Image To Image Translation

In [109]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from keras.preprocessing.image import img_to_array, load_img
from tqdm import tqdm
import cv2
from sklearn.model_selection import train_test_split
```

Loading Data

In [1]:

```
!wget --header="Host: efrosgans.eecs.berkeley.edu" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.61 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/" "http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/maps.tar.gz" -c -O 'maps.tar.gz'
```

```
--2021-10-05 10:15:08--  http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/maps.tar.gz
Resolving efrosgans.eecs.berkeley.edu (efrosgans.eecs.berkeley.edu)... 128.32.244.190
Connecting to efrosgans.eecs.berkeley.edu (efrosgans.eecs.berkeley.edu)|128.32.244.190|:80... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 250242400 (239M) [application/x-gzip]
Saving to: ‘maps.tar.gz’

maps.tar.gz         100%[===================>] 238.65M  1.55MB/s    in 86s

2021-10-05 10:16:35 (2.77 MB/s) - ‘maps.tar.gz’ saved [250242400/250242400]
```

In [ ]:

```
# unzipping dataset
!tar -xzvf "/content/maps.tar.gz" -C "/content"
```

EDA

In [4]:

```
train_path = '/content/maps/train'
val_path = '/content/maps/val'
```

In [141]:

```
train_images_path = []
val_images_path = []

for img_path in os.listdir(train_path):
    train_images_path.append(os.path.join(train_path, img_path))

for img_path in os.listdir(val_path):
    val_images_path.append(os.path.join(val_path, img_path))
```

Splitting dataset

In [142]:

```
val_images_path, test_images_path = train_test_split(val_images_path, test_size = 0.05, random_state =4
2, shuffle = True, )
```

In [143]:

```
print('Count of images in train data : ', len(train_images_path))
print('Count of images in val data : ', len(val_images_path))
print('Count of images in test data: ',len(test_images_path))
```

```
Count of images in train data :  1096
Count of images in val data :  1043
Count of images in test data:  55
```

> Train data contains 1096 images while val data contain 988 images. Let's look into train data.

Let's display few images to see how data looks like

In [36]:

```
#displaying sample images
plt.figure(figsize=(20,40))
for i, img_path in enumerate(train_images_path[25:30]):
    plt.subplot(5,1,i+1)
    img = plt.imread(img_path) # reading image path
    plt.title('Image {} from train dataset'.format(i+1))
    plt.imshow(img) #plotting image
```

Image 1 from train dataset

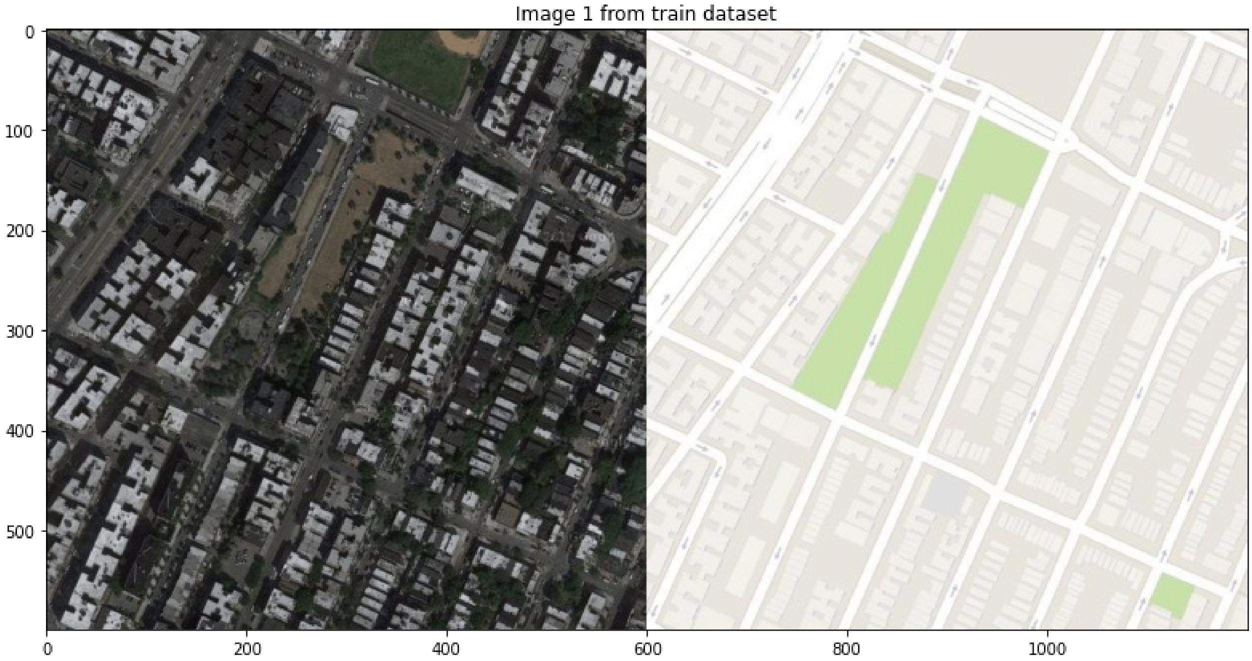


Image 2 from train dataset

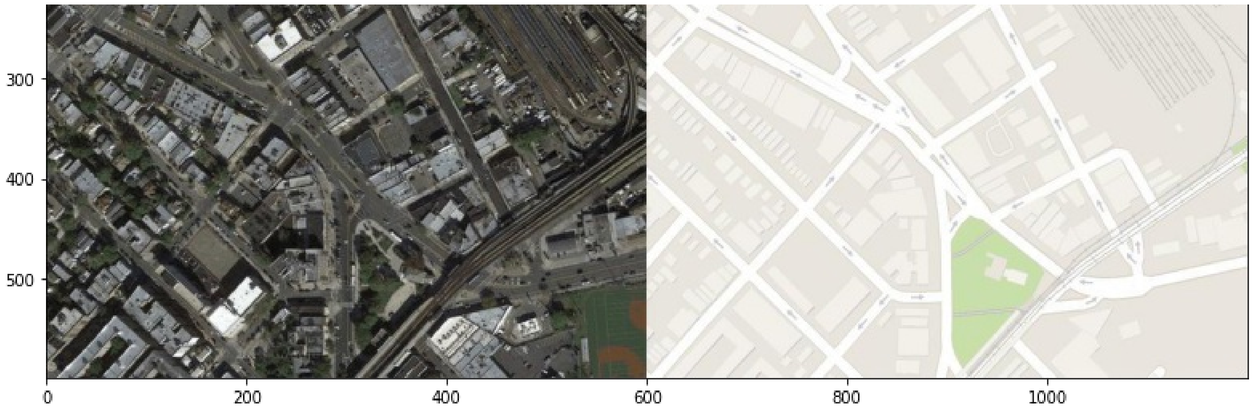


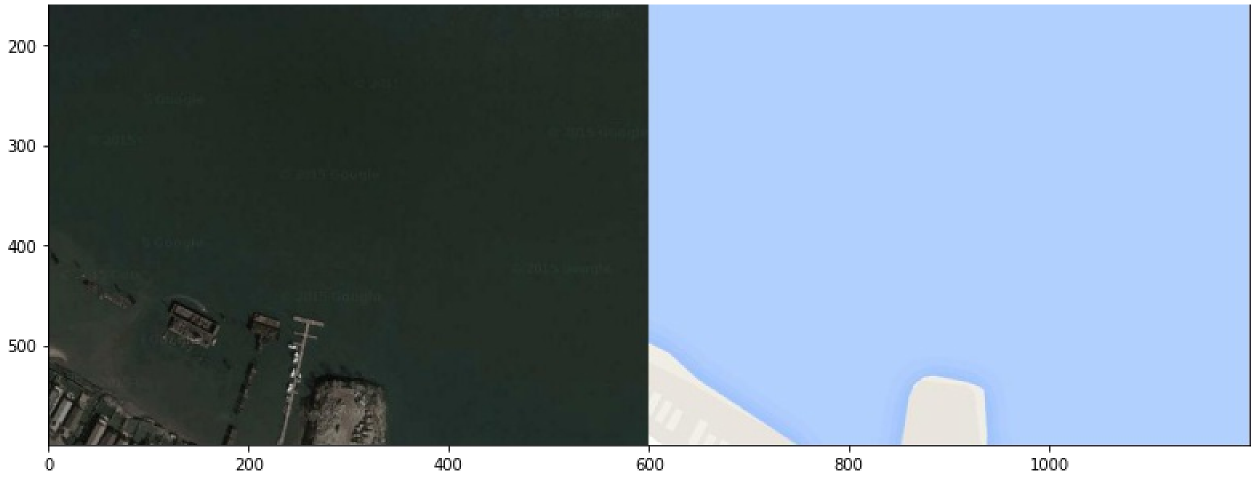
Image 3 from train dataset

0
100
200
300
400
500
0
200
400
600
800
1000

Image 4 from train dataset

0
100
200
300
400
500
0
200
400
600
800
1000

Image 5 from train dataset



> As you can see from displayed images above, an image contain 2 fragments: satellite and aerial map. For our purpose, satellite fragment is input and aerial map is output. So, we'll be splitting our given image into two.

In [41]:

```
img = Image.open(train_images_path[0])
```

In [43]:

```
print('Size of image: ',img.size)
```

```
Size of image:  (1200, 600)
```

> Each image is of size 1200x600 (width x height) which consits of both satellite and map image as well. So, after splitting image size will be 600x600

Let's split the image into sat. and map part.

Splitting images

In image splitting, we'll also be resizing images to 256x256 from the original size of 600x600.

In [69]:

```
#https://keras.io/api/preprocessing/image/#loadimg-function
#https://machinelearningmastery.com/how-to-develop-a-pix2pix-gan-for-image-to-image-translation/

def split_images(images):
    '''This function take list of image paths as input and return 2 arrays: source images and target im
ages'''
    source_images = []
    target_images = []
    for image_path in tqdm(images):
        img = load_img(path = image_path, target_size= (256,512))
        pixels = img_to_array(img)

        sat_img, map_img = pixels[:,:256], pixels[:,256:]

        source_images.append(sat_img)
        target_images.append(map_img)

    return np.asarray(source_images), np.asarray(target_images)
```

In [71]:

```
train_sat_images, train_map_images = split_images(train_images_path)
```

```
100%|██████████| 1096/1096 [00:18<00:00, 58.15it/s]
```

In [72]:

```
val_sat_images, val_map_images = split_images(val_images_path)
```

```
100%|██████████| 1098/1098 [00:18<00:00, 58.78it/s]
```

In [144]:

```
test_sat_images, test_map_images = split_images(test_images_path)
```

```
100%|██████████| 55/55 [00:01<00:00, 52.12it/s]
```

In [74]:

```
train_sat_images.shape
```

Out[74]:

```
(1096, 256, 256, 3)
```

In [75]:

```
train_map_images.shape
```

Out[75]:

```
(1096, 256, 256, 3)
```

In [79]:

```
val_sat_images.shape
```

Out[79]:

```
(1098, 256, 256, 3)
```

In [80]:

```
val_map_images.shape
```

Out[80]:

```
(1098, 256, 256, 3)
```

In [145]:

```
test_sat_images.shape
```

Out[145]:

```
(55, 256, 256, 3)
```

In [146]:

```
test_map_images.shape
```

Out[146]:

```
(55, 256, 256, 3)
```

Data distribution for Satellite images

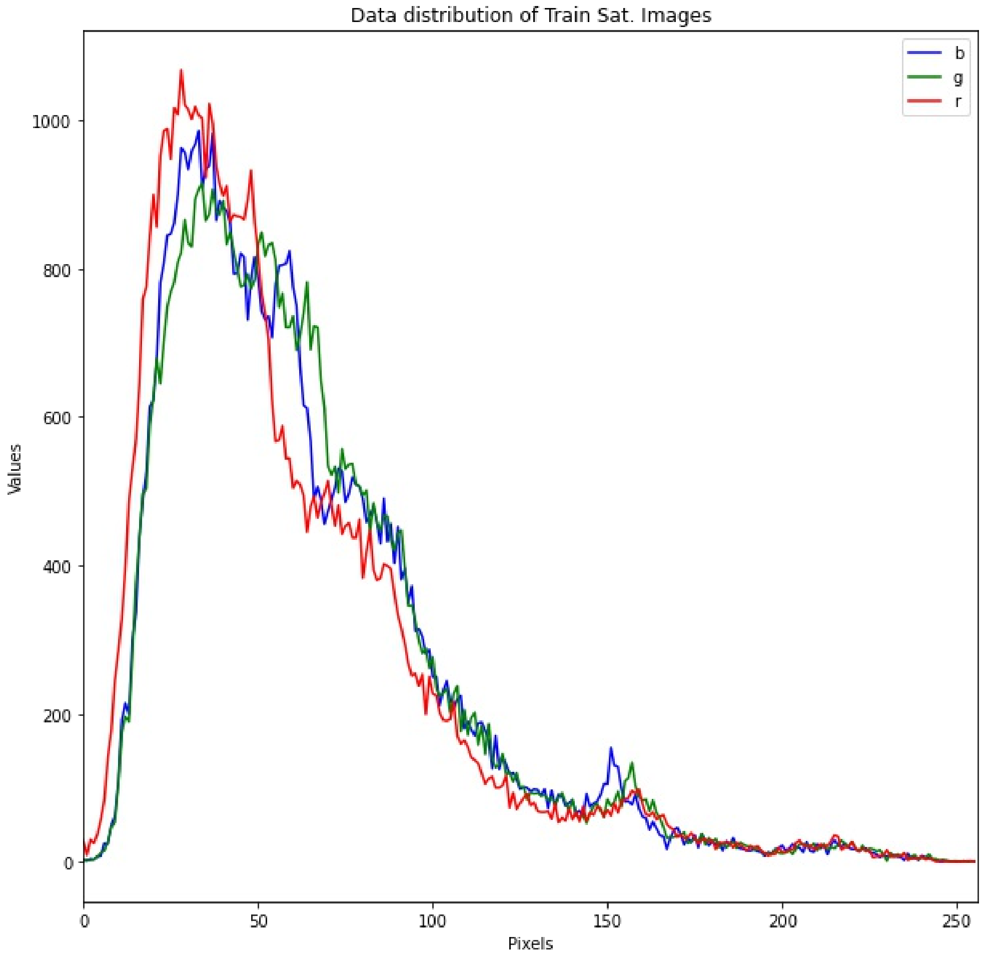
In [104]:

```
def plot_dist(data,label):
    color = ('b','g','r')
    plt.figure(figsize=(10,10))
    for i,col in enumerate(color):
        histr = cv2.calcHist(data,[i],None,[256],[0,256])
        plt.plot(histr,color = col, label = col)
        plt.xlim([0,256])
    plt.title('Data distribution of {} Images'.format(label))
    plt.xlabel('Pixels')
    plt.ylabel('Values')
    plt.legend()
    plt.show()
```

In [107]:

```
#https://docs.opencv.org/3.1.0/d1/db7/tutorial_py_histogram_begins.html
#https://datascience.stackexchange.com/questions/45711/how-can-i-plot-display-a-dataset-or-an-image-dis
tribution

plot_dist(train_sat_images,'Train Sat.')
```



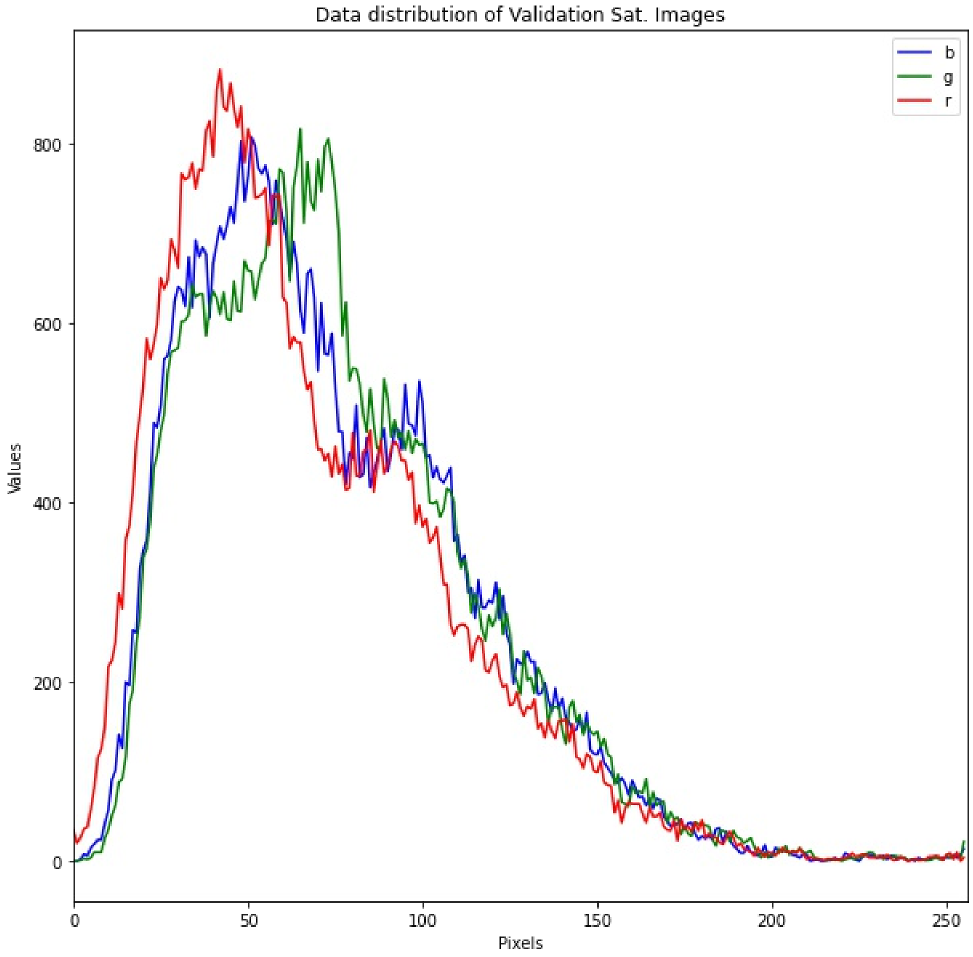
Observation:

> - most of pixels for satellite are concentrated in region 40 to 50.
> - Graph is skewed to right.

Let's compare the train distribution to validation

In [108]:

```
plot_dist(val_sat_images, 'Validation Sat.')
```



> Observation:
>
> - Both the distributions are similar in nature.
> - Validation dist. is also skewed to right.