```
print("Bismillah")
```

```
Bismillah
time: 1.35 ms (started: 2021-01-05 20:20:07 +00:00)
```

Imports

```
!pip install ipython-autotime
```

```
Requirement already satisfied: ipython-autotime in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: ipython in /usr/local/lib/python3.6/dist-packages (from i
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from p
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/dist-package
time: 2.07 s (started: 2021-01-05 20:20:08 +00:00)
```

```
# necessary imports
import os
import cv2
import numpy as np
from imutils import paths
from sklearn.preprocessing import LabelBinarizer
from tqdm import tqdm

import matplotlib.pyplot as plt
%matplotlib inline

from google.colab.patches import cv2_imshow

%load_ext autotime
```

```
The autotime extension is already loaded. To reload it, use:
  %reload_ext autotime
time: 7.14 ms (started: 2021-01-05 20:20:10 +00:00)
```

```
img_width = 150
img_width = 150
```

```
time: 696 µs (started: 2021-01-05 20:20:17 +00:00)
```

VGG Model

```
from keras.applications.vgg16 import VGG16
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten
```

```
time: 1.34 s (started: 2021-01-05 15:45:15 +00:00)
```

```
# load VGG16 model without classification layers
model = VGG16(include_top=False, input_shape=(150, 150, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16
58892288/58889256 [==============================] - 0s 0us/step
time: 6.02 s (started: 2021-01-05 15:45:16 +00:00)
```

```
# add new classification layers
flat1 = Flatten()(model.layers[-1].output) # flatten last layer
class1 = Dense(1024, activation='relu')(flat1) # add FC layer on previous layer
output = Dense(6, activation='softmax')(class1) # add softmax layer
```

```
time: 22 ms (started: 2021-01-05 15:45:22 +00:00)
```

```
# define the new model
model = Model(inputs=model.inputs, outputs=output)
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
```

```
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              8389632
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 6150
=================================================================
Total params: 23,110,470
Trainable params: 23,110,470
Non-trainable params: 0
_________________________________________________________________
time: 15.8 ms (started: 2021-01-05 15:45:22 +00:00)
```

Loading Data

```
# !unzip "/content/drive/MyDrive/CV/Assignment 3/intel-image-classification.zip" -d "/content
```

```
time: 613 µs (started: 2021-01-05 15:45:22 +00:00)
```

```
# !unzip "/content/drive/MyDrive/CV/Assignment 3/Test_data.zip" -d "/content/drive/MyDrive/CV
```

```
time: 514 µs (started: 2021-01-05 15:45:22 +00:00)
```

```
# A function to load data from a given directory
def load_data(data_dir):
  data = []
  labels = []
  class_dirs = os.listdir(data_dir)
```

```
  for direc in class_dirs:
    # i=0
    class_dir = os.path.join(data_dir, direc)
    for imagepath in tqdm(list(paths.list_images(class_dir))):
      image = cv2.imread(imagepath)
      image = cv2.resize(image, (img_width, img_height))  # incase images not of same size
      data.append(image)
      labels.append(direc)
      # i = i+1
      # if (i==10):
      #   break
  # normalizing and converting to numpy array format
  data = np.array(data, dtype='float')/255.0
  labels = np.array(labels)
  return data, labels
```

```
    time: 6.63 ms (started: 2021-01-05 20:25:46 +00:00)
```

```
train_dir = "/content/drive/MyDrive/CV/Assignment 3/seg_train/seg_train/"
test_dir = "/content/drive/MyDrive/CV/Assignment 3/seg_test/seg_test/"
pred_dir = "/content/drive/MyDrive/CV/Assignment 3/pred/seg_pred/seg_pred/"
```

```
    time: 5.72 ms (started: 2021-01-05 15:45:22 +00:00)
```

```
img_height = 150
img_width = 150
batch_size = 32
nb_epochs = 10
```

```
    time: 3.99 ms (started: 2021-01-05 15:45:22 +00:00)
```

Compile the model

```
from keras.optimizers import SGD
sgd = SGD(lr=0.001, decay=1e-7, momentum=.9)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])
```

```
    time: 29.6 ms (started: 2021-01-05 15:45:22 +00:00)
```

```
from keras.preprocessing.image import ImageDataGenerator
```

```
    time: 3.74 ms (started: 2021-01-05 15:45:22 +00:00)
```

```
train_datagen = ImageDataGenerator(rescale=1./255,
    shear_range=0.2,
```

```
    zoom_range=0.2,
    horizontal_flip=True
    ,validation_split=0.3) # set validation split
```

```
 time: 3.17 ms (started: 2021-01-05 15:45:22 +00:00)
```

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training') # set as training data
```

```
 Found 9825 images belonging to 6 classes.
 time: 32.1 s (started: 2021-01-05 15:45:22 +00:00)
```

```
validation_generator = train_datagen.flow_from_directory(
    test_dir, # directory for validation data
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation') # set as validation data
```

```
 Found 898 images belonging to 6 classes.
 time: 3.69 s (started: 2021-01-05 15:45:54 +00:00)
```

```
validation_generator[0]
```

```
              ...,

              [[2.69284714e-02, 2.30069030e-02, 7.66896736e-03],
               [7.24770362e-03, 3.32613476e-03, 1.10871170e-03],
               [2.30019726e-02, 1.90804042e-02, 8.17731675e-03],
               ...,
               [3.48340633e-04, 4.26990958e-03, 0.00000000e+00],
               [6.38608541e-03, 7.11461157e-03, 6.38608541e-03],
               [1.29463412e-02, 1.29463412e-02, 1.29463412e-02]],

              [[2.69284714e-02, 2.30069030e-02, 7.66896736e-03],
               [7.24770362e-03, 3.32613476e-03, 1.10871170e-03],
               [2.30019726e-02, 1.90804042e-02, 8.17731675e-03],
               ...,
               [3.48340633e-04, 4.26990958e-03, 0.00000000e+00],
               [6.38608541e-03, 7.11461157e-03, 6.38608541e-03],
               [1.29463412e-02, 1.29463412e-02, 1.29463412e-02]],

              [[2.69284714e-02, 2.30069030e-02, 7.66896736e-03],
               [7.24770362e-03, 3.32613476e-03, 1.10871170e-03],
               [2.30019726e-02, 1.90804042e-02, 8.17731675e-03],
               ...,
               [3.48340633e-04, 4.26990958e-03, 0.00000000e+00],
               [6.38608541e-03, 7.11461157e-03, 6.38608541e-03],
               [1.29463412e-02, 1.29463412e-02, 1.29463412e-02]]]],
```

```
dtype=float32), array([[1., 0., 0., 0., 0., 0.],
 [0., 0., 0., 0., 0., 1.],
 [0., 0., 1., 0., 0., 0.],

 [0., 0., 0., 0., 0., 1.],
 [0., 0., 0., 0., 1., 0.],
 [0., 0., 0., 0., 1., 0.],
 [0., 0., 0., 0., 1., 0.],
 [0., 0., 1., 0., 0., 0.],
 [0., 0., 1., 0., 0., 0.],
 [0., 0., 0., 0., 0., 1.],
 [0., 0., 0., 0., 1., 0.],
 [0., 0., 0., 0., 0., 1.],
 [0., 0., 0., 0., 1., 0.],
 [0., 0., 0., 0., 0., 1.],
 [0., 0., 0., 0., 0., 1.],
 [0., 0., 0., 0., 0., 1.],
 [0., 0., 0., 0., 1., 0.],
 [1., 0., 0., 0., 0., 0.],
 [0., 0., 0., 1., 0., 0.],
 [0., 1., 0., 0., 0., 0.],
 [1., 0., 0., 0., 0., 0.],
 [0., 0., 0., 0., 0., 1.],
 [1., 0., 0., 0., 0., 0.],
 [0., 0., 1., 0., 0., 0.],
 [0., 0., 1., 0., 0., 0.],
 [0., 0., 1., 0., 0., 0.],
 [1., 0., 0., 0., 0., 0.],
 [0., 0., 0., 0., 0., 1.],
 [1., 0., 0., 0., 0., 0.],
 [1., 0., 0., 0., 0., 0.],
 [0., 0., 0., 0., 1., 0.],
 [0., 0., 1., 0., 0., 0.]], dtype=float32))time: 193 ms (started: 2021-01-05 2
```

```
H = model.fit(
    train_generator,
    steps_per_epoch = train_generator.samples // batch_size,
    validation_data = validation_generator,
    validation_steps = validation_generator.samples // batch_size,
    epochs = nb_epochs)
```

```
Epoch 1/10
307/307 [==============================] - 4075s 13s/step - loss: 0.7479 - accuracy: 0.7
Epoch 2/10
307/307 [==============================] - 87s 281ms/step - loss: 0.2799 - accuracy: 0.9
Epoch 3/10
307/307 [==============================] - 87s 282ms/step - loss: 0.2258 - accuracy: 0.9
Epoch 4/10
307/307 [==============================] - 86s 282ms/step - loss: 0.2309 - accuracy: 0.9
Epoch 5/10
307/307 [==============================] - 86s 280ms/step - loss: 0.1641 - accuracy: 0.9
Epoch 6/10
307/307 [==============================] - 86s 280ms/step - loss: 0.1394 - accuracy: 0.9
Epoch 7/10
307/307 [==============================] - 87s 282ms/step - loss: 0.1255 - accuracy: 0.9
```

```
Epoch 8/10
307/307 [==============================] - 87s 282ms/step - loss: 0.1051 - accuracy: 0.9
Epoch 9/10
307/307 [==============================] - 87s 281ms/step - loss: 0.0990 - accuracy: 0.9
Epoch 10/10
307/307 [==============================] - 87s 282ms/step - loss: 0.0872 - accuracy: 0.9
time: 1h 21min 7s (started: 2021-01-05 15:45:58 +00:00)
```

```
# save the model's trained weights
model.save_weights('/content/drive/MyDrive/CV/Assignment 3/vgg_aug_transfer_trained_wts.h5')
```

```
time: 3.56 s (started: 2021-01-05 20:27:45 +00:00)
```

```
model.load_weights('/content/drive/MyDrive/CV/Assignment 3/vgg_aug_transfer_trained_wts.h5')
```

```
time: 194 ms (started: 2021-01-05 20:27:51 +00:00)
```

```
# print('loading train images')
# X_train, y_train = load_data(train_dir)
# print('loading test images')
# X_test, y_test = load_data(test_dir)
```

```
time: 2.17 ms (started: 2021-01-05 17:07:06 +00:00)
```

```
test_datagen = ImageDataGenerator()
```

```
time: 1.2 ms (started: 2021-01-05 18:29:25 +00:00)
```

```
test_generator = test_datagen.flow_from_directory(
    pred_dir, # directory for prediction data
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='Prediction') # set as validation data
```

```
print('loading pred images')
X_test, y_test = load_data(pred_dir)
```

```
  3%|▌          | 37/1330 [00:00<00:03, 366.13it/s]loading pred images
100%|██████████| 1330/1330 [00:03<00:00, 359.27it/s]
100%|██████████| 1297/1297 [00:03<00:00, 361.02it/s]
100%|██████████| 1128/1128 [00:03<00:00, 350.17it/s]
100%|██████████| 1166/1166 [00:03<00:00, 343.95it/s]
100%|██████████| 1236/1236 [00:03<00:00, 356.69it/s]
100%|██████████| 1144/1144 [00:03<00:00, 351.41it/s]
time: 22.5 s (started: 2021-01-05 20:26:42 +00:00)
```

```
lb = LabelBinarizer()
```

```
# y_train = lb.fit_transform(y_train)
y_test = lb.fit_transform(y_test)
```

```
time: 7.89 ms (started: 2021-01-05 20:27:40 +00:00)
```

```
# from sklearn.model_selection import train_test_split
# (X_train, X_valid, y_train, y_valid) = train_test_split(X_train, y_train, test_size=0.2, ra
```

```
time: 1.69 ms (started: 2021-01-05 17:07:06 +00:00)
```

Train the model

```
# H = model.fit(X_train, y_train, batch_size=128,
#               epochs=10,
#               validation_data=(X_valid, y_valid))
```

```
time: 1.2 ms (started: 2021-01-05 17:07:06 +00:00)
```

```
score = model.evaluate(X_test, y_test, batch_size=64)
print('Test Loss = ', score[0])
print('Test Accuracy = ', score[1])
```

```
115/115 [==============================] - 15s 120ms/step - loss: 0.4727 - accuracy: 0.8
Test Loss =  0.4727141559123993
Test Accuracy =  0.8522120118141174
time: 17.7 s (started: 2021-01-05 20:27:57 +00:00)
```

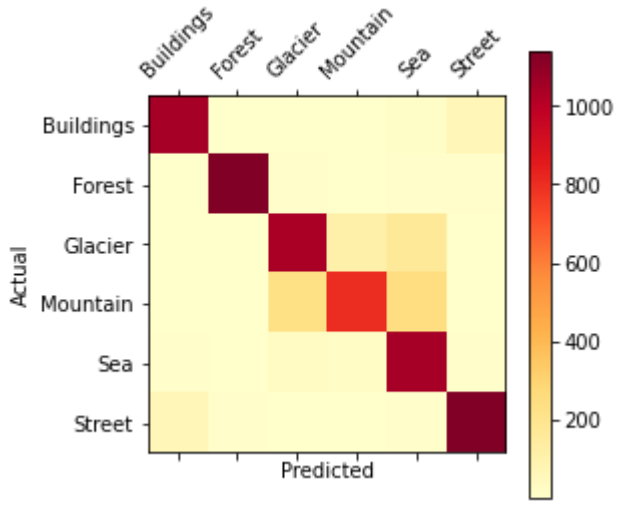
```
'''CONFUSION MATRIX'''
# Making prediction
y_pred = model.predict(X_test)
y_true = np.argmax(y_test, axis=-1)

# Plotting the confusion matrix
from sklearn.metrics import confusion_matrix
confusion_mtx = confusion_matrix(y_true, np.argmax(y_pred, axis=1))
```

```
time: 16.7 s (started: 2021-01-05 20:29:16 +00:00)
```

```
confusion_mtx
```

```
array([[1051,    2,    4,    4,   15,   68],
       [   3, 1133,    8,    2,   13,    7],
       [   2,    3, 1039,  116,  167,    3],
       [   2,    3,  234,  807,  247,    4],
       [   8,    1,   36,   26, 1051,    6],
       [  71,    6,    5,    1,   12, 1141]])time: 4.79 ms (started: 2021-01-05 20:29:47
```

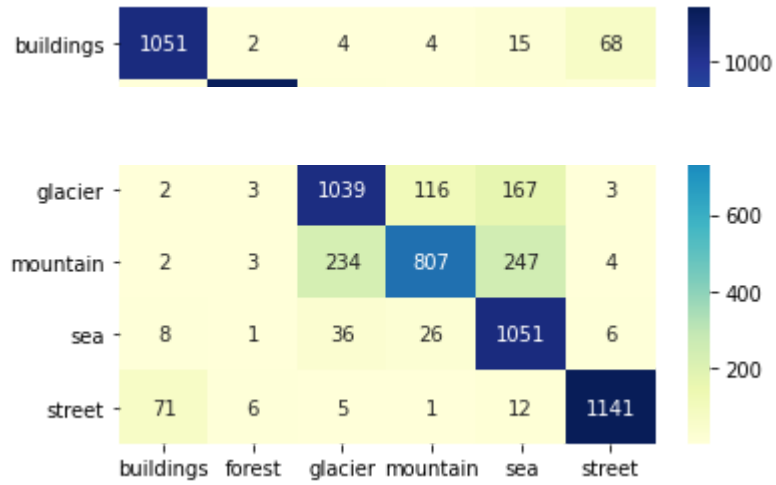
```
def plot_confusion_matrix(df_confusion, title='Confusion matrix', cmap=plt.cm.YlOrRd):
  plt.matshow(df_confusion, cmap=cmap) # imshow
  plt.colorbar()
  tick_marks = np.arange(6)
  names = ["Buildings", "Forest", "Glacier", "Mountain", "Sea", "Street"]
  plt.xticks(tick_marks, names, rotation=45)
  plt.yticks(tick_marks, names)
  plt.ylabel("Actual")
  plt.xlabel("Predicted")
#call function
plot_confusion_matrix(confusion_mtx)
```




```
time: 292 ms (started: 2021-01-05 20:29:51 +00:00)
```

```
import seaborn as sns

class_names = ['buildings','street','forest','glacier','mountain','sea']

class_names = sorted(class_names)

sns.heatmap(confusion_mtx, xticklabels=class_names, yticklabels=class_names,

            annot=True, fmt='d', cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fede8aa99b0>
```




```
time: 250 ms (started: 2021-01-05 20:32:39 +00:00)
```