Intel natural scenes dataset classification using Transferlearning Resnet50

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
```

```
import os
import cv2
import numpy as np
from imutils import paths
import matplotlib.pyplot as plt
from keras.applications.resnet50 import ResNet50
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten
```

Setting up directory to load dataset

```
train_dir = "/content/drive/MyDrive/intel-image-classification.zip (Unzipped Files)/seg_train/seg_train/"
test_dir = "/content/drive/MyDrive/intel-image-classification.zip (Unzipped Files)/Test_data.zip (Unzipped Files)/seg_pred/s
valid_dir = "/content/drive/MyDrive/intel-image-classification.zip (Unzipped Files)/seg_test/seg_test/"
```

```
# Create dataset batches for model with image augmentation
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rotation_range=90, width_shift_range=0.2, height_shift_range=0.3, shear_range=0.1, zoom_range=0.2, horizontal_flip=True,

val_datagen = ImageDataGenerator()

train_generator = train_datagen.flow_from_directory(
```

```
        train_dir,
        target_size=(150, 150),
        batch_size=64,
        seed=1)

validation_generator = val_datagen.flow_from_directory(
        valid_dir,
        shuffle=False,
        target_size=(150, 150),
        batch_size=64,
        seed=1)

pred_datagen = ImageDataGenerator()
prediction_generator = pred_datagen.flow_from_directory(
        test_dir,
        shuffle=False,
        target_size=(150, 150),
        batch_size=64,
        seed=1)
```

```
Found 14034 images belonging to 6 classes.
Found 3000 images belonging to 6 classes.
Found 7301 images belonging to 6 classes.
```

```
#Resnet based model with weights from imagenet
model = ResNet50(weights='imagenet', include_top=False, input_shape=(150, 150, 3))


# Adding Custom layers to the model
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


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

```
Model: "model_6"
```

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	[(None, 150, 150, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 156, 156, 3)	0	input_7[0][0]
conv1_conv (Conv2D)	(None, 75, 75, 64)	9472	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 75, 75, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 75, 75, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 77, 77, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 38, 38, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 38, 38, 64)	4160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormali	(None, 38, 38, 64)	256	conv2_block1_1_conv[0][0]
conv2_block1_1_relu (Activation	(None, 38, 38, 64)	0	conv2_block1_1_bn[0][0]
conv2_block1_2_conv (Conv2D)	(None, 38, 38, 64)	36928	conv2_block1_1_relu[0][0]
conv2_block1_2_bn (BatchNormali	(None, 38, 38, 64)	256	conv2_block1_2_conv[0][0]
conv2_block1_2_relu (Activation	(None, 38, 38, 64)	0	conv2_block1_2_bn[0][0]
conv2_block1_0_conv (Conv2D)	(None, 38, 38, 256)	16640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 38, 38, 256)	16640	conv2_block1_2_relu[0][0]
conv2_block1_0_bn (BatchNormali	(None, 38, 38, 256)	1024	conv2_block1_0_conv[0][0]
conv2_block1_3_bn (BatchNormali	(None, 38, 38, 256)	1024	conv2_block1_3_conv[0][0]
conv2_block1_add (Add)	(None, 38, 38, 256)	0	conv2_block1_0_bn[0][0] conv2_block1_3_bn[0][0]
conv2_block1_out (Activation)	(None, 38, 38, 256)	0	conv2_block1_add[0][0]
conv2_block2_1_conv (Conv2D)	(None, 38, 38, 64)	16448	conv2_block1_out[0][0]

conv2_block2_1_bn (BatchNormali	(None, 38, 38, 64)	256	conv2_block2_1_conv[0][0]
conv2_block2_1_relu (Activation	(None, 38, 38, 64)	0	conv2_block2_1_bn[0][0]
conv2_block2_2_conv (Conv2D)	(None, 38, 38, 64)	36928	conv2_block2_1_relu[0][0]
conv2_block2_2_bn (BatchNormali	(None, 38, 38, 64)	256	conv2_block2_2_conv[0][0]
conv2_block2_2_relu (Activation	(None, 38, 38, 64)	0	conv2_block2_2_bn[0][0]
conv2_block2_3_conv (Conv2D)	(None, 38, 38, 256)	16640	conv2_block2_2_relu[0][0]
conv2_block2_3_bn (BatchNormali	(None, 38, 38, 256)	1024	conv2_block2_3_conv[0][0]

```
# Compiling SGD opt based model with learning rate=0.001 and momentum=0.9
from keras.optimizers import SGD
sgd = SGD(lr=0.001, decay=1e-7, momentum=.9)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
```

Model Training

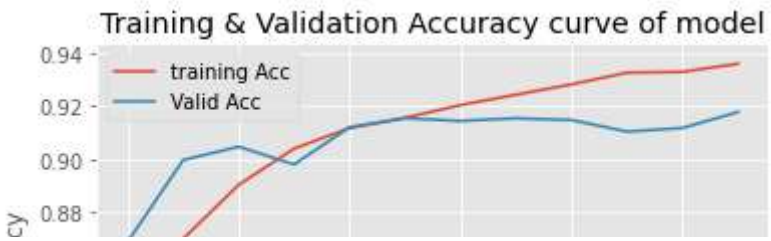
```
# Training the model
H = model.fit(
      train_generator,
      epochs=12,
      validation_data=validation_generator,
      verbose=1)
```

```
Epoch 1/12
220/220 [==============================] - 132s 579ms/step - loss: 0.9751 - accuracy: 0.7021 - val_loss: 0.3780 - val_a
Epoch 2/12
220/220 [==============================] - 124s 562ms/step - loss: 0.3669 - accuracy: 0.8679 - val_loss: 0.2868 - val_a
Epoch 3/12
220/220 [==============================] - 122s 555ms/step - loss: 0.2983 - accuracy: 0.8914 - val_loss: 0.2699 - val_a
Epoch 4/12
220/220 [==============================] - 122s 552ms/step - loss: 0.2726 - accuracy: 0.9022 - val_loss: 0.2926 - val_a
Epoch 5/12
```

```
220/220 [==============================] - 124s 563ms/step - loss: 0.2365 - accuracy: 0.9124 - val_loss: 0.2536 - val_a
Epoch 6/12
220/220 [==============================] - 124s 565ms/step - loss: 0.2160 - accuracy: 0.9197 - val_loss: 0.2411 - val_a
Epoch 7/12
220/220 [==============================] - 124s 564ms/step - loss: 0.2112 - accuracy: 0.9239 - val_loss: 0.2374 - val_a
Epoch 8/12
220/220 [==============================] - 125s 569ms/step - loss: 0.2073 - accuracy: 0.9250 - val_loss: 0.2465 - val_a
Epoch 9/12
220/220 [==============================] - 126s 571ms/step - loss: 0.1913 - accuracy: 0.9319 - val_loss: 0.2362 - val_a
Epoch 10/12
220/220 [==============================] - 126s 573ms/step - loss: 0.1819 - accuracy: 0.9329 - val_loss: 0.2628 - val_a
Epoch 11/12
220/220 [==============================] - 125s 569ms/step - loss: 0.1731 - accuracy: 0.9346 - val_loss: 0.2486 - val_a
Epoch 12/12
220/220 [==============================] - 126s 573ms/step - loss: 0.1727 - accuracy: 0.9364 - val_loss: 0.2285 - val_a
```

```
# save the model's trained weights
mdlpthwgts = "/content/drive/MyDrive/"
model.save_weights(mdlpthwgts + 'weights_of_mdl0.41.h5')


'''Training Acc CURVE'''
plt.plot(H.history['accuracy'])
plt.plot(H.history['val_accuracy'])
plt.title('Training & Validation Accuracy curve of model')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['training Acc', 'Valid Acc'], loc='upper left')
plt.show()
```




```
plt.plot(H.history['loss'])
plt.plot(H.history['val_loss'])
plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['training Loss', 'Valid Loss'], loc='upper left')
plt.show()
```


```
# evaluate the model on a test dataset
results = model.evaluate(prediction_generator, batch_size=64, verbose=1)
print("test loss, test acc:", results)
```

```
115/115 [==============================] - 16s 140ms/step - loss: 0.5046 - accuracy: 0.8951
test loss, test acc: [0.5045585036277771, 0.8950828909873962]
```

Confusion matrix Plot

```
prediction = model.predict(prediction_generator, batch_size=64, verbose=1)
```

```
115/115 [==============================] - 16s 136ms/step
```

```
from sklearn.metrics import classification_report, confusion_matrix
pred = np.argmax(prediction, axis=1)
print(confusion_matrix(prediction_generator.classes, pred))
```

```
[[1009    4    2    1   14  114]
 [   5 1152    1    2    4    2]
 [   4    8 1104  155   52    7]
 [   5   12  122 1080   73    5]
 [   9    6   25   53 1026    9]
 [  47   10    2    3   10 1164]]
```

Classification Report

```
print('Classification Report:')
target_names = ['Buildings', 'Forest', 'Glacier', 'Mountain', 'Sea', 'Street']
print(classification_report(prediction_generator.classes, pred, target_names=target_names))
```

```
Classification Report:
              precision    recall  f1-score   support

   Buildings       0.94      0.88      0.91      1144
      Forest       0.97      0.99      0.98      1166
     Glacier       0.88      0.83      0.85      1330
    Mountain       0.83      0.83      0.83      1297
         Sea       0.87      0.91      0.89      1128
      Street       0.89      0.94      0.92      1236

    accuracy                           0.90      7301
   macro avg       0.90      0.90      0.90      7301
weighted avg       0.90      0.90      0.89      7301
```

Heatmap of color encoded Confusion matrix

```
confusion_mtx= confusion_matrix(prediction_generator.classes, pred)
import seaborn as sns
sns.heatmap(confusion_mtx, xticklabels=target_names, yticklabels=target_names,
            annot=True, fmt='d', cmap="YlGnBu")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd321493780>
```