بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

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

```
Bismillah
```

```
# %reset
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
time: 2.23 ms (started: 2021-01-05 17:58:57 +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: setuptools>=18.5 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.6/
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/local/lib/pythor
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-packages
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: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (frc
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages (from p
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
```

```
# 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
```

```
%matplotlib inline

from google.colab.patches import cv2_imshow

%load_ext autotime
```

```
     time: 141 µs (started: 2021-01-05 17:58:54 +00:00)
```

Initializing

```
img_width = 90
img_height = 90
```

```
     time: 831 µs (started: 2021-01-05 17:58:54 +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: 1.44 s (started: 2021-01-05 17:58:54 +00:00)
```

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

```
     time: 1.14 s (started: 2021-01-05 17:58:56 +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
output = Dense(6, activation='softmax')(class1) # add softmax layer
```

```
     time: 21.8 ms (started: 2021-01-05 17:58:57 +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, 90, 90, 3)]       0
```

```
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 90, 90, 64)        1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 90, 90, 64)        36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 45, 45, 64)        0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 45, 45, 128)       73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 45, 45, 128)       147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 22, 22, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 22, 22, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 22, 22, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 22, 22, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 11, 11, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 11, 11, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 11, 11, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 11, 11, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 5, 5, 512)         0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 5, 5, 512)         2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 5, 5, 512)         2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 5, 5, 512)         2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0         
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176   
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 6150      
=================================================================
Total params: 16,819,014
Trainable params: 16,819,014
Non-trainable params: 0
_________________________________________________________________
time: 15 ms (started: 2021-01-05 17:58:57 +00:00)
```

Loading Data

Load data function

```
# 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: 14.8 ms (started: 2021-01-05 17:58:57 +00:00)
```

Data Paths

```
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: 1.58 ms (started: 2021-01-05 17:58:57 +00:00)
```

Loading Training Images

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

```
  0%|          | 0/2191 [00:00<?, ?it/s]loading train images
100%|██████████| 2191/2191 [00:07<00:00, 311.44it/s]
100%|██████████| 2271/2271 [00:07<00:00, 304.99it/s]
100%|██████████| 2404/2404 [00:07<00:00, 338.79it/s]
100%|██████████| 2512/2512 [00:07<00:00, 344.29it/s]
100%|██████████| 2274/2274 [00:06<00:00, 354.69it/s]
100%|██████████| 2382/2382 [00:06<00:00, 346.15it/s]
time: 43.4 s (started: 2021-01-05 17:58:57 +00:00)
```

Loading Validation images

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

```
100%|██████████| 437/437 [00:01<00:00, 352.50it/s]
100%|██████████| 474/474 [00:01<00:00, 318.78it/s]
```

```
100%|██████████| 553/553 [00:01<00:00, 343.63it/s]
100%|██████████| 525/525 [00:01<00:00, 354.31it/s]
100%|██████████| 510/510 [00:01<00:00, 358.15it/s]
100%|██████████| 501/501 [00:01<00:00, 350.86it/s]
time: 8.95 s (started: 2021-01-05 17:59:40 +00:00)
```

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

```
time: 1.24 s (started: 2021-01-05 17:59:49 +00:00)
```

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

```
time: 16.4 ms (started: 2021-01-05 17:59:51 +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, rand
```

```
time: 784 ms (started: 2021-01-05 17:59:51 +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: 22.3 ms (started: 2021-01-05 17:59:51 +00:00)
```

Train the model

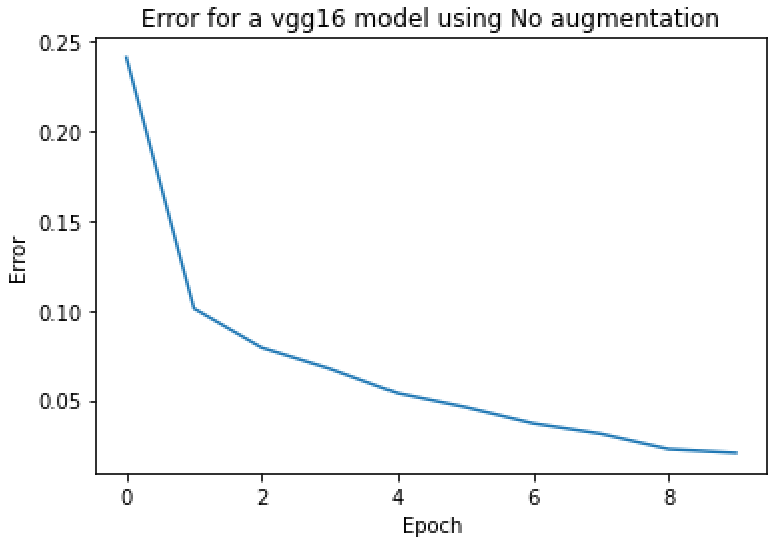
```
H = model.fit(X_train, y_train, batch_size=128,
              epochs=10,
              validation_data=(X_valid, y_valid))
```

```
Epoch 1/10
107/107 [==============================] - 47s 345ms/step - loss: 0.9978 - accuracy: 0.6
Epoch 2/10
107/107 [==============================] - 35s 327ms/step - loss: 0.2988 - accuracy: 0.8
Epoch 3/10
107/107 [==============================] - 34s 322ms/step - loss: 0.2360 - accuracy: 0.9
Epoch 4/10
107/107 [==============================] - 35s 326ms/step - loss: 0.1882 - accuracy: 0.9
Epoch 5/10
107/107 [==============================] - 35s 325ms/step - loss: 0.1628 - accuracy: 0.9
```

```
Epoch 6/10
107/107 [==============================] - 35s 325ms/step - loss: 0.1260 - accuracy: 0.9
Epoch 7/10
107/107 [==============================] - 35s 325ms/step - loss: 0.1011 - accuracy: 0.9
Epoch 8/10
107/107 [==============================] - 35s 325ms/step - loss: 0.0905 - accuracy: 0.9
Epoch 9/10
107/107 [==============================] - 35s 326ms/step - loss: 0.0714 - accuracy: 0.9
Epoch 10/10
107/107 [==============================] - 35s 325ms/step - loss: 0.0539 - accuracy: 0.9
time: 5min 59s (started: 2021-01-05 17:59:52 +00:00)
```

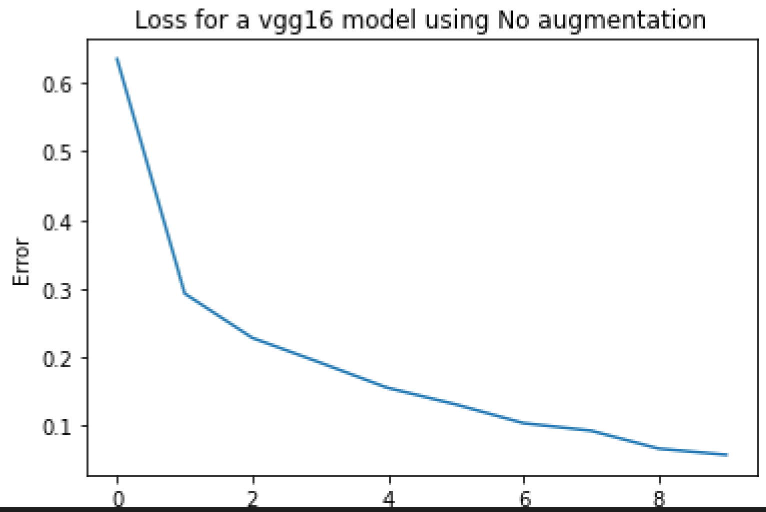
Some Graphs

```
simple_acc = H.history['accuracy']
plt.plot([1 - acc for acc in simple_acc])
plt.title('Error for a vgg16 model using No augmentation')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/vgg16/simple_acc_error.png')
plt.show()
```

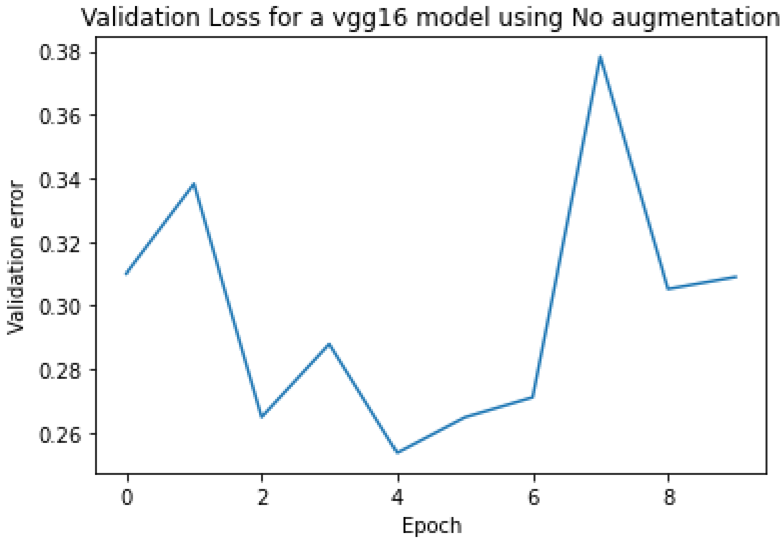


```
time: 176 ms (started: 2021-01-05 18:53:11 +00:00)
```

```
simple_loss = H.history['loss']
plt.plot([los for los in simple_loss])
plt.title('Loss for a vgg16 model using No augmentation')
plt.ylabel('Error')
plt.xlabel('Epoch')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/vgg16/simple_loss_error.png')
plt.show()
```

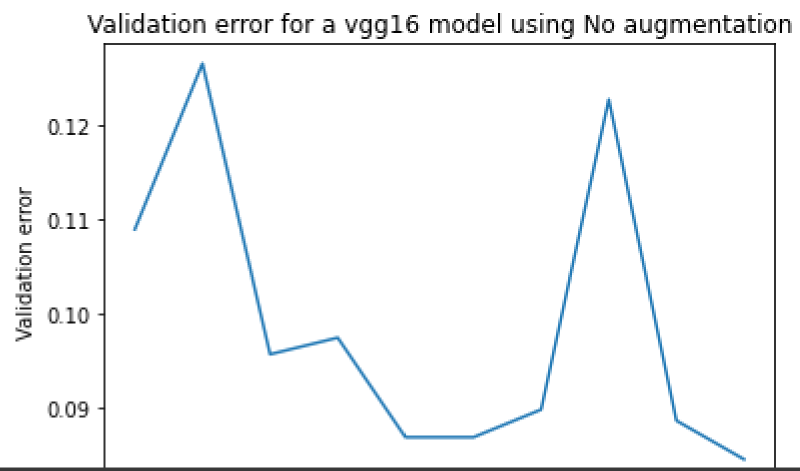


```
simple_val_loss = H.history['val_loss']
plt.plot([los for los in simple_val_loss])
plt.title('Validation Loss for a vgg16 model using No augmentation')
plt.ylabel('Validation error')
plt.xlabel('Epoch')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/vgg16/simple_Validation_loss_error.png')
plt.show()
```



```
time: 181 ms (started: 2021-01-05 18:55:28 +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 vgg16 model using No augmentation')
plt.ylabel('Validation error')
plt.xlabel('Epoch')
plt.savefig('/content/drive/MyDrive/CV/Assignment 3/vgg16/simple_Validation_error.png')
plt.show()
```



Saving Model weights

```
time: 313 ms (started: 2021-01-05 18:55:31 +00:00)
```

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

```
time: 1.75 s (started: 2021-01-05 18:05:51 +00:00)
```

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

Testing

Loading Testing Data

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

```
loading test images
100%|██████████| 1236/1236 [03:40<00:00,  5.61it/s]
100%|██████████| 1128/1128 [03:18<00:00,  5.68it/s]
100%|██████████| 1297/1297 [03:39<00:00,  5.90it/s]
100%|██████████| 1330/1330 [03:46<00:00,  5.87it/s]
100%|██████████| 1166/1166 [03:20<00:00,  5.83it/s]
100%|██████████| 1144/1144 [03:16<00:00,  5.83it/s]
time: 21min 11s (started: 2021-01-05 18:05:53 +00:00)
```

Testing Model

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

```
115/115 [==============================] - 6s 50ms/step - loss: 0.4411 - accuracy: 0.89
Test Loss =  0.44113513827323914
Test Accuracy =  0.8913847208023071
time: 7.51 s (started: 2021-01-05 18:27:05 +00:00)
```

Confusion Matrix

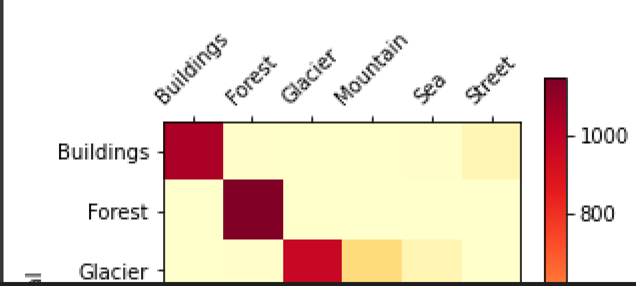
```
'''CONFUSION MATRIX'''
# Making prediction
y_pred = model.predict(X_test)
y_true = np.argmax(y_test, axis=-1)

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

```
time: 7.74 s (started: 2021-01-05 18:27:12 +00:00)
```

```
confusion_mtx
```

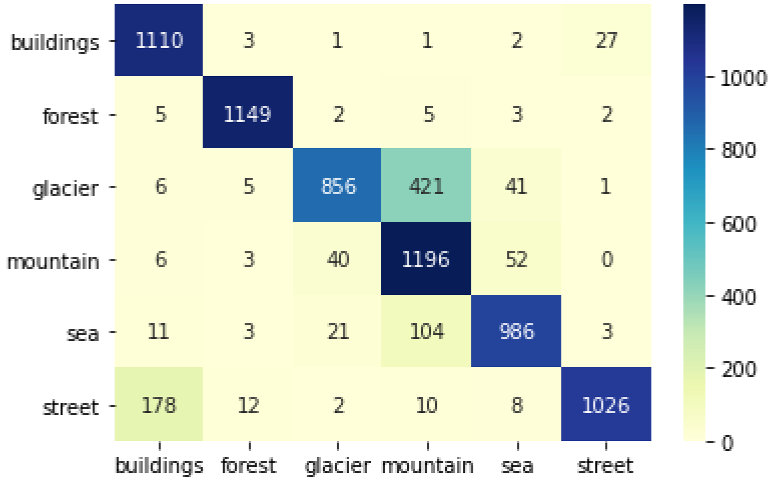
```
array([[1048,    6,    1,    1,   13,   75],
       [   4, 1152,    1,    3,    2,    4],
       [   4,    5,  966,  267,   87,    1],
       [   6,    7,   40, 1147,   94,    3],
       [   5,    5,   14,   40, 1057,    7],
       [  75,   11,    0,    3,    9, 1138]])time: 4.19 ms (started: 2021-01-05 18:27:2
```

```
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 0x7fd80e183c18>
```



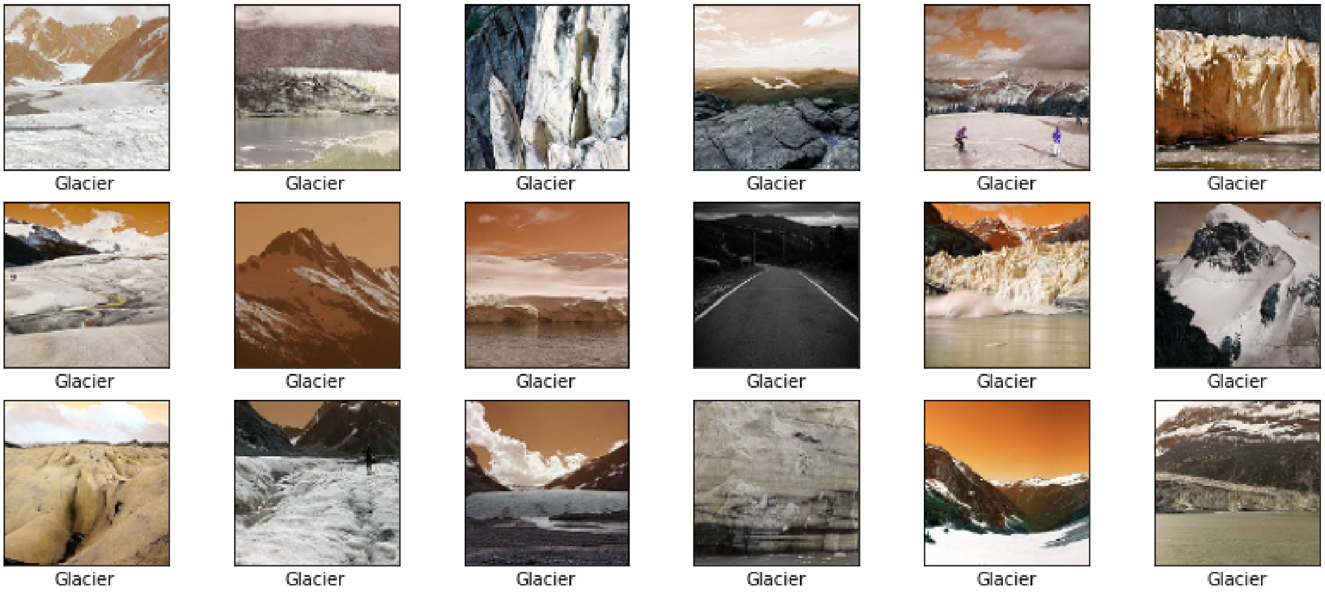
```
time: 349 ms (started: 2021-01-08 13:51:18 +00:00)
```

Visualizing some data

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

```
time: 3.75 ms (started: 2021-01-08 16:39:01 +00:00)
```

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

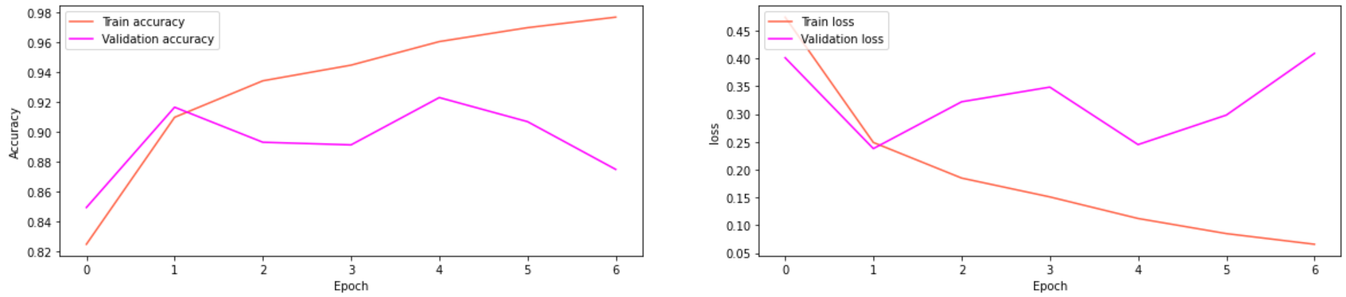


```
time: 879 ms (started: 2021-01-08 16:39:02 +00:00)
```

Loss/Error Graphs

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


axis[0].plot(H.history['accuracy'],
         label='Train accuracy',
         c='tomato', ls='-')
axis[0].plot(H.history['val_accuracy'],
         label='Validation accuracy',
         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',
         c='tomato', ls='-')
axis[1].plot(H.history['val_loss'],
         label='Validation loss',
         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()
```



```
time: 473 ms (started: 2021-01-08 13:18:12 +00:00)
```

Google Colab Link:

https://colab.research.google.com/drive/1VxFnPZbg0gKJU8KXDUHK5Cy3Gg4asQrv?usp=sharing