Assignment 3

Name: Ali Ahmed
Department: *SEECS*
Organization: *NUST,H-12*

Email:
aahmed.mscs20seecs@seecs.edu.pk

***Abstract*— This assignment tasked the students to classify the images in the given dataset into 6 categories using Neural Networks. The dataset used in the assignment is the Intel-Image-Classification which contained close to 25000 images in 6 classes, Buildings, Street, Mountain, Forest, Glacier and Sea. The students were encouraged to solve the assignment using any CNN architecture of their liking whether using building block layers to make the architecture or use pre-fabricated used implemented in various libraries. Finally there was an optional task to add data augmentation techniques discussed in class.**

I. Introduction

My attempt at the assignment was focused on learning how to both fabricate a network to suit my needs as well as to use a pre-fabricated architecture, to this end, I attempted two models. First implementation was a CNN using primitive building block layers available in Tensor Flow. The second attempt was to import the ResNet50 architecture from Keras (pre-trained on the dataset). During the image pre-processing some data-augmentation parameters were passed to the Image Data Generator with the hope of improving accuracy.

II. Methodology

The first part was to download the dataset from the given link and unzip them, this was done from the code shared in class and the next idea was to display the images with their labels to get a feel of the dataset.

Next Image Data Generator was imported from Tensor Flow and some data augmentation parameters was passed to help improve the accuracy.

```
train_datagen = ImageDataGenerator(rescale = 1./255.,horizontal_flip=True,shear_range=0.2,
    zoom_range=0.2, validation_split=0.1)
```

Four data-generators were invoked, training, validation, test and test1, with different batch sizes and other parameters. My CNN model used primitive layers from Tensor flow, the model itself was sequential. The architecture is given below.

```
tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape=(150,150,3)),
tf.keras.layers.Conv2D(16,(3,3),activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D(3,3),

tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
tf.keras.layers.BatchNormalization(),

tf.keras.layers.Conv2D(256,(3,3),activation='relu',padding='same'),
tf.keras.layers.Conv2D(256,(3,3),activation='relu',padding='same'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.MaxPooling2D(2,2),
tf.keras.layers.Dropout(0.4),
```

```
tf.keras.layers.Flatten(),

tf.keras.layers.Dense(256,activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.25),
tf.keras.layers.Dense(128,activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.25),
tf.keras.layers.Dense(64,activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.25),
tf.keras.layers.Dense(32,activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.25),
tf.keras.layers.Dense(16,activation='relu'),
tf.keras.layers.BatchNormalization(),
tf.keras.layers.Dropout(0.4),

tf.keras.layers.Dense(6,activation='softmax')
```

The second attempt used ResNet50 architecture from Keras. This one was pretrained on the image net dataset as the figure shows.

```
from keras.applications.resnet50 import ResNet50
import keras


base_model2=ResNet50(include_top=False, weights= 'imagenet',  input_shape=(150,150,3), pooling='avg')
base_model2.trainable = False

x = Dense(512, activation='relu')(base_model2.output)
x = Dropout(0.5)(x)
x = Dense(6, activation='softmax')(x)

transfer_model2 = Model(base_model2.input, x)
transfer_model2.compile(optimizer =keras.optimizers.SGD(lr=0.0001),
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

III. Results

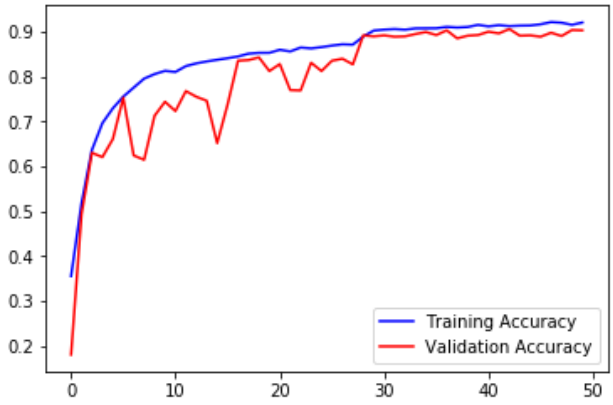
Before I discuss the results of the two models on the dataset, let us look at some visual results.

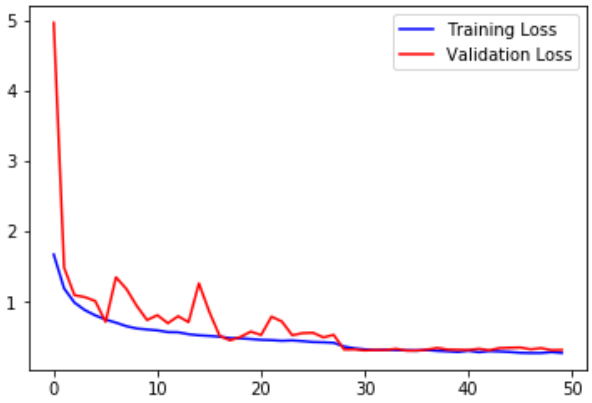
The first CNN model achieved the following accuracy on the testset:

```
[ ] accuracy=model1.evaluate_generator(test_generato
    print('Accuracy of the model on the test set: ',

2/2 - 0s - loss: 0.3731 - acc: 0.9062
Accuracy of the model on the test set:  0.90625
```

Plots showing the training and validation accuracy/loss per epoch are given below for model1:

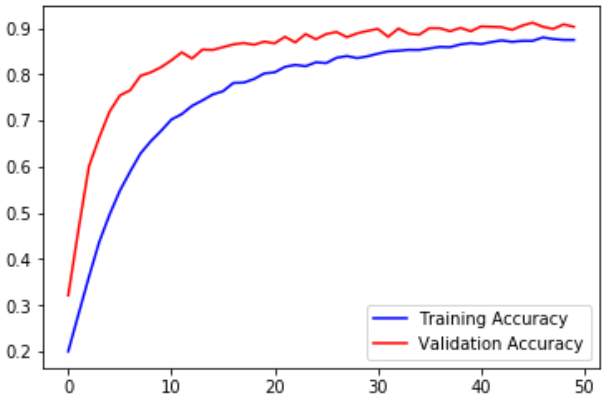


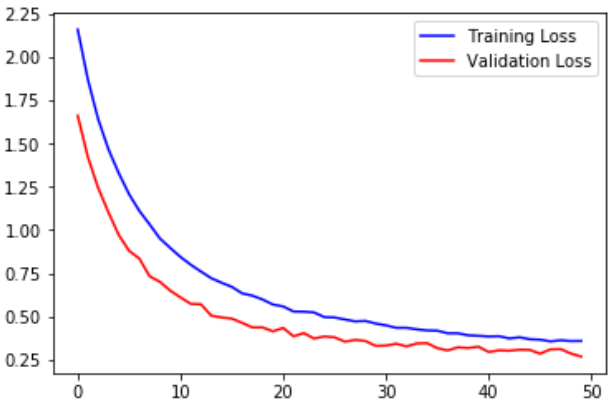


Similarly for Model2, ResNet50:

```
[ ] accuracy2=transfer_model2.evaluate_generator(test_
    print('Accuracy of the model on the test set: ',ac

Accuracy of the model on the test set:  0.953125
```





LINKS

[1] https://github.com/hid3nZm0k3y/Assignment-3.git