```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import sys
import os
import subprocess

from six import string_types

import numpy as np
import pandas as pd
from tqdm import tqdm
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import scipy
from skimage import io
from scipy import ndimage
from IPython.display import display
%matplotlib inline

import tensorflow as tf
from tensorflow import keras
```

```
file1 = '/content/drive/MyDrive/Colab Notebooks/Understanding the Amazon from space - Kaggle/train_v2.csv/train_v2.csv'
train = pd.read_csv(file1)
```

```
train.head()
```

	image_name	tags
0	train_0	haze primary
1	train_1	agriculture clear primary water
2	train_2	clear primary
3	train_3	clear primary
4	train_4	agriculture clear habitation primary road

```
train.shape
```

```
(40479, 2)
```

```
file2 = '/content/drive/MyDrive/Colab Notebooks/Understanding the Amazon from space - Kaggle/test_v2_file_mapping.csv/test_v2_file_mapping.csv'
test = pd.read_csv(file2)
```

```
test.head()
```

	old	new
0	file_4158.tif	file_18830.tif
1	file_1668.tif	file_19164.tif
2	file_2411.tif	file_15749.tif
3	file_16047.tif	file_7531.tif
4	file_1271.tif	file_18347.tif

```
test.shape
```

```
(20522, 2)
```

```
# creating the weather labels
weather_categories = ['partly_cloudy', 'haze', 'cloudy', 'clear']
weather_tag_list = [[weather for weather in tag.split() if weather in weather_categories] for tag in train['tags']]
train['weather_tags'] = [''.join(tag) for tag in weather_tag_list]
train.head()
```

	image_name	tags	weather_tags
0	train_0	haze primary	haze
1	train_1	agriculture clear primary water	clear

```
# Build list with unique labels
label_list = []
for tag_str in train.tags.values:
    labels = tag_str.split(' ')
    for label in labels:
        if label not in label_list:
            label_list.append(label)


# Display label list and length
print(f'There are {len(train)} data samples, with {len(label_list)} possible classes.', '\n')
print(f'The label list includes: ')
labels_dict = dict(zip(range(0,17), label_list))
labels_dict
```

```
There are 40479 data samples, with 17 possible classes.

The label list includes:
{0: 'haze',
 1: 'primary',
 2: 'agriculture',
 3: 'clear',
 4: 'water',
 5: 'habitation',
 6: 'road',
 7: 'cultivation',
 8: 'slash_burn',
 9: 'cloudy',
 10: 'partly_cloudy',
 11: 'conventional_mine',
 12: 'bare_ground',
 13: 'artisinal_mine',
 14: 'blooming',
 15: 'selective_logging',
 16: 'blow_down'}
```

```
# One-hot encode the features
train_tag = train.copy()
for label in label_list:
    train_tag[label] = train_tag['tags'].apply(lambda x: 1 if label in x.split() else 0)

train_tag.head()
```

	image_name	tags	weather_tags	haze	primary	agriculture	clear	water	habitation	road
0	train_0	haze primary	haze	1	1	0	0	0	0	0
1	train_1	agriculture clear primary water	clear	0	1	1	1	1	0	0
2	train_2	clear primary	clear	0	1	0	1	0	0	0
3	train_3	clear primary	clear	0	1	0	1	0	0	0
4	train_4	agriculture clear habitation primary road	clear	0	1	1	1	0	1	1

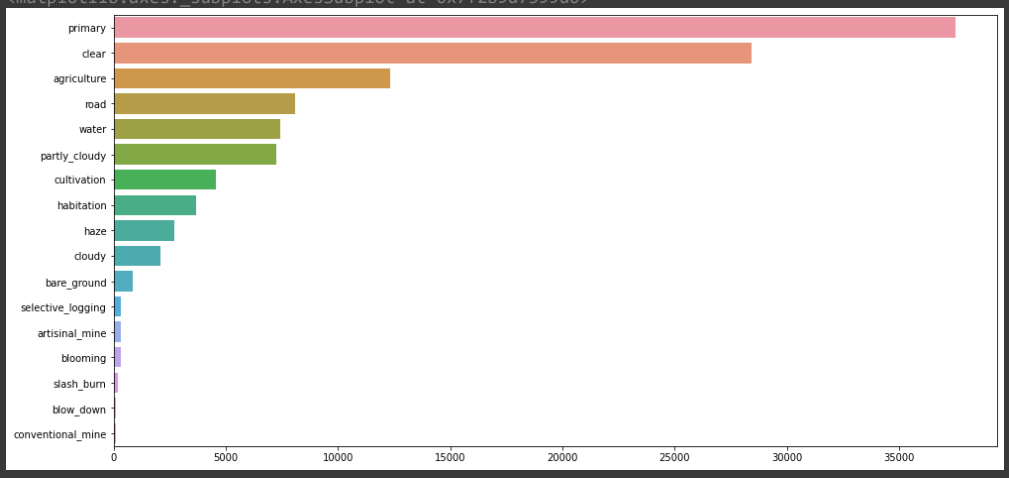
```
#print all unique tags
from itertools import chain
label_list = list(chain.from_iterable([tags.split(" ") for tags in train_tag['tags'].values]))
label_set = set(label_list)
print(f"There are {len(label_set)} unique labels", '\n')
print(f'These unique label sets are: ')
labels_set = dict(zip(range(0,17), label_set))
labels_set
```

```
There are 17 unique labels
```

```
These unique label sets are:
{0: 'clear',
 1: 'selective_logging',
 2: 'partly_cloudy',
 3: 'blow_down',
 4: 'cultivation',
 5: 'blooming',
 6: 'road',
 7: 'agriculture',
 8: 'haze',
 9: 'conventional_mine',
 10: 'bare_ground',
 11: 'slash_burn',
 12: 'primary',
 13: 'artisinal_mine',
 14: 'water',
 15: 'habitation',
 16: 'cloudy'}
```

```
#Histogram of label instances
tag_labels = pd.Series(label_list).value_counts()
fig, ax = plt.subplots(figsize=(16, 8))
sns.barplot(x=tag_labels, y=tag_labels.index, orient='h')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2b9d7399d0>
```



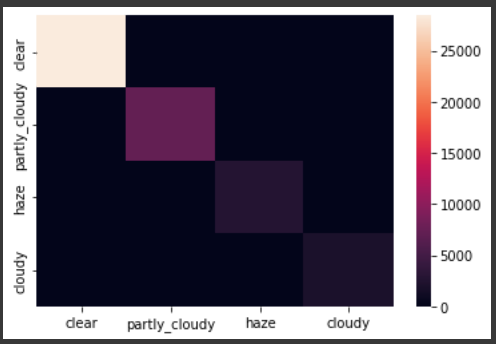

```
#function for cooocurence matrix plotting
def make_cooccurence_matrix(labels):
  num_data = train_tag[labels];
  c_matrix = num_data.T.dot(num_data)
  sns.heatmap(c_matrix)
  return c_matrix

#compute the cooccurence
make_cooccurence_matrix(label_set)
```

	clear	selective_logging	partly_cloudy	blow_down	cultivation	blooming	road
clear	28431	308	0	85	3527	311	6295
selective_logging	308	340	27	1	58	7	151
partly_cloudy	0	27	7261	13	748	17	1382
blow_down	85	1	13	98	8	1	2
cultivation	3527	58	748	8	4477	35	1294
blooming	311	7	17	1	35	332	10
road	6295	151	1382	2	1294	10	8071
agriculture	9150	65	2493	22	3377	32	6034
haze	0	5	0	0	202	4	394
conventional_mine	70	0	28	0	4	0	59
bare_ground	747	13	74	4	89	3	323
slash_burn	173	2	33	2	126	2	36
primary	27668	340	7175	98	4455	332	7728
artisinal_mine	307	6	27	0	18	0	110
water	5502	49	1295	3	868	16	2125
habitation	3090	13	441	3	895	4	2786

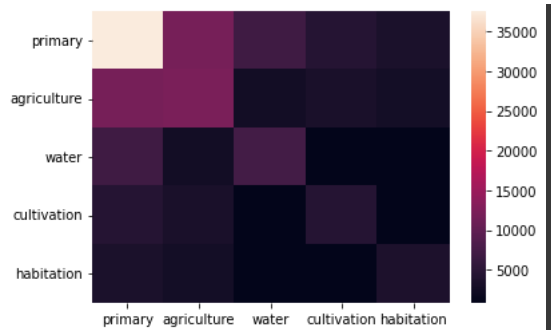
```
# plot weather element cooccurence matrix
weather_labels = ['clear', 'partly_cloudy', 'haze', 'cloudy']
make_cooccurence_matrix(weather_labels)
```

	clear	partly_cloudy	haze	cloudy
clear	28431	0	0	0
partly_cloudy	0	7261	0	0
haze	0	0	2697	0
cloudy	0	0	0	2089




```
# plot land-use element classes cooccurence matrix
land_labels = ['primary', 'agriculture', 'water', 'cultivation', 'habitation']
make_cooccurence_matrix(land_labels)
```

	primary	agriculture	water	cultivation	habitation
primary	37513	11972	7001	4455	3469
agriculture	11972	12315	2712	3377	2737
water	7001	2712	7411	868	915
cultivation	4455	3377	868	4477	895
habitation	3469	2737	915	895	3660



```
# for the analysis we need columns after tag and image_name
train_tag_columns = list(train_tag.columns[2:])
print(train_tag_columns,end='')
```

```
['weather_tags', 'haze', 'primary', 'agriculture', 'clear', 'water', 'habitation', 'road', 'cultivation', 'slash_burn', 'cloudy', 'partly_cloudy', 'cor
```

```
#onehotencode the image name
train_tag['image_name'] = train_tag['image_name'].apply(lambda x:
                                                        f'{x}.jpg')
train_tag.head()
```

	image_name	tags	weather_tags	haze	primary	agriculture	clear	water	habitation	road
0	train_0.jpg	haze primary	haze	1	1	0	0	0	0	0
1	train_1.jpg	agriculture clear primary water	clear	0	1	1	1	1	0	0
2	train_2.jpg	clear primary	clear	0	1	0	1	0	0	0
3	train_3.jpg	clear primary	clear	0	1	0	1	0	0	0
4	train_4.jpg	agriculture clear habitation primary road	clear	0	1	1	1	0	1	1

```
image_train_path = '/content/drive/MyDrive/Colab Notebooks/Understanding the Amazon from space - Kaggle/planet/train-jpg'
```

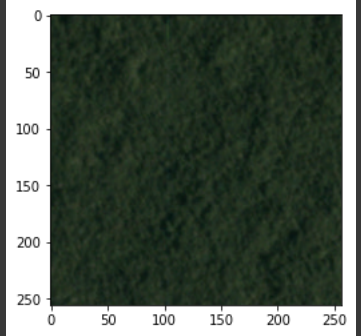
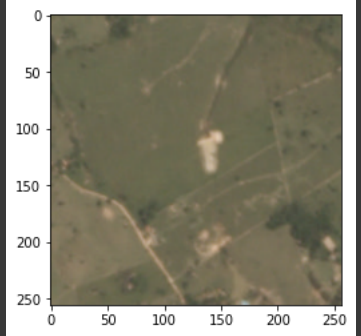
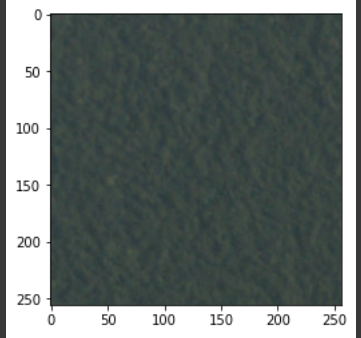
```
image_train_path
```

```
'/content/drive/MyDrive/Colab Notebooks/Understanding the Amazon from space - Kaggle/planet/train
-jpg'
```

```
dir = tf.data.Dataset.list_files(image_train_path + '/*')
```

```
image_train_path = '/content/drive/MyDrive/Colab Notebooks/Understanding the Amazon from space - Kaggle/planet/train-jpg'
```

```
for filename in os.listdir(image_train_path):
  if filename.endswith(".jpg") or filename.endswith(".png"):
    img = plt.imread(os.path.join(image_train_path, filename))
    plt.imshow(img)
    plt.show()
```


Data pre-processing

```
#Determine if length of the train and test dataset csv file equals the actual number of images in the folder
import pathlib
#train path
train_image_dir = pathlib.Path(image_train_path)
train_img_path = sorted(list(train_image_dir.glob('*.jpg')))

#test path
test_img_dir = pathlib.Path('/content/drive/MyDrive/Colab Notebooks/Understanding the Amazon from space - Kaggle/planet/test-jpg')
test_img_path = sorted(list(test_img_dir.glob('*.jpg')))

#additional test path
test_add_img_dir = pathlib.Path('test-jpg-additional')
test_add_img_path = sorted(list(test_add_img_dir.glob('*/*.jpg')))
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-24-8fa64bc2fbae> in <module>
      7 #test path
      8 test_img_dir = pathlib.Path('/content/drive/MyDrive/Colab Notebooks/Understanding the
Amazon from space - Kaggle/planet/test-jpg')
----> 9 test_img_path = sorted(list(test_img_dir.glob('*.jpg')))
     10
     11 #additional test path

 1 frames
/usr/lib/python3.8/pathlib.py in _select_from(self, parent_path, is_dir, exists, scandir)
    532     def _select_from(self, parent_path, is_dir, exists, scandir):
    533         try:
--> 534             with scandir(parent_path) as scandir_it:
    535                 entries = list(scandir_it)
    536             for entry in entries:

OSError: [Errno 5] Input/output error: '/content/drive/MyDrive/Colab Notebooks/Understanding the
Amazon from space - Kaggle/planet/test-jpg'
```

Image preprocessing

```
#define input size
input_size = 64
```

```
#create x_train and y_train
x_train = []
y_train = []

for f, tags in tqdm(train_tag.values, miniters=1000):
  img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/Understanding the Amazon from space - Kaggle/planet/train-jpg/{}.jpg' .format(f))
```