بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

```
# %reset
```

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

```
Mounted at /content/drive
```

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: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/python
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: pygments 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: ipython-genutils in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-packages
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: 122 µs (started: 2021-01-08 07:15:14 +00:00)
```

Initializing

```
img_width = 50
img_height = 50
nb_epochs = 25
batch_siz = 32
```

```
time: 1.04 ms (started: 2021-01-08 07:21:31 +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: 1.3 s (started: 2021-01-08 08:44:02 +00:00)
```

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16
58892288/58889256 [==============================] - 0s 0us/step
time: 6.23 s (started: 2021-01-08 08:44:03 +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: 20.7 ms (started: 2021-01-08 08:44:09 +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, 50, 50, 3)]       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 50, 50, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 50, 50, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 25, 25, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 25, 25, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 25, 25, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 12, 12, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 12, 12, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 6, 6, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 6, 6, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 3, 3, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 1024)              525312
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 6150
=================================================================
Total params: 15,246,150
Trainable params: 15,246,150
Non-trainable params: 0
_________________________________________________________________
time: 14.8 ms (started: 2021-01-08 08:44:09 +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: 21.5 ms (started: 2021-01-08 08:44:09 +00:00)
```

Loading Data

```
# Cutout Function
def apply_mask(image, size=12, n_squares=1):
  h, w, channels = image.shape
  new_image = image
  y = np.random.randint(h)
  x = np.random.randint(w)
  y1 = np.clip(y - size // 2, 0, h)
  y2 = np.clip(y + size // 2, 0, h)
  x1 = np.clip(x - size // 2, 0, w)
  x2 = np.clip(x + size // 2, 0, w)
  new_image[y1:y2,x1:x2,:] = 0
  return new_image
```

```
time: 10.4 ms (started: 2021-01-08 08:44:13 +00:00)
```

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

  for direc in class_dirs:
    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
      image = apply_mask(image)
      data.append(image)
      labels.append(direc)
  # normalizing and converting to numpy array format
  data = np.array(data, dtype='float')/255.0
  labels = np.array(labels)
  return data, labels
```

```
time: 9.89 ms (started: 2021-01-08 08:44:17 +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:
```

```
    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)
  # normalizing and converting to numpy array format
  data = np.array(data, dtype='float')/255.0
  labels = np.array(labels)
  return data, labels
```

```
time: 15.6 ms (started: 2021-01-08 07:20:15 +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: 1.02 ms (started: 2021-01-08 07:15:54 +00:00)
```

```
print('loading train images')
X_train, y_train = load_cutout_data(train_dir)
```

```
loading train images
100%|██████████| 2190/2190 [09:55<00:00,  3.68it/s]
100%|██████████| 2271/2271 [10:05<00:00,  3.75it/s]
100%|██████████| 2404/2404 [10:51<00:00,  3.69it/s]
100%|██████████| 2512/2512 [11:01<00:00,  3.80it/s]
100%|██████████| 2274/2274 [10:00<00:00,  3.79it/s]
100%|██████████| 2382/2382 [11:01<00:00,  3.60it/s]
time: 1h 3min 11s (started: 2021-01-08 08:44:26 +00:00)
```

```
X_valid, y_valid = load_data(test_dir)
```

```
100%|██████████| 437/437 [01:53<00:00,  3.86it/s]
100%|██████████| 474/474 [02:01<00:00,  3.90it/s]
100%|██████████| 553/553 [02:21<00:00,  3.92it/s]
100%|██████████| 525/525 [02:15<00:00,  3.86it/s]
100%|██████████| 510/510 [02:12<00:00,  3.85it/s]
100%|██████████| 501/501 [02:17<00:00,  3.65it/s]time: 13min 5s (started: 2021-01-08 09
```

```
# np.random.shuffle()
# np.random.shuffle()
```

```
X_train = np.append(X_train, X_valid, axis=0)
y_train = np.append(y_train, y_valid, axis=0)
```

```
time: 471 ms (started: 2021-01-08 10:11:00 +00:00)
```

```
lb = LabelBinarizer()
```

```
time: 962 µs (started: 2021-01-08 10:00:46 +00:00)
```

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

```
time: 23.5 ms (started: 2021-01-08 10:11:21 +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=0.3, rand
```

```
time: 290 ms (started: 2021-01-08 10:11:42 +00:00)
```

```
# (X_valid, X_valid0, y_valid, y_valid0) = train_test_split(X_train, y_train, test_size=0.0,
```

Train the model

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
```

```
time: 821 µs (started: 2021-01-08 10:11:50 +00:00)
```

```
# patient early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)
mc = ModelCheckpoint('best_model.h5', monitor='val_accuracy', mode='max', verbose=1, save_bes
```

```
time: 1.24 ms (started: 2021-01-08 10:18:29 +00:00)
```

```
H = model.fit(X_train, y_train, batch_size=batch_siz,
              epochs=nb_epochs,
              validation_data=(X_valid, y_valid),
              verbose=0, callbacks=[es, mc])
```

```
Epoch 00001: val_accuracy improved from -inf to 0.89451, saving model to best_model.h5

Epoch 00002: val_accuracy improved from 0.89451 to 0.89534, saving model to best_model.h

Epoch 00003: val_accuracy did not improve from 0.89534

Epoch 00004: val_accuracy improved from 0.89534 to 0.89784, saving model to best_model.h

Epoch 00005: val_accuracy did not improve from 0.89784

Epoch 00006: val_accuracy improved from 0.89784 to 0.90532, saving model to best_model.h

Epoch 00007: val_accuracy did not improve from 0.90532
```

```
Epoch 00008: val_accuracy improved from 0.90532 to 0.91032, saving model to best_model.h
Epoch 00008: early stopping
time: 2min 41s (started: 2021-01-08 10:18:33 +00:00)
```

```
save_path = '/content/drive/MyDrive/CV/Assignment 3/vgg16_sgd_cutout'
```

```
time: 784 µs (started: 2021-01-08 10:14:44 +00:00)
```

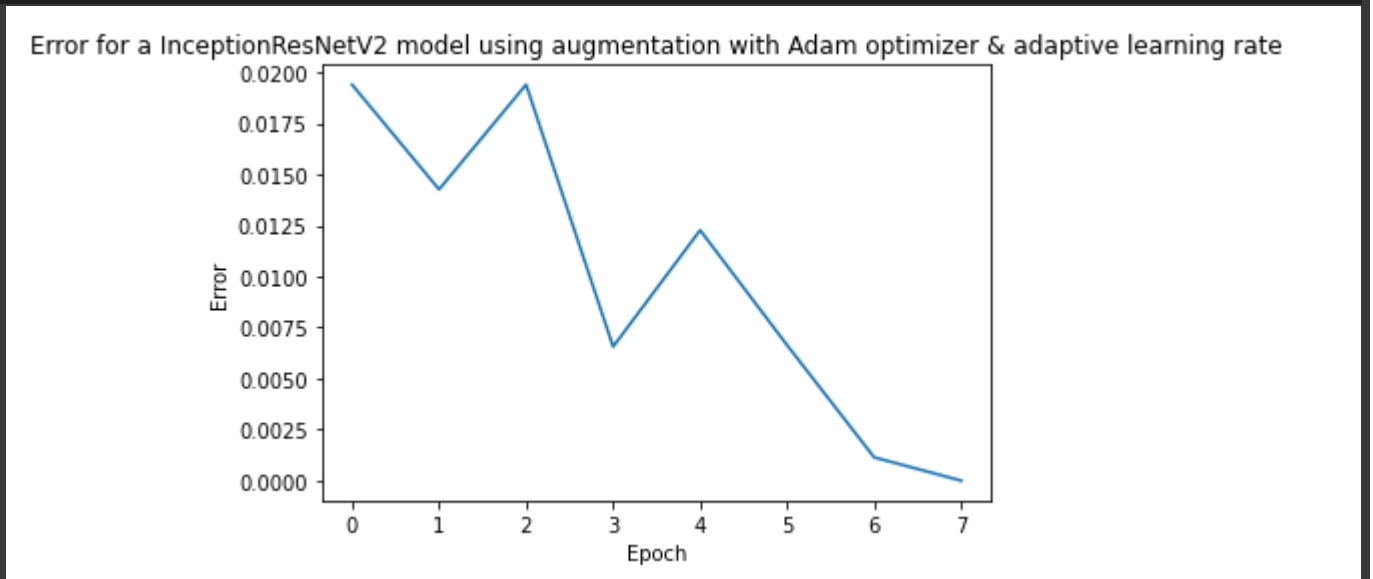
```
# save the model's trained weights
model.save_weights(save_path+"transfer_trained_wts.h5")
```

```
time: 824 ms (started: 2021-01-08 10:22:33 +00:00)
```

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

```
time: 852 µs (started: 2021-01-08 10:14:45 +00:00)
```

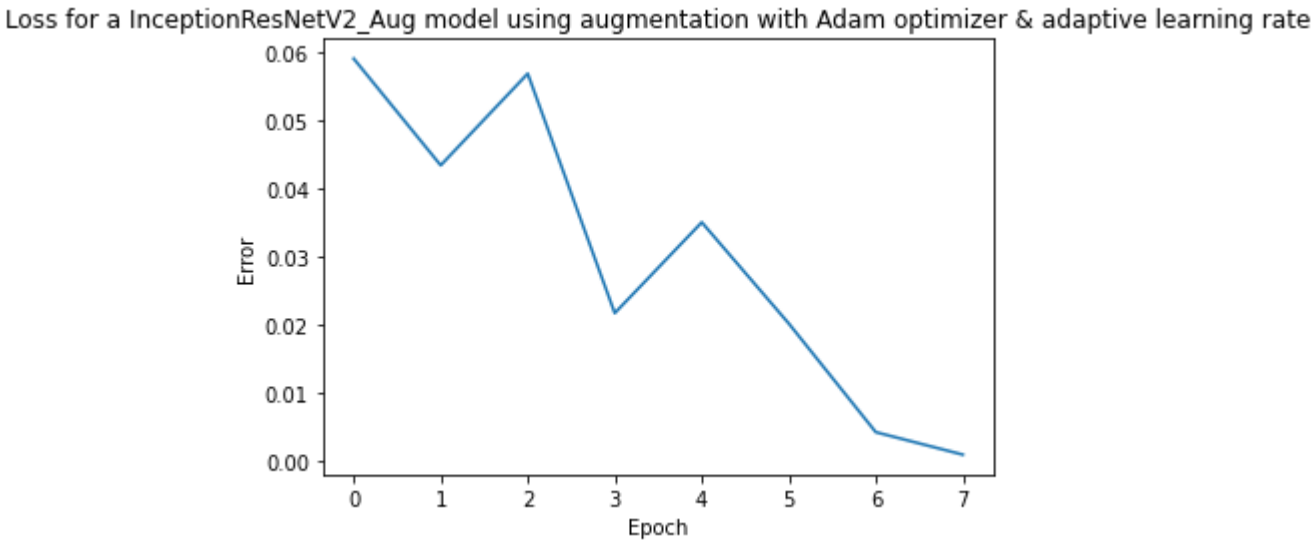
```
simple_acc = H.history['accuracy']
plt.plot([1 - acc for acc in simple_acc])
plt.title('Error for a InceptionResNetV2 model using augmentation with Adam optimizer & adapt
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.savefig(save_path+'/simple_acc_error.png')
plt.show()
```




```
time: 236 ms (started: 2021-01-08 10:22:36 +00:00)
```

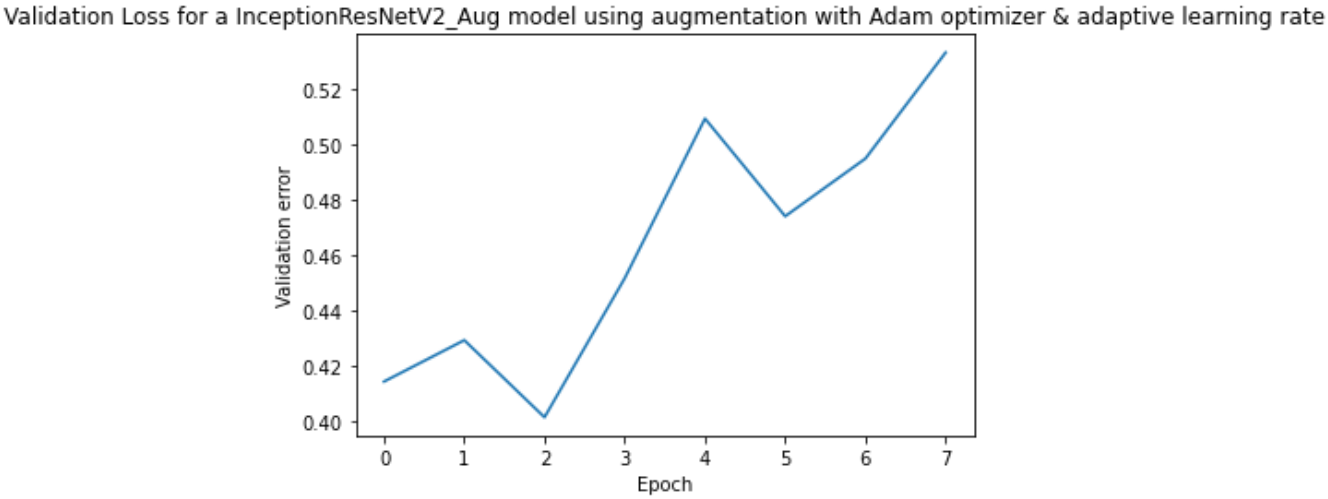
```
simple_loss = H.history['loss']
plt.plot([los for los in simple_loss])
plt.title('Loss for a InceptionResNetV2_Aug model using augmentation with Adam optimizer & ad
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.savefig(save_path+'/simple_loss_error.png')
```

```
plt.show()
```



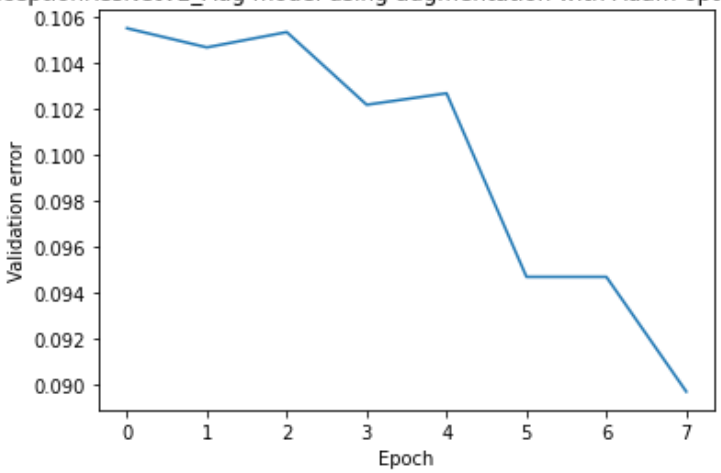

```
time: 211 ms (started: 2021-01-08 10:22:52 +00:00)
```

```
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 augmentation with Adam opt
plt.ylabel('Validation error')
plt.xlabel('Epoch')
plt.savefig(save_path+'/simple_Validation_loss_error.png')
plt.show()
```




```
time: 225 ms (started: 2021-01-08 10:22:56 +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 augmentation with Adam op
plt.ylabel('Validation error')
plt.xlabel('Epoch')
plt.savefig(save_path+'/simple_Validation_error.png')
plt.show()
```



time: 227 ms (started: 2021-01-08 10:22:59 +00:00)

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

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

time: 12.6 ms (started: 2021-01-08 10:16:47 +00:00)

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

```
229/229 [==============================] - 3s 13ms/step - loss: 0.8339 - accuracy: 0.885
Test Loss =  0.8338767290115356
Test Accuracy =  0.885495126247406
time: 3 s (started: 2021-01-08 10:21:15 +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))
```

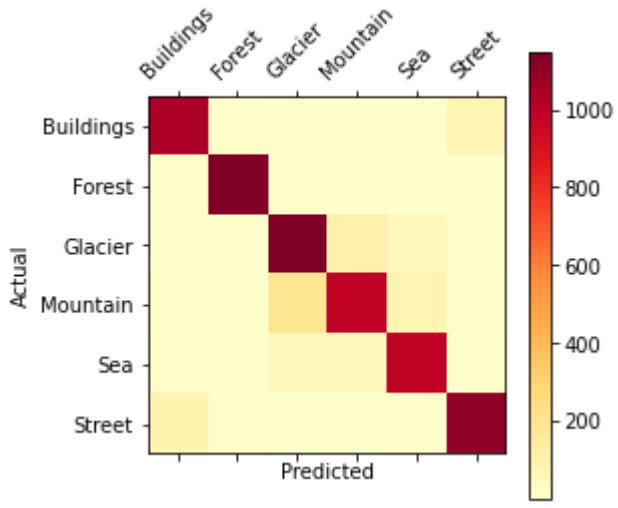
time: 3.05 s (started: 2021-01-08 10:17:02 +00:00)

```
confusion_mtx
```

```
array([[1044,    7,    2,    2,    4,   85],
       [   7, 1148,    1,    4,    2,    4],
```

```
[  12,    5, 1149,  102,   57,    5],
[  18,    5,  191,  990,   87,    6],
[  27,    2,   55,   48,  992,    4],
[  92,   18,    4,    5,   11, 1106]])time: 2.82 ms (started: 2021-01-08 10:17:05
```

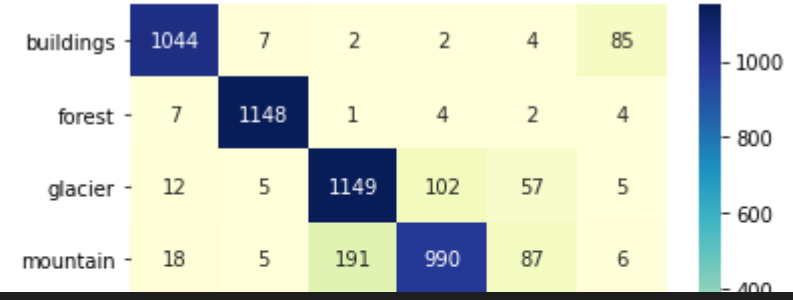
```
def plot_confusion_matrix(df_confusion, title='Confusion matrix', cmap=plt.cm.YlOrRd):
  plt.matshow(df_confusion, cmap=cmap) # imshow
  plt.colorbar()
  tick_marks = np.arange(6)
  names = ["Buildings", "Forest", "Glacier", "Mountain", "Sea", "Street"]
  plt.xticks(tick_marks, names, rotation=45)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




```
time: 182 ms (started: 2021-01-08 10:17:08 +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 0x7fb0720802e8>
```

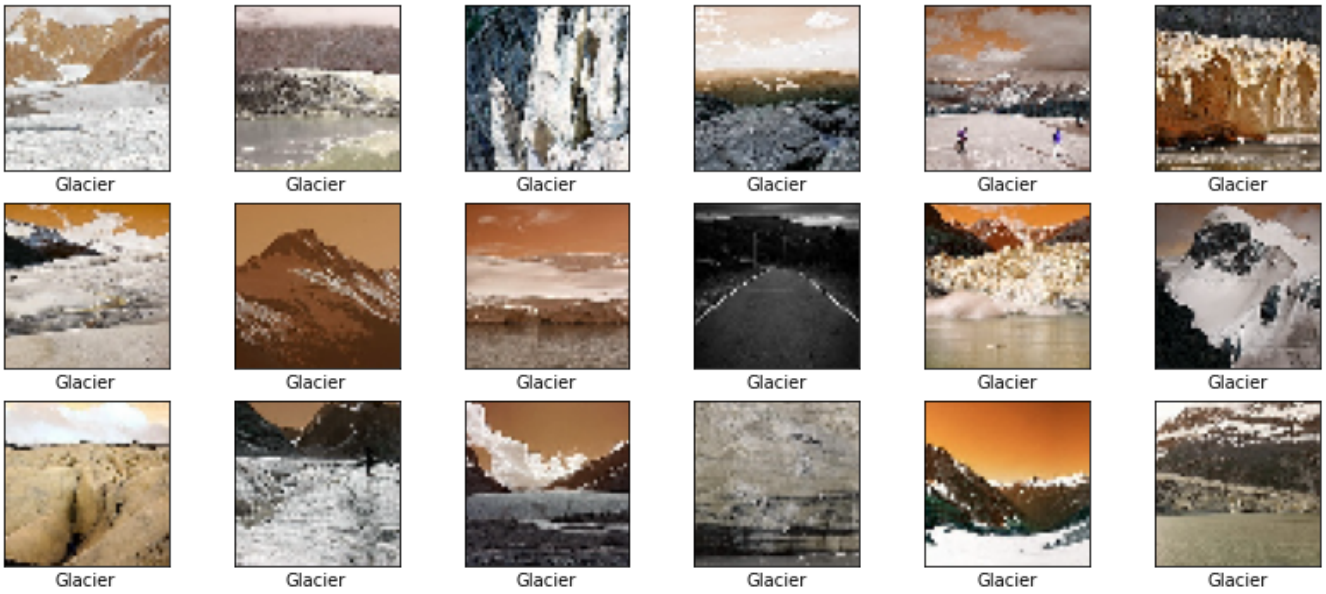



```
fig, axis = plt.subplots(1, 2, figsize=(20, 4))


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(save_path+'/simple_Validation_error&loss.png')
plt.show()
```


```
def visualize_data(images, categories, class_names):
    fig = plt.figure(figsize=(14, 6))
    fig.patch.set_facecolor('white')
    already = []
    for i in range(3 * 6):
        plt.subplot(3, 6, i+1)
        plt.xticks([])
        plt.yticks([])
        plt.imshow(images[i])
        already.append(images[i])
        class_index = categories[i].argmax()
        plt.xlabel(class_names[class_index])
    plt.show()
```

time: 6.2 ms (started: 2021-01-08 10:17:25 +00:00)

```
names = ["Buildings", "Forest", "Glacier", "Mountain", "Sea", "Street"]
visualize_data(X_test, y_test, names)
```



time: 610 ms (started: 2021-01-08 10:17:28 +00:00)