataset)
mask_dataset = np.array(mask_dataset)
```

```
print(len(image_dataset))
print(len(mask_dataset))
```

```
945
945
```

```
type(image_dataset[0])
```

```
numpy.ndarray
```

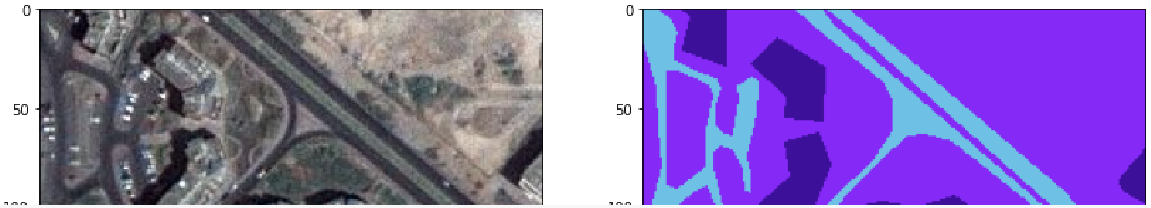
```
type(np.reshape(image_dataset[0], (image_patch_size, image_patch_size, 3)))
```

```
numpy.ndarray
```

```
random_image_id = random.randint(0, len(image_dataset))

plt.figure(figsize=(14,8))
plt.subplot(121)
plt.imshow(image_dataset[random_image_id])
plt.subplot(122)
plt.imshow(mask_dataset[random_image_id])
```

```
<matplotlib.image.AxesImage at 0x7feadb413bd0>
```



```
class_building = '#3C1098'
class_building = class_building.lstrip('#')
class_building = np.array(tuple(int(class_building[i:i+2], 16) for i in (0,2,4)))
print(class_building)

class_land = '#8429F6'
class_land = class_land.lstrip('#')
class_land = np.array(tuple(int(class_land[i:i+2], 16) for i in (0,2,4)))
print(class_land)

class_road = '#6EC1E4'
class_road = class_road.lstrip('#')
class_road = np.array(tuple(int(class_road[i:i+2], 16) for i in (0,2,4)))
print(class_road)

class_vegetation = '#FEDD3A'
class_vegetation = class_vegetation.lstrip('#')
class_vegetation = np.array(tuple(int(class_vegetation[i:i+2], 16) for i in (0,2,4)))
print(class_vegetation)

class_water = '#E2A929'
class_water = class_water.lstrip('#')
class_water = np.array(tuple(int(class_water[i:i+2], 16) for i in (0,2,4)))
print(class_water)

class_unlabeled = '#9B9B9B'
class_unlabeled = class_unlabeled.lstrip('#')
class_unlabeled = np.array(tuple(int(class_unlabeled[i:i+2], 16) for i in (0,2,4)))
print(class_unlabeled)
```

```
[ 60  16 152]
[132  41 246]
[110 193 228]
[254 221  58]
[226 169  41]
[155 155 155]
```

```
mask_dataset.shape[0]
```

```
945
```

```
label = individual_patched_mask
```

```
def rgb_to_label(label):
  label_segment = np.zeros(label.shape, dtype=np.uint8)
  label_segment[np.all(label == class_water, axis=-1)] = 0
  label_segment[np.all(label == class_land, axis=-1)] = 1
  label_segment[np.all(label == class_road, axis=-1)] = 2
  label_segment[np.all(label == class_building, axis=-1)] = 3
  label_segment[np.all(label == class_vegetation, axis=-1)] = 4
  label_segment[np.all(label == class_unlabeled, axis=-1)] = 5
  #print(label_segment)
  label_segment = label_segment[:,:,0]
  #print(label_segment)
  return label_segment
```

```
labels = []
for i in range(mask_dataset.shape[0]):
  label = rgb_to_label(mask_dataset[i])
  labels.append(label)
```

```
print(len(labels))
```

```
     945
```

```
labels = np.array(labels)
```

```
labels[3]
```

```
     array([[1, 1, 1, ..., 1, 1, 1],
            [1, 1, 1, ..., 1, 1, 1],
            [1, 1, 1, ..., 1, 1, 1],
            ...,
            [1, 1, 1, ..., 1, 1, 1],
            [1, 1, 1, ..., 1, 1, 1],
            [1, 1, 1, ..., 1, 1, 1]], dtype=uint8)
```

```
labels = np.expand_dims(labels, axis=3)
```

```
labels[0]
```

```
     array([[[1],
             [1],
             [1],
             ...,
             [1],
             [1],
             [1]],
       [[1],
        [1],
        [1],
        ...,
        [1],
        [1],
        [1]],

       [[1],
        [1],
        [1],
        ...,
        [1],
        [1],
        [1]],

       ...,

       [[1],
        [1],
        [1],
        ...,
        [1],
        [1],
        [1]],

       [[1],
        [1],
        [1],
        ...,
        [1],
        [1],
        [1]],

       [[1],
        [1],
        [1],
        ...,
        [1],
        [1],
        [1]]], dtype=uint8)
```

```
np.unique(labels)
```

```
array([0, 1, 2, 3, 4, 5], dtype=uint8)
```

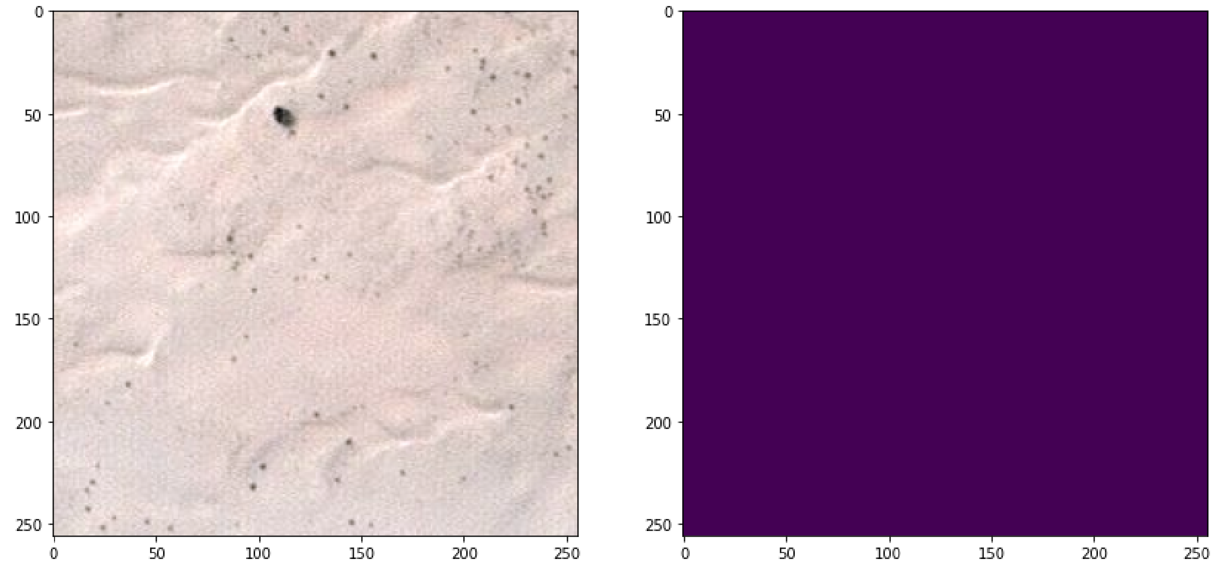
```
print("Total unique labels based on masks: ",format(np.unique(labels)))
```

```
Total unique labels based on masks:  [0 1 2 3 4 5]
```

```
random_image_id = random.randint(0, len(image_dataset))

plt.figure(figsize=(14,8))
plt.subplot(121)
plt.imshow(image_dataset[random_image_id])
plt.subplot(122)
#plt.imshow(mask_dataset[random_image_id])
plt.imshow(labels[random_image_id][:,:,0])
```

```
<matplotlib.image.AxesImage at 0x7feadb106410>
```



```
labels[0][:,:,0]
```

```
array([[1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       ...,
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 1]], dtype=uint8)
```

```
total_classes = len(np.unique(labels))
```

```
total_classes
```

```
6
```

```
from tensorflow.keras.utils import to_categorical
```

```
labels_categorical_dataset = to_categorical(labels, num_classes=total_classes)
```

```
labels_categorical_dataset.shape
```

```
    (945, 256, 256, 6)
```

```
master_trianing_dataset.shape
```

```
    (945, 256, 256, 3)
```

```
master_trianing_dataset = image_dataset
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(master_trianing_dataset, labels_categorical_dataset, test_size=0.15, random_state=100)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
    (803, 256, 256, 3)
    (142, 256, 256, 3)
    (803, 256, 256, 6)
    (142, 256, 256, 6)
```

```
image_height = X_train.shape[1]
image_width = X_train.shape[2]
image_channels = X_train.shape[3]
total_classes = y_train.shape[3]
```

```
print(image_height)
print(image_width)
print(image_channels)
print(total_classes)
```

```
    256
    256
    3
    6
```