In [87]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.layers import Convolution2D, MaxPooling2D, Flatten, Dense, Drop
out,GlobalAveragePooling2D
from keras.applications import VGG16
```

In [88]:

```
from keras.models import Sequential
```

In [89]:

```
from numpy import append
```

import glob,os,cv2 train_images = [] for directory_path in glob.glob('../input/intel-image-classification/seg_train/seg_train/'): train_labels=[] label = directory_path.split("\")[-1] for img_path in glob.glob(os.path.join(directory_path,"*.jpg")): img = cv2.imread(img_path) img = cv2.resize(img, (256,256)) train_images.append(img) train_labels.append(label) train_images = np.array(train_images) train_labels = np.array(train_labels) test_images = [] for directory_path in glob.glob('../input/intel-image-classification/seg_test/seg_test/'):

```
    train_labels = []
    label = directory_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(directory_path,"*.jpg")):
        img = cv2.imread(img_path)
        img = cv2.resize(img,(256,256))
        train_images.append(img)
        train_labels.append(label)
```

test_images = np.array(test_images) test_labels = np.array(test_labels)

In [90]:

```
train_images = '../input/intel-image-classification/seg_train/seg_train/'
test_images = '../input/intel-image-classification/seg_test/seg_test/'
```

In [91]:

```
import cv2
y = cv2.imread('../input/intel-image-classification/seg_train/seg_train/bu
ildings/0.jpg')
print(y.shape)
```

```
(150, 150, 3)
```

In [92]:

```
conv_base  = VGG16(weights = 'imagenet',
                   include_top = False, input_shape=(256,256,3))
```

In [93]:

```
import os
import shutil
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale=1./255)
batch_size = 16

def extract_features(directory,sample_count):
    features = np.zeros(shape=(sample_count,8,8,512))
    labels = np.zeros(shape=(sample_count,6))
    generator = datagen.flow_from_directory(directory, target_size=(256,25
6),
                                            batch_size = batch_size, class_
mode = 'categorical')
    i = 0
    for inputs_batch, labels_batch in generator:
        features_batch = conv_base.predict(inputs_batch)
        features[i*batch_size:(i+1) * batch_size]=features_batch
        labels[i*batch_size:(i+1)*batch_size] = labels_batch
        i += 1
        if i*batch_size >= sample_count:
            break
    return features,labels
```

In [94]:

```
train_features , train_no = extract_features(train_images,2000)
test_features , test_no = extract_features(test_images,2000)
```

```
Found 14034 images belonging to 6 classes.
Found 3000 images belonging to 6 classes.
```

In [95]:

```
epochs = 10
model = Sequential()
model.add(GlobalAveragePooling2D(input_shape=(8,8,512)))
model.add(Dense(6, activation='sigmoid'))
```

In [96]:

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
____
Layer (type)                 Output Shape              Param
#
=================================================================
====
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
____
dense_21 (Dense)             (None, 6)                 3078
=================================================================
====
Total params: 3,078
```

```
Trainable params: 3,078
Non-trainable params: 0
_________________________________________________________________
____
```

In [97]:

```
from keras.optimizers import Adam
model.compile(optimizer=Adam(),
              loss ='categorical_crossentropy',
              metrics = ['accuracy'])
```

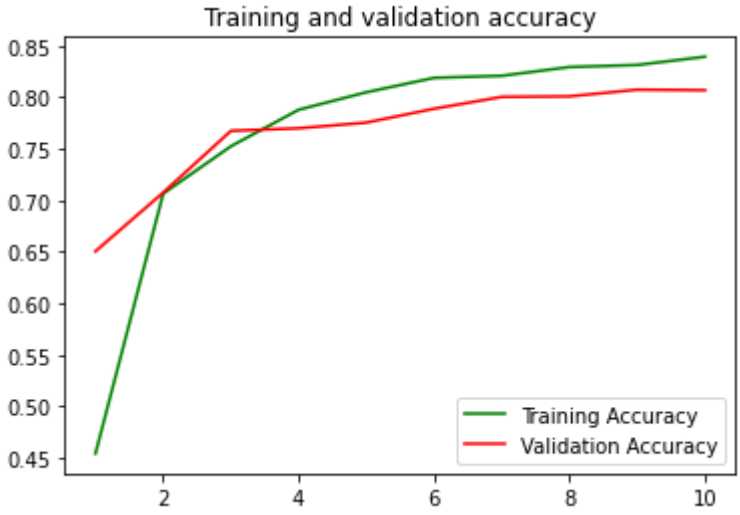
In [98]:

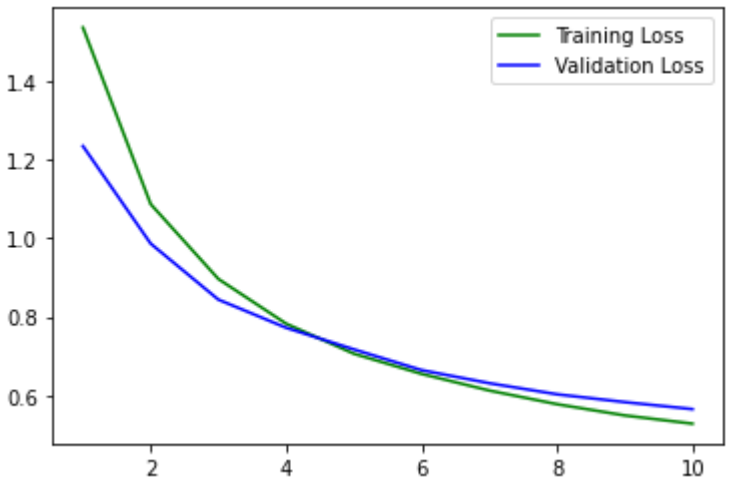
```
history = model.fit(train_features, train_no ,epochs=epochs,
                    batch_size = batch_size , validation_data=(test_featur
es,test_no))
```

```
Epoch 1/10
125/125 [==============================] - 1s 9ms/step - los
s: 1.7945 - accuracy: 0.3034 - val_loss: 1.2345 - val_accurac
y: 0.6505
Epoch 2/10
125/125 [==============================] - 1s 6ms/step - los
s: 1.1432 - accuracy: 0.6880 - val_loss: 0.9861 - val_accurac
y: 0.7075
Epoch 3/10
125/125 [==============================] - 1s 6ms/step - los
s: 0.9309 - accuracy: 0.7464 - val_loss: 0.8435 - val_accurac
y: 0.7675
Epoch 4/10
125/125 [==============================] - 1s 5ms/step - los
s: 0.7936 - accuracy: 0.7907 - val_loss: 0.7718 - val_accurac
y: 0.7700
Epoch 5/10
125/125 [==============================] - 1s 6ms/step - los
s: 0.7306 - accuracy: 0.8025 - val_loss: 0.7167 - val_accurac
y: 0.7755
Epoch 6/10
125/125 [==============================] - 1s 6ms/step - los
s: 0.6632 - accuracy: 0.8201 - val_loss: 0.6639 - val_accurac
y: 0.7890
Epoch 7/10
125/125 [==============================] - 1s 6ms/step - los
s: 0.6246 - accuracy: 0.8118 - val_loss: 0.6306 - val_accurac
y: 0.8005
Epoch 8/10
125/125 [==============================] - 1s 6ms/step - los
s: 0.5793 - accuracy: 0.8403 - val_loss: 0.6024 - val_accurac
y: 0.8010
Epoch 9/10
125/125 [==============================] - 1s 5ms/step - los
s: 0.5535 - accuracy: 0.8296 - val_loss: 0.5824 - val_accurac
y: 0.8075
Epoch 10/10
125/125 [==============================] - 1s 6ms/step - los
s: 0.5450 - accuracy: 0.8291 - val_loss: 0.5646 - val_accurac
y: 0.8070
```

In [99]:

```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1,len(acc)+1)
plt.plot(epochs , acc ,'g' , label = 'Training Accuracy')
plt.plot(epochs , val_acc,'r' ,label = 'Validation Accuracy')
plt.legend()
plt.title('Training and validation accuracy')

plt.figure()

plt.plot(epochs , loss , 'g' , label ='Training Loss')
plt.plot(epochs , val_loss , 'b', label = 'Validation Loss')
plt.legend()

plt.show()
```





In [100]:

```
from keras.preprocessing import image
def prediction(img_path):
    orig_img = image.load_img(img_path)
```

```
    img = image.load_img(img_path, target_size=(256,256))
    img_tensor = image.img_to_array(img)
    img_tensor /= 255.
    plt.imshow(orig_img)
    plt.axis('off')
    plt.show()

    features = conv_base.predict(img_tensor.reshape(1,256,256,3))
    try :
        prediction = model.predict(features)
    except:
        prediction = model.predict(features.reshape(1,8*8*512))

    classes = ['buildings','forest','glacier','mountains','sea','street']
    print('I see ..... '+str(classes[np.argmax(np.array(prediction[0]))]))
```

In [101]:

```
pred_dir = '../input/intel-image-classification/seg_pred/seg_pred/'
import random
import os
pred_files = random.sample(os.listdir(pred_dir),10)
for f in pred_files:

    prediction(pred_dir+f)
```

I see ..... sea

I see ..... glacier

I see ..... glacier

I see ..... sea

I see ..... buildings

I see ..... glacier

I see ..... mountains

I see ..... sea

I see ..... forest

I see ..... forest