بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Imports

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

```
Collecting ipython-autotime
  Downloading https://files.pythonhosted.org/packages/d6/c5/013f5aa3b56c6d2c58634bc97977
Requirement already satisfied: ipython in /usr/local/lib/python3.6/dist-packages (from i
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from
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: setuptools>=18.5 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from
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
Installing collected packages: ipython-autotime
Successfully installed ipython-autotime-0.3.0
```

```
# 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
```

```
time: 102 µs (started: 2021-01-06 02:15:54 +00:00)
```

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

```
time: 1.94 ms (started: 2021-01-06 02:15:54 +00:00)
```

InceptionResNetV2 Model

```
from keras.applications import InceptionResNetV2
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten
```

```
time: 1.49 s (started: 2021-01-06 02:15:54 +00:00)
```

```
model = InceptionResNetV2(include_top=False, weights='imagenet', input_shape=(img_width, img_
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/incep
219062272/219055592 [==============================] - 1s 0us/step
time: 11.8 s (started: 2021-01-06 02:15:56 +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
class2 = Dense(1024, activation='relu')(class1) # add FC layer on previous layer
output = Dense(6, activation='softmax')(class2) # add softmax layer
```

```
time: 28.7 ms (started: 2021-01-06 02:16:08 +00:00)
```

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

```
block8_9 (Lambda)               (None, 3, 3, 2080)   0           block8_8_ac[0][0]
                                                                 block8_9_conv[0][0]
__________________________________________________________________________________________
block8_9_ac (Activation)        (None, 3, 3, 2080)   0           block8_9[0][0]
__________________________________________________________________________________________
conv2d_200 (Conv2D)             (None, 3, 3, 192)    399360      block8_9_ac[0][0]
__________________________________________________________________________________________
batch_normalization_200 (BatchN (None, 3, 3, 192)    576         conv2d_200[0][0]
__________________________________________________________________________________________
activation_200 (Activation)     (None, 3, 3, 192)    0           batch_normalization_
__________________________________________________________________________________________
conv2d_201 (Conv2D)             (None, 3, 3, 224)    129024      activation_200[0][0]
__________________________________________________________________________________________
batch_normalization_201 (BatchN (None, 3, 3, 224)    672         conv2d_201[0][0]
__________________________________________________________________________________________
activation_201 (Activation)     (None, 3, 3, 224)    0           batch_normalization_
__________________________________________________________________________________________
conv2d_199 (Conv2D)             (None, 3, 3, 192)    399360      block8_9_ac[0][0]
__________________________________________________________________________________________
```

```
conv2d_202 (Conv2D)             (None, 3, 3, 256)    172032      activation_201[0][0]
__________________________________________________________________________________________
batch_normalization_199 (BatchN (None, 3, 3, 192)    576         conv2d_199[0][0]
__________________________________________________________________________________________
batch_normalization_202 (BatchN (None, 3, 3, 256)    768         conv2d_202[0][0]
__________________________________________________________________________________________
activation_199 (Activation)     (None, 3, 3, 192)    0           batch_normalization_
__________________________________________________________________________________________
activation_202 (Activation)     (None, 3, 3, 256)    0           batch_normalization_
__________________________________________________________________________________________
block8_10_mixed (Concatenate)   (None, 3, 3, 448)    0           activation_199[0][0]
                                                                 activation_202[0][0]
__________________________________________________________________________________________
block8_10_conv (Conv2D)         (None, 3, 3, 2080)   933920      block8_10_mixed[0][0
__________________________________________________________________________________________
block8_10 (Lambda)              (None, 3, 3, 2080)   0           block8_9_ac[0][0]
                                                                 block8_10_conv[0][0]
__________________________________________________________________________________________
conv_7b (Conv2D)                (None, 3, 3, 1536)   3194880     block8_10[0][0]
__________________________________________________________________________________________
conv_7b_bn (BatchNormalization) (None, 3, 3, 1536)   4608        conv_7b[0][0]
__________________________________________________________________________________________
conv_7b_ac (Activation)         (None, 3, 3, 1536)   0           conv_7b_bn[0][0]
__________________________________________________________________________________________
flatten (Flatten)               (None, 13824)        0           conv_7b_ac[0][0]
__________________________________________________________________________________________
dense (Dense)                   (None, 1024)         14156800    flatten[0][0]
__________________________________________________________________________________________
dense_1 (Dense)                 (None, 1024)         1049600     dense[0][0]
__________________________________________________________________________________________
dense_2 (Dense)                 (None, 6)            6150        dense_1[0][0]
==========================================================================================
Total params: 69,549,286
Trainable params: 69,488,742
Non-trainable params: 60,544

__________________________________________________________________________________________
time: 295 ms (started: 2021-01-06 02:16:08 +00:00)
```

Loading Data

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

```
time: 554 µs (started: 2021-01-06 02:16:08 +00:00)
```

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

```
time: 555 µs (started: 2021-01-06 02:16:08 +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: 19.6 ms (started: 2021-01-06 02:16:08 +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: 792 µs (started: 2021-01-06 02:16:08 +00:00)

Compile the model

```
# initial_learning_rate = 0.1
# lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
#     initial_learning_rate,
#     decay_steps=7000,
#     decay_rate=0.96,
#     staircase=True)

# opt = keras.optimizers.Adam(learning_rate=lr_schedule)
# model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

time: 2.69 ms (started: 2021-01-06 02:16:08 +00:00)

```
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: 36.1 ms (started: 2021-01-06 02:16:08 +00:00)

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

```
time: 2.75 ms (started: 2021-01-06 02:16:08 +00:00)
```

```
img_height = 150
img_width = 150
batch_size = 16
nb_epochs = 25
```

```
time: 1.59 ms (started: 2021-01-06 02:16:08 +00:00)
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2, # zoom
    rotation_range=10, # rotation
    width_shift_range=0.2, # horizontal shift
    height_shift_range=0.2, # vertical shift
    horizontal_flip=True) # horizontal flip
    # channel_shift_range = [-0.1, 0.1]
    # )
    # ,validation_split=0.3) # set validation split
```

```
time: 3.53 ms (started: 2021-01-06 02:16:08 +00:00)
```

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

```
Found 14033 images belonging to 6 classes.
time: 24.9 s (started: 2021-01-06 02:20:08 +00:00)
```

```
val_datagen = ImageDataGenerator(rescale=1. / 255)
```

```
time: 2.08 ms (started: 2021-01-06 02:20:33 +00:00)
```

```
validation_generator = val_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 3000 images belonging to 6 classes.
time: 3.99 s (started: 2021-01-06 02:20:33 +00:00)
```

```
# validation_generator[0]
```

```
time: 816 µs (started: 2021-01-06 02:20:37 +00:00)
```

```
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 1/25
877/877 [==============================] - 6053s 7s/step - loss: 0.8461 - accuracy: 0.66
Epoch 2/25
877/877 [==============================] - 149s 170ms/step - loss: 0.4194 - accuracy: 0
Epoch 3/25
877/877 [==============================] - 151s 172ms/step - loss: 0.3348 - accuracy: 0
Epoch 4/25
877/877 [==============================] - 152s 173ms/step - loss: 0.2888 - accuracy: 0
Epoch 5/25
877/877 [==============================] - 152s 173ms/step - loss: 0.2700 - accuracy: 0
Epoch 6/25
877/877 [==============================] - 152s 174ms/step - loss: 0.2338 - accuracy: 0
Epoch 7/25
877/877 [==============================] - 153s 174ms/step - loss: 0.2293 - accuracy: 0
Epoch 8/25
877/877 [==============================] - 152s 174ms/step - loss: 0.2032 - accuracy: 0
Epoch 9/25
877/877 [==============================] - 153s 174ms/step - loss: 0.1838 - accuracy: 0
Epoch 10/25
877/877 [==============================] - 151s 173ms/step - loss: 0.1754 - accuracy: 0
Epoch 11/25
877/877 [==============================] - 149s 170ms/step - loss: 0.1643 - accuracy: 0
Epoch 12/25
877/877 [==============================] - 149s 169ms/step - loss: 0.1492 - accuracy: 0
Epoch 13/25
877/877 [==============================] - 150s 171ms/step - loss: 0.1461 - accuracy: 0
Epoch 14/25
877/877 [==============================] - 149s 170ms/step - loss: 0.1412 - accuracy: 0
Epoch 15/25
877/877 [==============================] - 149s 170ms/step - loss: 0.1365 - accuracy: 0
Epoch 16/25
877/877 [==============================] - 149s 169ms/step - loss: 0.1215 - accuracy: 0
Epoch 17/25
877/877 [==============================] - 148s 169ms/step - loss: 0.1270 - accuracy: 0
Epoch 18/25
877/877 [==============================] - 148s 168ms/step - loss: 0.1186 - accuracy: 0
Epoch 19/25
877/877 [==============================] - 147s 168ms/step - loss: 0.1157 - accuracy: 0
Epoch 20/25
```

```
877/877 [==============================] - 147s 168ms/step - loss: 0.0962 - accuracy: 0
Epoch 21/25
877/877 [==============================] - 147s 168ms/step - loss: 0.1024 - accuracy: 0
Epoch 22/25
877/877 [==============================] - 146s 167ms/step - loss: 0.0811 - accuracy: 0
Epoch 23/25
877/877 [==============================] - 145s 166ms/step - loss: 0.0767 - accuracy: 0
Epoch 24/25
877/877 [==============================] - 146s 166ms/step - loss: 0.0844 - accuracy: 0
Epoch 25/25
877/877 [==============================] - 146s 166ms/step - loss: 0.0862 - accuracy: 0
time: 2h 40min 39s (started: 2021-01-06 02:20:37 +00:00)
```

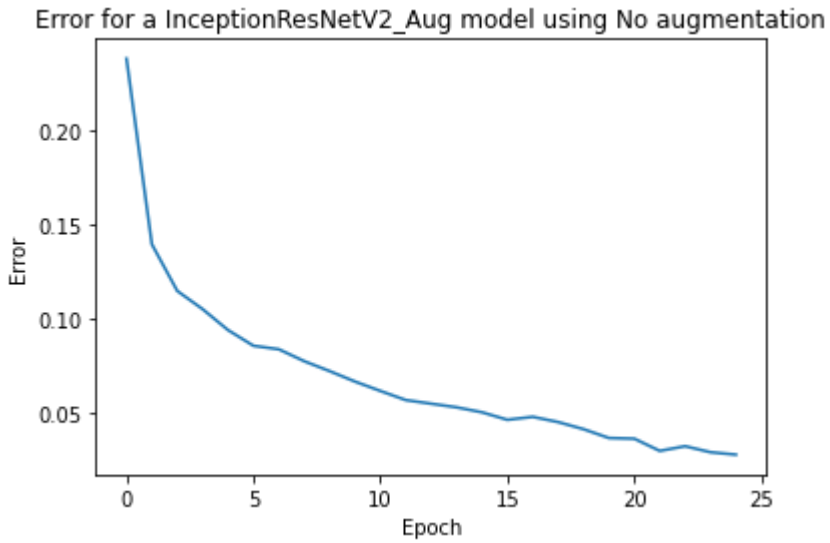
```
# save the model's trained weights
model.save_weights('/content/drive/MyDrive/CV/Assignment 3/InceptionResNetV2_Aug_transfer_tra
```

```
time: 1.84 s (started: 2021-01-06 05:01:17 +00:00)
```

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

```
time: 610 µs (started: 2021-01-06 05:01:19 +00:00)
```

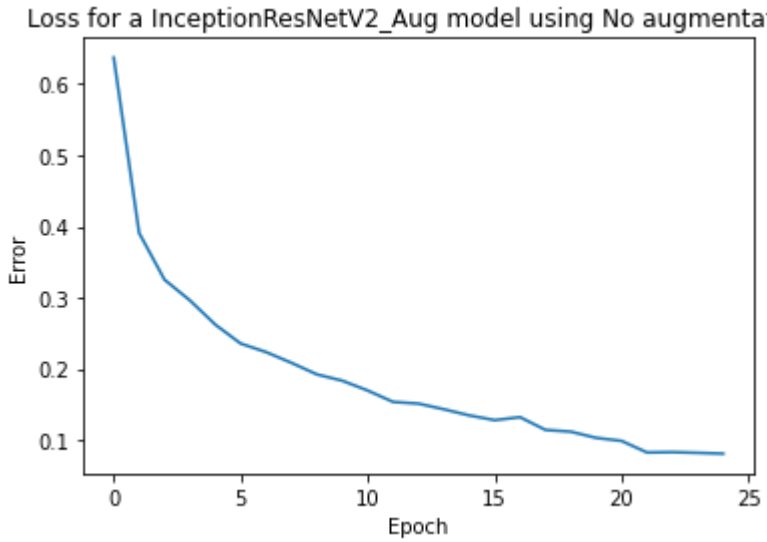
```
simple_acc = H.history['accuracy']
plt.plot([1 - acc for acc in simple_acc])
plt.title('Error for a InceptionResNetV2_Aug model using No augmentation')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/InceptionResNetV2_Aug/simple_acc_error.pn
plt.show()
```




```
time: 212 ms (started: 2021-01-06 05:01:19 +00:00)
```

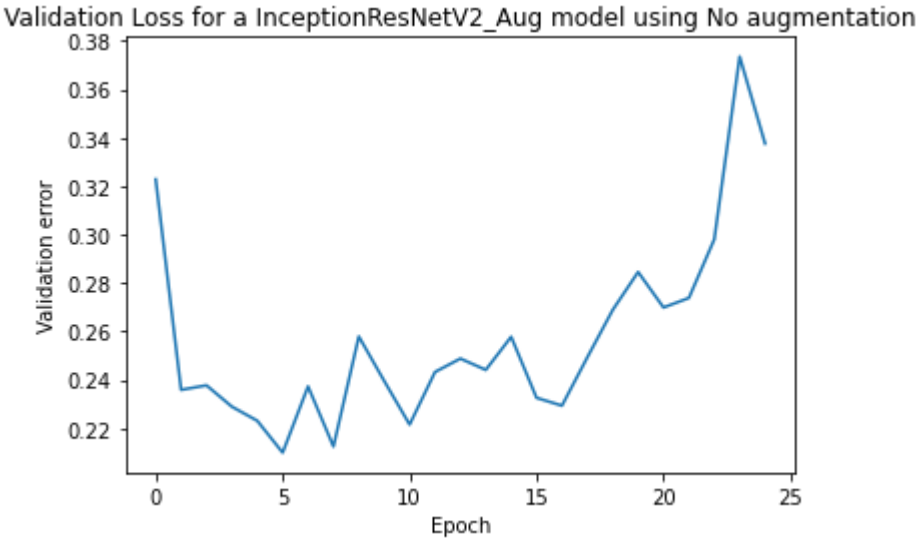
```
simple_loss = H.history['loss']
plt.plot([los for los in simple_loss])
plt.title('Loss for a InceptionResNetV2_Aug model using No augmentation')
```

```
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/InceptionResNetV2_Aug/simple_loss_error.p
plt.show()
```



time: 207 ms (started: 2021-01-06 05:01:19 +00:00)

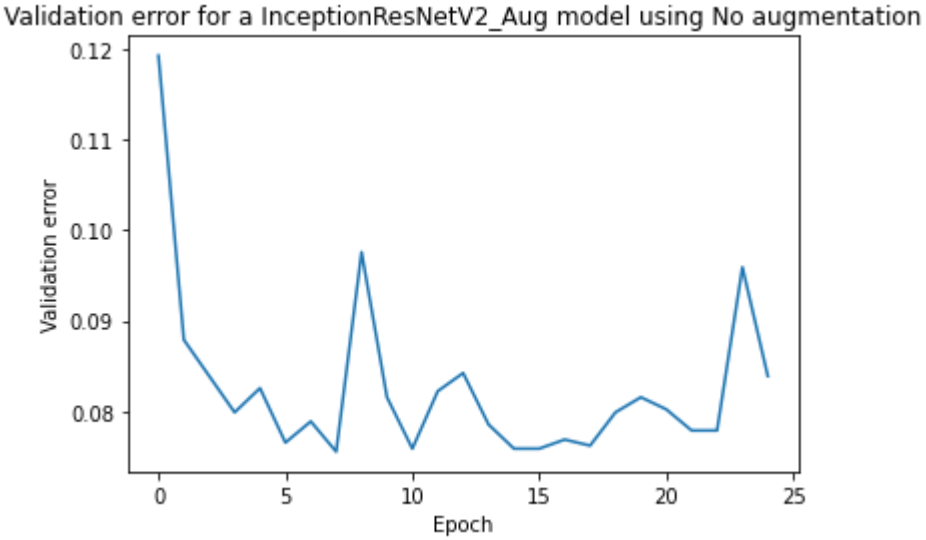
```
simple_val_loss = H.history['val_loss']
plt.plot([los for los in simple_val_loss])
plt.title('Validation Loss for a InceptionResNetV2_Aug model using No augmentation')
plt.ylabel('Validation error')
plt.xlabel('Epoch')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/InceptionResNetV2_Aug/simple_Validation_l
plt.show()
```



time: 199 ms (started: 2021-01-06 05:01:19 +00:00)

```
simple_val_acc = H.history['val_accuracy']
plt.plot([1 - acc for acc in simple_val_acc])
plt.title('Validation error for a InceptionResNetV2_Aug model using No augmentation')
plt.ylabel('Validation error')
```

```
plt.xlabel('Epoch')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/InceptionResNetV2_Aug/simple_Validation_e
plt.show()
```




```
time: 186 ms (started: 2021-01-06 05:01:19 +00:00)
```

```
# test_datagen = ImageDataGenerator()
```

```
time: 734 µs (started: 2021-01-06 05:01:19 +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
```

```
time: 1.22 ms (started: 2021-01-06 05:01:20 +00:00)
```

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

```
loading pred images
100%|██████████| 1330/1330 [08:01<00:00,  2.76it/s]
100%|██████████| 1297/1297 [07:52<00:00,  2.75it/s]
100%|██████████| 1128/1128 [06:56<00:00,  2.71it/s]
100%|██████████| 1166/1166 [07:08<00:00,  2.72it/s]
100%|██████████| 1236/1236 [07:36<00:00,  2.70it/s]
100%|██████████| 1144/1144 [06:54<00:00,  2.76it/s]
time: 44min 48s (started: 2021-01-06 05:01:20 +00:00)
```

```
lb = LabelBinarizer()
y_test = lb.fit_transform(y_test)
```

```
time: 11.3 ms (started: 2021-01-06 05:46:08 +00:00)
```

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

```
115/115 [==============================] - 20s 143ms/step - loss: 0.9228 - accuracy: 0.8
Test Loss =  0.9227569103240967
Test Accuracy =  0.821668267250061
time: 22.2 s (started: 2021-01-06 05:46:08 +00:00)
```

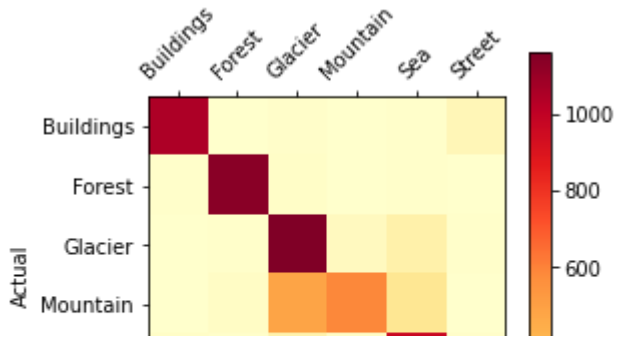
```
'''CONFUSION MATRIX'''
# Making prediction
y_pred = model.predict(X_test)
y_true = np.argmax(y_test, axis=-1)

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

```
time: 21.9 s (started: 2021-01-06 05:46:30 +00:00)
```

```
confusion_mtx
```

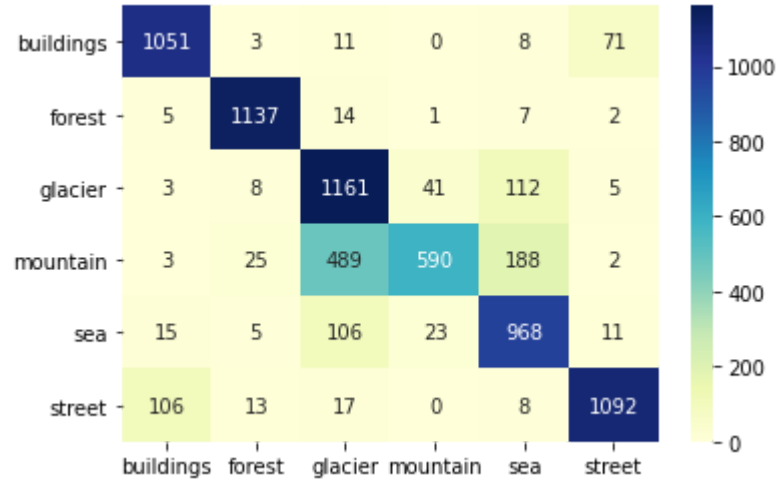
```
array([[1051,    3,   11,    0,    8,   71],
       [   5, 1137,   14,    1,    7,    2],
       [   3,    8, 1161,   41,  112,    5],
       [   3,   25,  489,  590,  188,    2],
       [  15,    5,  106,   23,  968,   11],
       [ 106,   13,   17,    0,    8, 1092]])time: 2.97 ms (started: 2021-01-06 05:46:52
```

```
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)
```




```
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")
```

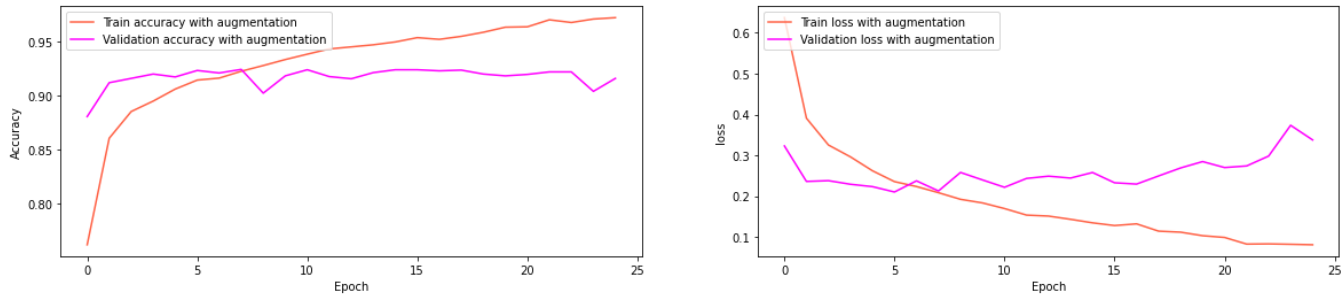
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f78e4111e48>
```




```
time: 404 ms (started: 2021-01-06 05:46:52 +00:00)
```

```
fig, axis = plt.subplots(1, 2, figsize=(20, 4))


axis[0].plot(H.history['accuracy'],
         label='Train accuracy with augmentation',
         c='tomato', ls='-')
axis[0].plot(H.history['val_accuracy'],
         label='Validation accuracy with augmentation',
         c='magenta', ls='-')

axis[0].set_xlabel('Epoch')
axis[0].set_ylabel('Accuracy')
axis[0].legend(loc='upper left')
```

```
axis[1].plot(H.history['loss'],
         label='Train loss with augmentation',
         c='tomato', ls='-')
axis[1].plot(H.history['val_loss'],
         label='Validation loss with augmentation',
         c='magenta', ls='-')

axis[1].set_xlabel('Epoch')
axis[1].set_ylabel('loss')
axis[1].legend(loc='upper left')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/InceptionResNetV2_Aug/simple_Validation_e
plt.show()
```




```
time: 478 ms (started: 2021-01-06 05:46:53 +00:00)
```