In [2]:
```
cd "E:\ME-COMPUTERS\THIRD SEM\ADM\project"
```

```
E:\ME-COMPUTERS\THIRD SEM\ADM\project
```

In [3]:
```
import numpy as np
import pandas as pd
import sys, requests, shutil, os

data=pd.read_csv("train.csv")
data_test=pd.read_csv("test.csv")
```

In [4]:
```
landmark_list = [str(x) for x in list(range(1000,7000))]
data_sample = data[data['landmark_id'].isin(landmark_list)]
counts=data_sample.landmark_id.value_counts()
sum(data_sample.landmark_id.value_counts())
```

Out[4]:
```
523603
```

In [5]:
```
import re
TARGET_SIZE = 96 #imports images of resolution 96x96

'''change URLs to resize images to target size'''
def overwrite_urls(df):
    def reso_overwrite(url_tail, reso=TARGET_SIZE):
        pattern = 's[0-9]+'
        search_result = re.match(pattern, url_tail)
        if search_result is None:
            return url_tail
        else:
            return 's{}'.format(reso)

    def join_url(parsed_url, s_reso):
        parsed_url[-2] = s_reso
        return '/'.join(parsed_url)

    df = df[df.url.apply(lambda x: len(x.split('/'))>1)]
    parsed_url = df.url.apply(lambda x: x.split('/'))
    train_url_tail = parsed_url.apply(lambda x: x[-2])
    resos = train_url_tail.apply(lambda x: reso_overwrite(x, reso=TARGE
T_SIZE))

    overwritten_df = pd.concat([parsed_url, resos], axis=1)
    overwritten_df.columns = ['url', 's_reso']
    df['url'] = overwritten_df.apply(lambda x: join_url(x['url'], x['s_
reso']), axis=1)
    return df

data_sample_resize = overwrite_urls(data_sample)
print ('1. URLs overwritten')

'''Split to test and train'''
data_test = pd.DataFrame(columns = ['id','url','landmark_id'])
data_training_all = pd.DataFrame(columns = ['id','url','landmark_id'])
percent_test = 0.05

import random
random.seed(42)
for landmark_id in set(data_sample_resize['landmark_id']):
    n=1
    t = data_sample_resize[(data_sample_resize.landmark_id == landmark_
id)] #get all images for a landmark id
    i = 0
    r =[]
    #print(len(t.id))
    while i < len(t.id) and i<6000:
        it = i
        r.append(t.id.iloc[it])  #create a list of all these images
        i += 1

    test = random.sample(r,int(percent_test*len(r))) #randomly pick a s
ample of 1% images from list 'r'
    training = list(set(r) - set(test))  #get the remaining images
    data_t = data_sample_resize[data_sample_resize.id.isin(test)] #hold
```

```
    out dataset
        data_tr = data_sample_resize[data_sample_resize.id.isin(training)]
    #training dataset
        data_test = data_test.append(data_t)
        data_training_all = data_training_all.append(data_tr)
        n+=1

    print ('2. train and test set created')


    '''Split into train and validation set'''
    data_valid = pd.DataFrame(columns = ['id','url','landmark_id'])
    data_train = pd.DataFrame(columns = ['id','url','landmark_id'])
    percent_validation = 0.2 #takes 20% from each class as holdout data
    import random
    random.seed(42)
    for landmark_id in set(data_training_all['landmark_id']):
        n=1
        t = data_training_all[(data_training_all.landmark_id == landmark_i
    d)]
        i = 0
        r =[]
        while i < len(t.id):
            it = i
            r.append(t.id.iloc[it])
            i += 1

        valid = random.sample(r,int(percent_validation*len(r)))
        train = list(set(r) - set(valid))
        data_v = data_training_all[data_training_all.id.isin(valid)]
        data_t = data_training_all[data_training_all.id.isin(train)]
        data_valid = data_valid.append(data_v)
        data_train = data_train.append(data_t)
        n+=1

    print ('3. train and validation set created')
```

```
1. URLs overwritten
2. train and test set created
3. train and validation set created
```

In [ ]:
```
train_sample=data_train.to_csv()
test_sample=data_test.to_csv()
valdi_sample=data_test.to_csv()
```

Create directories 'train_images_model', 'validation_images_model', 'test_images_from_train' before running the code ahead.

In [ ]:
```
def fetch_image(path,folder):
    try:
        url=path
        response=requests.get(url, stream=True)
        with open(folder + '/image.jpg', 'wb') as out_file:
            shutil.copyfileobj(response.raw, out_file)
        del response
    except:
        print("error")
'''TRAIN SET - fetch images for the resized URLs and save in the alread
y created directory train_images_model'''
i=0
for link in data_train['url']:
    if i%10000==0:
        print(i)
    #looping over links to get images
    if os.path.exists('train_images_model/'+str(data_train['id'].iloc
[i])+'.jpg'):
        i+=1
        print(i)
        continue
    fetch_image(link,'train_images_model')
    try:
        os.rename('train_images_model/image.jpg','train_images_model/'+
str(data_train['id'].iloc[i])+ '.jpg')
    except:
        print("not found")
    i+=1
#     if(i==50):   #uncomment to test in your machine
#         break
print('4. train images fetched')
i=0
for link in data_valid['url']:              #looping over links to get
images
    if os.path.exists('validation_images_model/'+str(data_valid['id'].i
loc[i])+'.jpg'):
        i+=1
        continue
    fetch_image(link,'validation_images_model')
    try:
        os.rename('validation_images_model/image.jpg','validation_image
s_model/'+ str(data_valid['id'].iloc[i])+ '.jpg')
    except:
        print("not found")
    i+=1
#     if(i==50):   #uncomment to test in your machine
#         break
print('5. Validation images fetched')

i=0
for link in data_test['url']:              #looping over links to get i
mages
    if os.path.exists('test_images_from_train/'+str(data_test['id'].ilo
c[i])+'.jpg'):
        i+=1
```

```
        continue
    fetch_image(link,'test_images_from_train')
    try:
        os.rename('test_images_from_train/image.jpg','test_images_from_
train/'+ str(data_test['id'].iloc[i])+ '.jpg')
    except:
        print("not found")
    i+=1
#     if(i==50):   #uncomment to test in your machine
#         break
print('6. Test images fetched')
```

Data Preprocessing

Creating folders for each landmark ID (Class label)

In [ ]:
```
##create folders for landmark IDs in Training folder
import pandas as pd
import os
import shutil
from shutil import copyfile
import urllib

train_data = data_train

temp = pd.DataFrame(data_train.landmark_id.value_counts())
temp.reset_index(inplace=True)
temp.columns = ['landmark_id','count']

def createfolders(dataset,folder):
    i = 0
    while i < len(dataset):
        landmark = str(dataset.landmark_id.iloc[i])
        path = folder + '/'+ landmark
        if not os.path.exists(path):
            os.makedirs(path)
        i+=1
createfolders(temp,'train_images_model')
available = [int((x[0].split('/'))[-1]) for x in os.walk(r'train_images
_model/') if len((x[0].split('/'))[-1]) > 0]
new = [str(x) for x in range(1000,6999) if x not in available]
for i in new:
    path = 'train_images_model/' + i
    if not os.path.exists(path):
        os.makedirs(path)
print ('Train folders created')

rootdirpics = r'train_images_model/'
rootdirfolders = r'train_images_model/'

def transformdata(data,path1, path2):

    n = 1
    for landmark_id in set(data['landmark_id']):
        t = data[(data.landmark_id == landmark_id)]
        i = 1
        r =[]
        while i <= len(t.id):
            it = i - 1
            r.append(t.id.iloc[it])
            i += 1
        for files in os.listdir(rootdirpics):    # loop through startfo
lders
            inpath = path1 + files
            folder = str(landmark_id)
            outpath = path2 + folder
            if ((files.split('.')[0] in r) & (os.path.getsize(inpath) >
1000)):
#                 print('move')
                shutil.move(inpath, outpath)
            elif ((files.split('.')[0] in r) & (os.path.getsize(inpath)
```

```
<= 1000)):
                os.remove(inpath)
        n+=1

transformdata(train_data,rootdirpics, rootdirfolders)
print ('Train images moved')
```

In [ ]:
```
##create folders for landmark IDs in Validation folder

temp = pd.DataFrame(data_valid.landmark_id.value_counts())
temp.reset_index(inplace=True)
temp.columns = ['landmark_id','count']
createfolders(temp,'validation_images_model')
print ('Validation folders created')

#make folders for landmark ID which had no images in validation sets - required for codes running next
available = [int((x[0].split('/'))[-1]) for x in os.walk(r'validation_images_model/') if len((x[0].split('/'))[-1]) > 0]
new = [str(x) for x in range(1000,6999) if x not in available]
for i in new:
    path = 'validation_images_model/' + i
    if not os.path.exists(path):
        os.makedirs(path)

rootdirpics = r'validation_images_model/'
rootdirfolders = r'validation_images_model/'
transformdata(data_valid,rootdirpics, rootdirfolders)
print ('Validation images moved')
```

In [ ]:
```
#remove corrupted images from the dataset
def cleaning(dir):
    i = 1000
    count = 0
    while i <= 6999:
        f = str(i)
        print (f)
        for root, dirs, files in os.walk(dir +'/'+ f):  # loop through startfolders
            for pic in files:
                p=dir+'/'+f+'/'+pic
                try:
                    im=Image.open(p)
                except IOError:
                    count+=1
                    print(p)
                    os.remove(p)
            i += 1
    print(count)
```

In [ ]:
```
cleaning("/train_images_model")
cleaning("/validation_images_model")
```