Load data from kaggle we used kaggle api to download intel image classification

About the data sets

Context

This is image data of Natural Scenes around the world.

Content

This Data contains around 25k images of size 150x150 distributed under 6 categories.

{'buildings' -> 0,

'forest' -> 1,

'glacier' -> 2,

'mountain' -> 3,

'sea' -> 4,

'street' -> 5 }

The Train, Test and Prediction data is separated in each zip files. There are around 14k images in Train, 3k in Test and 7k in Prediction.***

Load data from the kaggle

```
#load data set from the kaggle
! pip install -q kaggle
#Make a directory named ".kaggle"
! mkdir ~/.kaggle
from google.colab import files
! cp /content/kaggle.json ~/.kaggle/
! cp /content/kaggle.json ~/.kaggle/
!  kaggle datasets download -d puneet6060/intel-image-classification
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggl
Downloading intel-image-classification.zip to /content
 97% 337M/346M [00:12<00:00, 42.2MB/s]
100% 346M/346M [00:13<00:00, 27.8MB/s]
```

Unzip the data sets

```
#create the directory of the data
! mkdir data


#unzip train data there

! unzip /content/intel-image-classification.zip -d data
```

```
  inflating: data/seg_train/seg_train/street/9484.jpg
  inflating: data/seg_train/seg_train/street/9489.jpg
  inflating: data/seg_train/seg_train/street/9516.jpg
  inflating: data/seg_train/seg_train/street/9521.jpg
  inflating: data/seg_train/seg_train/street/9530.jpg
  inflating: data/seg_train/seg_train/street/9535.jpg
  inflating: data/seg_train/seg_train/street/9581.jpg
  inflating: data/seg_train/seg_train/street/9595.jpg
  inflating: data/seg_train/seg_train/street/9600.jpg
  inflating: data/seg_train/seg_train/street/9609.jpg
  inflating: data/seg_train/seg_train/street/9649.jpg
  inflating: data/seg_train/seg_train/street/9652.jpg
  inflating: data/seg_train/seg_train/street/9655.jpg
  inflating: data/seg_train/seg_train/street/9662.jpg
  inflating: data/seg_train/seg_train/street/9685.jpg
  inflating: data/seg_train/seg_train/street/9687.jpg
  inflating: data/seg_train/seg_train/street/9690.jpg
  inflating: data/seg_train/seg_train/street/9693.jpg
  inflating: data/seg_train/seg_train/street/970.jpg
  inflating: data/seg_train/seg_train/street/9704.jpg
  inflating: data/seg_train/seg_train/street/9707.jpg
  inflating: data/seg_train/seg_train/street/971.jpg
```

```
inflating: data/seg_train/seg_train/street/9717.jpg
inflating: data/seg_train/seg_train/street/9720.jpg
inflating: data/seg_train/seg_train/street/9733.jpg
inflating: data/seg_train/seg_train/street/9734.jpg
inflating: data/seg_train/seg_train/street/974.jpg
inflating: data/seg_train/seg_train/street/9744.jpg
inflating: data/seg_train/seg_train/street/9749.jpg
inflating: data/seg_train/seg_train/street/9760.jpg
inflating: data/seg_train/seg_train/street/9770.jpg
inflating: data/seg_train/seg_train/street/9798.jpg
inflating: data/seg_train/seg_train/street/9817.jpg
inflating: data/seg_train/seg_train/street/9843.jpg
inflating: data/seg_train/seg_train/street/9846.jpg
inflating: data/seg_train/seg_train/street/9847.jpg
inflating: data/seg_train/seg_train/street/9863.jpg
inflating: data/seg_train/seg_train/street/9867.jpg
inflating: data/seg_train/seg_train/street/9872.jpg
inflating: data/seg_train/seg_train/street/9875.jpg
inflating: data/seg_train/seg_train/street/9878.jpg
inflating: data/seg_train/seg_train/street/9879.jpg
inflating: data/seg_train/seg_train/street/9885.jpg
inflating: data/seg_train/seg_train/street/9891.jpg
inflating: data/seg_train/seg_train/street/9895.jpg
inflating: data/seg_train/seg_train/street/9903.jpg
inflating: data/seg_train/seg_train/street/9911.jpg
inflating: data/seg_train/seg_train/street/992.jpg
inflating: data/seg_train/seg_train/street/9926.jpg
inflating: data/seg_train/seg_train/street/9931.jpg
inflating: data/seg_train/seg_train/street/9934.jpg
inflating: data/seg_train/seg_train/street/994.jpg
inflating: data/seg_train/seg_train/street/9953.jpg
inflating: data/seg_train/seg_train/street/9959.jpg
inflating: data/seg_train/seg_train/street/9961.jpg
inflating: data/seg_train/seg_train/street/9967.jpg
inflating: data/seg_train/seg_train/street/9978.jpg
inflating: data/seg_train/seg_train/street/9989.jpg
inflating: data/seg_train/seg_train/street/999.jpg
```

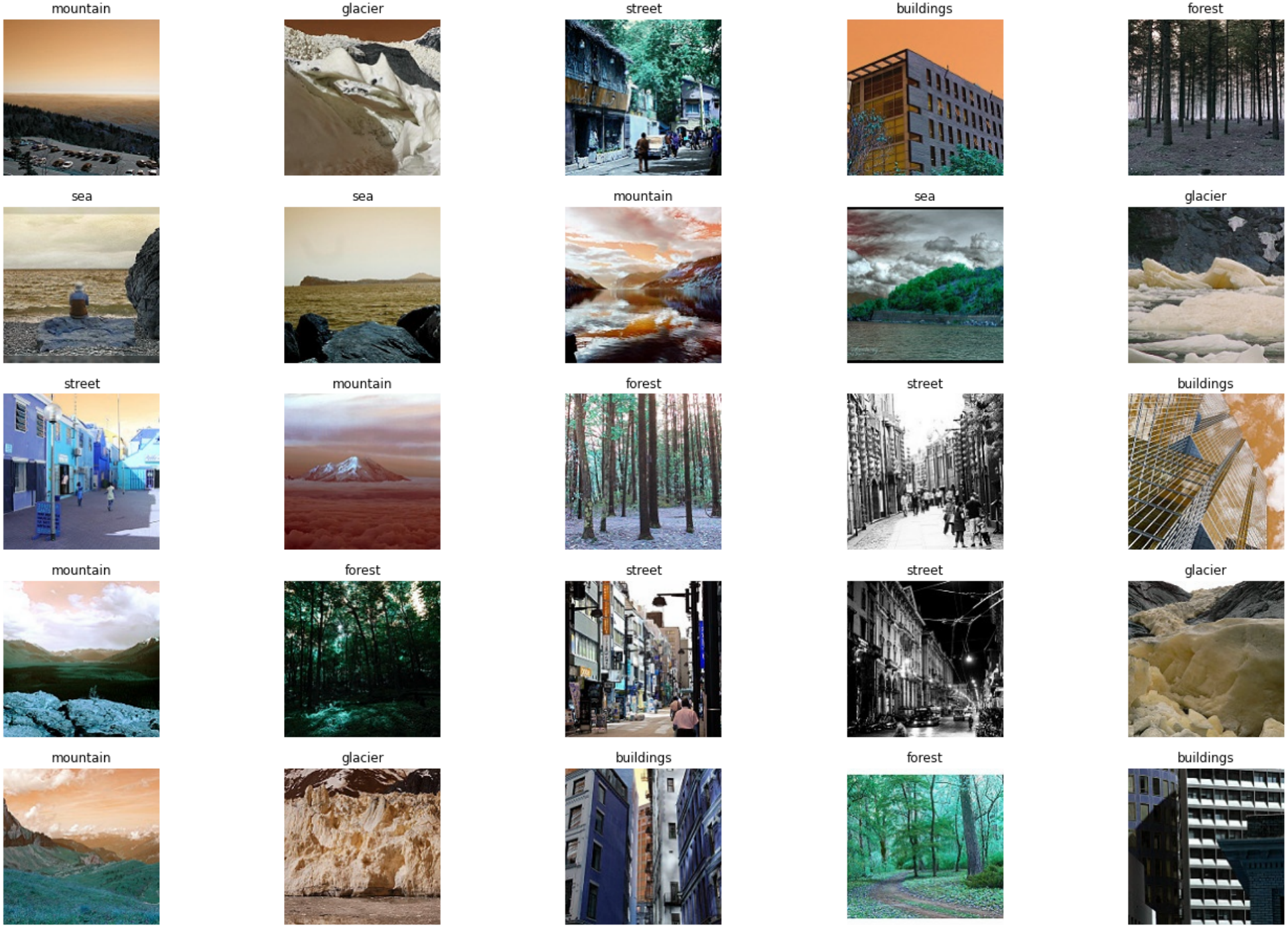
```
#import library
import tensorflow.keras.layers as Layers
import tensorflow.keras.activations as Actications
```

```
import tensorflow.keras.models as Models
import tensorflow.keras.optimizers as Optimizer
import tensorflow.keras.metrics as Metrics
import tensorflow.keras.utils as Utils
from keras.utils.vis_utils import model_to_dot
import os
import matplotlib.pyplot as plot
import cv2
import numpy as np
from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix as CM
from random import randint
import matplotlib.gridspec as gridspec
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt


def get_images(file_directory):
    Images = []
    Labels = []
    label = 0
    for labels in os.listdir(file_directory):
        if labels == 'glacier':
            label = 2
        elif labels == 'sea':
            label = 4
        elif labels == 'buildings':
            label = 0
        elif labels == 'forest':
            label = 1
        elif labels == 'street':
            label = 5
        elif labels == 'mountain':
            label = 3
        for image_file in os.listdir(file_directory+labels):
           #Extracting the file name of the image from Class Label folder
            image = cv2.imread(file_directory+labels+r'/'+image_file)
            #Reading the image (OpenCV)
```

```
            image = cv2.resize(image,(150,150))
            #Resize the image, Some images are different sizes. (Resizing is very Important)
            Images.append(image)
            Labels.append(label)
    return shuffle(Images,Labels,random_state=0) #Shuffle the dataset you just prepared.


def get_classlabel(class_code):
    labels = {2:'glacier', 4:'sea', 0:'buildings', 1:'forest', 5:'street', 3:'mountain'}

    return labels[class_code]
Images, Labels = get_images('data/seg_train/seg_train/')
 #Extract the training images

Images = np.array(Images)
 #converting the list of images to numpy array
Labels = np.array(Labels)


print("Shape of Images:",Images.shape)
print("Shape of Labels:",Labels.shape)
```

```
Shape of Images: (14034, 150, 150, 3)
Shape of Labels: (14034,)
```

```
f,ax = plot.subplots(5,5)
f.subplots_adjust(0,0,3,3)
for i in range(0,5,1):
    for j in range(0,5,1):
        rnd_number = randint(0,len(Images))
        ax[i,j].imshow(Images[rnd_number])
        ax[i,j].set_title(get_classlabel(Labels[rnd_number]))
        ax[i,j].axis('off')
```


mountain
glacier
street
buildings
forest
sea
sea
mountain
sea
glacier
street
mountain
forest
street
buildings
mountain
forest
street
street
glacier
mountain
glacier
buildings
forest
buildings

```
model = Models.Sequential()

model.add(Layers.Conv2D(200,kernel_size=(3,3),activation='relu',input_shape=(150,150,3)))
model.add(Layers.Conv2D(180,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Conv2D(180,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(140,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(100,kernel_size=(3,3),activation='relu'))
model.add(Layers.Conv2D(50,kernel_size=(3,3),activation='relu'))
model.add(Layers.MaxPool2D(5,5))
model.add(Layers.Flatten())
model.add(Layers.Dense(180,activation='relu'))
model.add(Layers.Dense(100,activation='relu'))
model.add(Layers.Dense(50,activation='relu'))
model.add(Layers.Dropout(rate=0.5))
model.add(Layers.Dense(6,activation='softmax'))

model.compile(optimizer=Optimizer.Adam(lr=0.0001),loss='sparse_categorical_crossentropy',metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 200)     5600
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 146, 146, 180)     324180
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 29, 29, 180)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 27, 27, 180)       291780
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 25, 25, 140)       226940
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 23, 23, 100)       126100
_________________________________________________________________
```

```
conv2d_5 (Conv2D)            (None, 21, 21, 50)        45050
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 50)          0
_________________________________________________________________
flatten (Flatten)            (None, 800)               0
_________________________________________________________________
dense (Dense)                (None, 180)               144180
_________________________________________________________________
dense_1 (Dense)              (None, 100)               18100
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 306
=================================================================
Total params: 1,187,286
Trainable params: 1,187,286
Non-trainable params: 0
_________________________________________________________________
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/optimizer_v2.py:356: UserWarning: The `lr` argument is deprecated, us
  "The `lr` argument is deprecated, use `learning_rate` instead.")
```

```
train_data = model.fit(Images,Labels,epochs=5,validation_split=0)
```

```
Epoch 1/5
439/439 [==============================] - 241s 549ms/step - loss: 1.1474 - accuracy: 0.5613
Epoch 2/5
439/439 [==============================] - 234s 534ms/step - loss: 0.9514 - accuracy: 0.6526
Epoch 3/5
439/439 [==============================] - 234s 534ms/step - loss: 0.8191 - accuracy: 0.7165
Epoch 4/5
439/439 [==============================] - 234s 533ms/step - loss: 0.7315 - accuracy: 0.7505
Epoch 5/5
439/439 [==============================] - 234s 533ms/step - loss: 0.6599 - accuracy: 0.7714
```

Here we get 77 percent accuracy if we increase the epoch accuracy also increase

**Thank you* * link off the colab:- (https://colab.research.google.com/drive/1lKhqmF-kmKZhtesian-sO5qgnc0l3hF9?usp=sharing)

0s completed at 2:28 AM