GNR 638 Mini Project 1

Name : Sarvesh Patil(22b2276) and Stuti Singla(22b0046)

Final code after tring out different pretrainded models

Phase 1 : Data importing and preprocessing.

```
# Importing important libraries
import os
import random
import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D,
Dropout, Flatten

from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter)
from dataclasses import dataclass

from tensorflow.keras import models
from keras.models import Model

# Setting few variables
N = 11788
t = 64
epoch = 0

import torch

# Read data from train_test_split to seperate training and testing
data.
with open("CUB_200_2011/train_test_split.txt", "r") as file:
    data = file.readlines()

# Convert data to list of floats
data = [list(map(int, line.strip().split())) for line in data]

# Convert list to tensor
tensor_data = torch.tensor(data)


# Storing indices of train and test in two different list.
train_index = []
```

```
test_index = []

for i in range(0,N):
    if tensor_data[i][1] == 1:
        train_index.append(i)
    else:
        test_index.append(i)

# Creating Y train and y test data sets
with open("CUB_200_2011/image_class_labels.txt", "r") as file:
    data = file.readlines()

# Convert data to list of floats
data = [list(map(int, line.strip().split())) for line in data]

# Convert list to tensor
class_labels_num = torch.tensor(data)

y_train = []
y_test = []
for i in range(0, N):
    if i in train_index:
        y_train.append(int(class_labels_num[i][1]))
    else:
        y_test.append(int(class_labels_num[i][1]))

# Creating x train and x test datasets
with open("CUB_200_2011/images.txt", 'r') as file:
    lines = file.readlines()

data = []
# creating dataset of file locations of images
for line in lines:
    parts = line.split()
    num = int(parts[0])
    link = ' '.join(parts[1:])
    link= ".//CUB_200_2011//images//"+link
    data.append([num, link])

data_array = np.array(data)

from PIL import Image
import numpy as np

# Creating two list ie x_train and x_test containing array of image
matrix
x_train = []
x_test = []

for i, item in enumerate(data_array, start=1):
    link = item[1]
```

```
    try:
        image = Image.open(link)

        resized_image = image.resize((t, t))  # Resize the image to
txt pixels
        resized_image_array = np.array(resized_image)   # Convert the
resized image to a NumPy array
        if i in train_index:
            x_train.append(resized_image_array)
        else:
            x_test.append(resized_image_array)
    except Exception as e:
        print(f"Error loading image from {link}: {e}")

# Now x_train contains the resized images loaded from the links
corresponding to indices present in train_index
# and x_test contains the resized images loaded from the links
corresponding to indices not present in train_index
```

while analyising dimensions of data we found that few images doesnt have RGB values.

8 such images are found.

because, the number is less we are not considering them for training or testing purpose.

```
# TO drop images whoes dimensions are not consistant with requirnments
of model
deleted_test_sample_index = [] #this index is with respect test sample

deleted_train_sample_index = [] #This index is with respect to train
sample

# finding index of samples having no RGB dimensions (Number of such
images are very less. so deleting them seems to be a good option)
for i in range(len(x_test)):
    if (x_test[i].shape)!=(t,t,3):
        deleted_test_sample_index.append(i)
        print(f'{i}th element of x test is found inhomogenous')
for j in range(len(x_train)):
    if (x_train[j].shape)!=(t,t,3):
        deleted_train_sample_index.append(j)
        print(f'{j}th element of x train is found inhomogenous')

print(deleted_test_sample_index)
print(deleted_train_sample_index)

# Droping thoes datapoints from dataset
for i in range(len(deleted_test_sample_index)):
    x_test.pop(deleted_test_sample_index[i]-i)
    y_test.pop(deleted_test_sample_index[i]-i)
```

```
    print(deleted_test_sample_index[i])

for i in range(len(deleted_train_sample_index)):
    x_train.pop(deleted_train_sample_index[i]-i)
    y_train.pop(deleted_train_sample_index[i]-i)
    print(deleted_train_sample_index[i])
```

```
195th element of x test is found inhomogenous
3081th element of x test is found inhomogenous
749th element of x train is found inhomogenous
1868th element of x train is found inhomogenous
1869th element of x train is found inhomogenous
1951th element of x train is found inhomogenous
2580th element of x train is found inhomogenous
2762th element of x train is found inhomogenous
[195, 3081]
[749, 1868, 1869, 1951, 2580, 2762]
195
3081
749
1868
1869
1951
2580
2762
```

```
x_train = np.array(x_train)
x_test = np.array(x_test)

# Normalize images to the range [0, 1].
x_train = x_train.astype("float32") / 255
x_test  = x_test.astype("float32") / 255

#TO convert y train and test set into categorical format
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
print(y_train)
```

```
[[0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]]
```

Now, our data preprocessing is done and data is ready to load into different models of our choice.

Phase 1 end

Phase 2 : Defining model

```
# Defining complete model
import efficientnet.keras as efn
base_model = efn.EfficientNetB0(input_shape = (t, t, 3), include_top =
False, weights = 'imagenet')

# Including fine tuning for last few layers
fine_tune = 5
for layer in base_model.layers:
            layer.trainable = False

if fine_tune > 0:
        for layer in base_model.layers[len(base_model.layers)-
fine_tune:len(base_model.layers)]:
            layer.trainable = True

# Adding few layers to help model classify images of our choice
x = base_model.output
x = Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.5)(x)

# Add a final sigmoid layer with 1 node for classification output
x = Dense(201, activation="softmax")(x)
model = Model(base_model.input, x)

from keras.optimizers import RMSprop
model.compile(optimizer = RMSprop(learning_rate=0.0001), loss =
'categorical_crossentropy', metrics = ['acc'])
```

Here, we are using RMSprop as optimizer

```
# To get summary of our model.
model.summary()
```

```
Model: "model"
_____________________________________________________________________
_____________________________
 Layer (type)                Output Shape              Param #
Connected to
=====================================================================
============================
 input_1 (InputLayer)        [(None, 64, 64, 3)]       0         []



 stem_conv (Conv2D)          (None, 32, 32, 32)        864
```

```
['input_1[0][0]']


 stem_bn (BatchNormalizatio  (None, 32, 32, 32)           128
['stem_conv[0][0]']
 n)



 stem_activation (Activatio  (None, 32, 32, 32)           0
['stem_bn[0][0]']
 n)



 block1a_dwconv (DepthwiseC  (None, 32, 32, 32)           288
['stem_activation[0][0]']
 onv2D)



 block1a_bn (BatchNormaliza  (None, 32, 32, 32)           128
['block1a_dwconv[0][0]']
 tion)



 block1a_activation (Activa  (None, 32, 32, 32)           0
['block1a_bn[0][0]']
 tion)



 block1a_se_squeeze (Global  (None, 32)                   0
['block1a_activation[0][0]']
 AveragePooling2D)



 block1a_se_reshape (Reshap  (None, 1, 1, 32)             0
['block1a_se_squeeze[0][0]']
 e)



 block1a_se_reduce (Conv2D)  (None, 1, 1, 8)              264
['block1a_se_reshape[0][0]']


 block1a_se_expand (Conv2D)  (None, 1, 1, 32)             288
['block1a_se_reduce[0][0]']
```

```
 block1a_se_excite (Multipl  (None, 32, 32, 32)           0
['block1a_activation[0][0]',
 y)
'block1a_se_expand[0][0]']


 block1a_project_conv (Conv  (None, 32, 32, 16)           512
['block1a_se_excite[0][0]']
 2D)



 block1a_project_bn (BatchN  (None, 32, 32, 16)           64
['block1a_project_conv[0][0]']
 ormalization)



 block2a_expand_conv (Conv2  (None, 32, 32, 96)           1536
['block1a_project_bn[0][0]']
 D)



 block2a_expand_bn (BatchNo  (None, 32, 32, 96)           384
['block2a_expand_conv[0][0]']
 rmalization)



 block2a_expand_activation   (None, 32, 32, 96)           0
['block2a_expand_bn[0][0]']
 (Activation)



 block2a_dwconv (DepthwiseC  (None, 16, 16, 96)           864
['block2a_expand_activation[0]
 onv2D)
[0]']


 block2a_bn (BatchNormaliza  (None, 16, 16, 96)           384
['block2a_dwconv[0][0]']
 tion)



 block2a_activation (Activa  (None, 16, 16, 96)           0
```

```
['block2a_bn[0][0]']
 tion)



 block2a_se_squeeze (Global  (None, 96)                    0
['block2a_activation[0][0]']
 AveragePooling2D)



 block2a_se_reshape (Reshap  (None, 1, 1, 96)              0
['block2a_se_squeeze[0][0]']
 e)



 block2a_se_reduce (Conv2D)  (None, 1, 1, 4)               388
['block2a_se_reshape[0][0]']


 block2a_se_expand (Conv2D)  (None, 1, 1, 96)              480
['block2a_se_reduce[0][0]']


 block2a_se_excite (Multipl  (None, 16, 16, 96)            0
['block2a_activation[0][0]',
 y)
'block2a_se_expand[0][0]']


 block2a_project_conv (Conv  (None, 16, 16, 24)            2304
['block2a_se_excite[0][0]']
 2D)



 block2a_project_bn (BatchN  (None, 16, 16, 24)            96
['block2a_project_conv[0][0]']
 ormalization)



 block2b_expand_conv (Conv2  (None, 16, 16, 144)           3456
['block2a_project_bn[0][0]']
 D)



 block2b_expand_bn (BatchNo  (None, 16, 16, 144)           576
['block2b_expand_conv[0][0]']
```

```
 rmalization)


 block2b_expand_activation   (None, 16, 16, 144)          0
['block2b_expand_bn[0][0]']
 (Activation)


 block2b_dwconv (DepthwiseC  (None, 16, 16, 144)          1296
['block2b_expand_activation[0]
 onv2D)
[0]']


 block2b_bn (BatchNormaliza  (None, 16, 16, 144)          576
['block2b_dwconv[0][0]']
 tion)


 block2b_activation (Activa  (None, 16, 16, 144)          0
['block2b_bn[0][0]']
 tion)


 block2b_se_squeeze (Global  (None, 144)                  0
['block2b_activation[0][0]']
 AveragePooling2D)


 block2b_se_reshape (Reshap  (None, 1, 1, 144)            0
['block2b_se_squeeze[0][0]']
 e)


 block2b_se_reduce (Conv2D)  (None, 1, 1, 6)              870
['block2b_se_reshape[0][0]']


 block2b_se_expand (Conv2D)  (None, 1, 1, 144)            1008
['block2b_se_reduce[0][0]']


 block2b_se_excite (Multipl  (None, 16, 16, 144)          0
['block2b_activation[0][0]',
 y)
```

```
'block2b_se_expand[0][0]']


 block2b_project_conv (Conv  (None, 16, 16, 24)           3456
['block2b_se_excite[0][0]']
 2D)



 block2b_project_bn (BatchN  (None, 16, 16, 24)           96
['block2b_project_conv[0][0]']
 ormalization)



 block2b_drop (FixedDropout  (None, 16, 16, 24)           0
['block2b_project_bn[0][0]']
 )



 block2b_add (Add)           (None, 16, 16, 24)           0
['block2b_drop[0][0]',

'block2a_project_bn[0][0]']


 block3a_expand_conv (Conv2  (None, 16, 16, 144)          3456
['block2b_add[0][0]']
 D)



 block3a_expand_bn (BatchNo  (None, 16, 16, 144)          576
['block3a_expand_conv[0][0]']
 rmalization)



 block3a_expand_activation   (None, 16, 16, 144)          0
['block3a_expand_bn[0][0]']
 (Activation)



 block3a_dwconv (DepthwiseC  (None, 8, 8, 144)            3600
['block3a_expand_activation[0]
 onv2D)
[0]']
```

```
 block3a_bn (BatchNormaliza  (None, 8, 8, 144)            576
['block3a_dwconv[0][0]']
 tion)


 block3a_activation (Activa  (None, 8, 8, 144)            0
['block3a_bn[0][0]']
 tion)


 block3a_se_squeeze (Global  (None, 144)                  0
['block3a_activation[0][0]']
 AveragePooling2D)


 block3a_se_reshape (Reshap  (None, 1, 1, 144)            0
['block3a_se_squeeze[0][0]']
 e)


 block3a_se_reduce (Conv2D)  (None, 1, 1, 6)              870
['block3a_se_reshape[0][0]']


 block3a_se_expand (Conv2D)  (None, 1, 1, 144)            1008
['block3a_se_reduce[0][0]']


 block3a_se_excite (Multipl  (None, 8, 8, 144)            0
['block3a_activation[0][0]',
 y)
'block3a_se_expand[0][0]']


 block3a_project_conv (Conv  (None, 8, 8, 40)             5760
['block3a_se_excite[0][0]']
 2D)


 block3a_project_bn (BatchN  (None, 8, 8, 40)             160
['block3a_project_conv[0][0]']
 ormalization)


 block3b_expand_conv (Conv2  (None, 8, 8, 240)            9600
```

```
['block3a_project_bn[0][0]']
 D)



 block3b_expand_bn (BatchNo  (None, 8, 8, 240)             960
['block3b_expand_conv[0][0]']
 rmalization)



 block3b_expand_activation   (None, 8, 8, 240)             0
['block3b_expand_bn[0][0]']
 (Activation)



 block3b_dwconv (DepthwiseC  (None, 8, 8, 240)             6000
['block3b_expand_activation[0]
 onv2D)
[0]']


 block3b_bn (BatchNormaliza  (None, 8, 8, 240)             960
['block3b_dwconv[0][0]']
 tion)



 block3b_activation (Activa  (None, 8, 8, 240)             0
['block3b_bn[0][0]']
 tion)



 block3b_se_squeeze (Global  (None, 240)                   0
['block3b_activation[0][0]']
 AveragePooling2D)



 block3b_se_reshape (Reshap  (None, 1, 1, 240)             0
['block3b_se_squeeze[0][0]']
 e)



 block3b_se_reduce (Conv2D)  (None, 1, 1, 10)              2410
['block3b_se_reshape[0][0]']
```

```
 block3b_se_expand (Conv2D)  (None, 1, 1, 240)            2640
['block3b_se_reduce[0][0]']


 block3b_se_excite (Multipl  (None, 8, 8, 240)            0
['block3b_activation[0][0]',
 y)
'block3b_se_expand[0][0]']


 block3b_project_conv (Conv  (None, 8, 8, 40)             9600
['block3b_se_excite[0][0]']
 2D)



 block3b_project_bn (BatchN  (None, 8, 8, 40)             160
['block3b_project_conv[0][0]']
 ormalization)



 block3b_drop (FixedDropout  (None, 8, 8, 40)             0
['block3b_project_bn[0][0]']
 )



 block3b_add (Add)           (None, 8, 8, 40)             0
['block3b_drop[0][0]',

'block3a_project_bn[0][0]']


 block4a_expand_conv (Conv2  (None, 8, 8, 240)            9600
['block3b_add[0][0]']
 D)



 block4a_expand_bn (BatchNo  (None, 8, 8, 240)            960
['block4a_expand_conv[0][0]']
 rmalization)



 block4a_expand_activation   (None, 8, 8, 240)            0
['block4a_expand_bn[0][0]']
 (Activation)
```

```
 block4a_dwconv (DepthwiseC  (None, 4, 4, 240)            2160
['block4a_expand_activation[0]
 onv2D)
[0]']


 block4a_bn (BatchNormaliza  (None, 4, 4, 240)            960
['block4a_dwconv[0][0]']
 tion)



 block4a_activation (Activa  (None, 4, 4, 240)            0
['block4a_bn[0][0]']
 tion)



 block4a_se_squeeze (Global  (None, 240)                  0
['block4a_activation[0][0]']
 AveragePooling2D)



 block4a_se_reshape (Reshap  (None, 1, 1, 240)            0
['block4a_se_squeeze[0][0]']
 e)



 block4a_se_reduce (Conv2D)  (None, 1, 1, 10)             2410
['block4a_se_reshape[0][0]']


 block4a_se_expand (Conv2D)  (None, 1, 1, 240)            2640
['block4a_se_reduce[0][0]']


 block4a_se_excite (Multipl  (None, 4, 4, 240)            0
['block4a_activation[0][0]',
 y)
'block4a_se_expand[0][0]']


 block4a_project_conv (Conv  (None, 4, 4, 80)             19200
['block4a_se_excite[0][0]']
 2D)
```

```
 block4a_project_bn (BatchN  (None, 4, 4, 80)             320
['block4a_project_conv[0][0]']
 ormalization)



 block4b_expand_conv (Conv2  (None, 4, 4, 480)            38400
['block4a_project_bn[0][0]']
 D)



 block4b_expand_bn (BatchNo  (None, 4, 4, 480)            1920
['block4b_expand_conv[0][0]']
 rmalization)



 block4b_expand_activation   (None, 4, 4, 480)            0
['block4b_expand_bn[0][0]']
 (Activation)



 block4b_dwconv (DepthwiseC  (None, 4, 4, 480)            4320
['block4b_expand_activation[0]
 onv2D)
[0]']


 block4b_bn (BatchNormaliza  (None, 4, 4, 480)            1920
['block4b_dwconv[0][0]']
 tion)



 block4b_activation (Activa  (None, 4, 4, 480)            0
['block4b_bn[0][0]']
 tion)



 block4b_se_squeeze (Global  (None, 480)                  0
['block4b_activation[0][0]']
 AveragePooling2D)



 block4b_se_reshape (Reshap  (None, 1, 1, 480)            0
['block4b_se_squeeze[0][0]']
 e)
```

```
 block4b_se_reduce (Conv2D)  (None, 1, 1, 20)              9620
['block4b_se_reshape[0][0]']


 block4b_se_expand (Conv2D)  (None, 1, 1, 480)             10080
['block4b_se_reduce[0][0]']


 block4b_se_excite (Multipl  (None, 4, 4, 480)             0
['block4b_activation[0][0]',
 y)
'block4b_se_expand[0][0]']


 block4b_project_conv (Conv  (None, 4, 4, 80)              38400
['block4b_se_excite[0][0]']
 2D)



 block4b_project_bn (BatchN  (None, 4, 4, 80)              320
['block4b_project_conv[0][0]']
 ormalization)



 block4b_drop (FixedDropout  (None, 4, 4, 80)              0
['block4b_project_bn[0][0]']
 )



 block4b_add (Add)           (None, 4, 4, 80)              0
['block4b_drop[0][0]',

'block4a_project_bn[0][0]']


 block4c_expand_conv (Conv2  (None, 4, 4, 480)             38400
['block4b_add[0][0]']
 D)



 block4c_expand_bn (BatchNo  (None, 4, 4, 480)             1920
['block4c_expand_conv[0][0]']
 rmalization)
```

```
 block4c_expand_activation   (None, 4, 4, 480)            0
['block4c_expand_bn[0][0]']
 (Activation)


 block4c_dwconv (DepthwiseC  (None, 4, 4, 480)            4320
['block4c_expand_activation[0]
 onv2D)
[0]']


 block4c_bn (BatchNormaliza  (None, 4, 4, 480)            1920
['block4c_dwconv[0][0]']
 tion)


 block4c_activation (Activa  (None, 4, 4, 480)            0
['block4c_bn[0][0]']
 tion)


 block4c_se_squeeze (Global  (None, 480)                  0
['block4c_activation[0][0]']
 AveragePooling2D)


 block4c_se_reshape (Reshap  (None, 1, 1, 480)            0
['block4c_se_squeeze[0][0]']
 e)


 block4c_se_reduce (Conv2D)  (None, 1, 1, 20)             9620
['block4c_se_reshape[0][0]']


 block4c_se_expand (Conv2D)  (None, 1, 1, 480)            10080
['block4c_se_reduce[0][0]']


 block4c_se_excite (Multipl  (None, 4, 4, 480)            0
['block4c_activation[0][0]',
 y)
'block4c_se_expand[0][0]']
```

```
 block4c_project_conv (Conv  (None, 4, 4, 80)             38400
['block4c_se_excite[0][0]']
 2D)



 block4c_project_bn (BatchN  (None, 4, 4, 80)             320
['block4c_project_conv[0][0]']
 ormalization)



 block4c_drop (FixedDropout  (None, 4, 4, 80)             0
['block4c_project_bn[0][0]']
 )



 block4c_add (Add)           (None, 4, 4, 80)             0
['block4c_drop[0][0]',

'block4b_add[0][0]']


 block5a_expand_conv (Conv2  (None, 4, 4, 480)            38400
['block4c_add[0][0]']
 D)



 block5a_expand_bn (BatchNo  (None, 4, 4, 480)            1920
['block5a_expand_conv[0][0]']
 rmalization)



 block5a_expand_activation   (None, 4, 4, 480)            0
['block5a_expand_bn[0][0]']
 (Activation)



 block5a_dwconv (DepthwiseC  (None, 4, 4, 480)            12000
['block5a_expand_activation[0]
 onv2D)
[0]']


 block5a_bn (BatchNormaliza  (None, 4, 4, 480)            1920
['block5a_dwconv[0][0]']
```

```
 tion)


 block5a_activation (Activa  (None, 4, 4, 480)             0
['block5a_bn[0][0]']
 tion)


 block5a_se_squeeze (Global  (None, 480)                   0
['block5a_activation[0][0]']
 AveragePooling2D)


 block5a_se_reshape (Reshap  (None, 1, 1, 480)             0
['block5a_se_squeeze[0][0]']
 e)


 block5a_se_reduce (Conv2D)  (None, 1, 1, 20)              9620
['block5a_se_reshape[0][0]']


 block5a_se_expand (Conv2D)  (None, 1, 1, 480)             10080
['block5a_se_reduce[0][0]']


 block5a_se_excite (Multipl  (None, 4, 4, 480)             0
['block5a_activation[0][0]',
 y)
'block5a_se_expand[0][0]']


 block5a_project_conv (Conv  (None, 4, 4, 112)             53760
['block5a_se_excite[0][0]']
 2D)


 block5a_project_bn (BatchN  (None, 4, 4, 112)             448
['block5a_project_conv[0][0]']
 ormalization)


 block5b_expand_conv (Conv2  (None, 4, 4, 672)             75264
['block5a_project_bn[0][0]']
 D)
```

```
 block5b_expand_bn (BatchNo  (None, 4, 4, 672)            2688
['block5b_expand_conv[0][0]']
 rmalization)


 block5b_expand_activation   (None, 4, 4, 672)            0
['block5b_expand_bn[0][0]']
 (Activation)


 block5b_dwconv (DepthwiseC  (None, 4, 4, 672)            16800
['block5b_expand_activation[0]
 onv2D)
[0]']


 block5b_bn (BatchNormaliza  (None, 4, 4, 672)            2688
['block5b_dwconv[0][0]']
 tion)


 block5b_activation (Activa  (None, 4, 4, 672)            0
['block5b_bn[0][0]']
 tion)


 block5b_se_squeeze (Global  (None, 672)                  0
['block5b_activation[0][0]']
 AveragePooling2D)


 block5b_se_reshape (Reshap  (None, 1, 1, 672)            0
['block5b_se_squeeze[0][0]']
 e)


 block5b_se_reduce (Conv2D)  (None, 1, 1, 28)             18844
['block5b_se_reshape[0][0]']


 block5b_se_expand (Conv2D)  (None, 1, 1, 672)            19488
['block5b_se_reduce[0][0]']
```

```
 block5b_se_excite (Multipl  (None, 4, 4, 672)            0
['block5b_activation[0][0]',
 y)
'block5b_se_expand[0][0]']


 block5b_project_conv (Conv  (None, 4, 4, 112)            75264
['block5b_se_excite[0][0]']
 2D)



 block5b_project_bn (BatchN  (None, 4, 4, 112)            448
['block5b_project_conv[0][0]']
 ormalization)



 block5b_drop (FixedDropout  (None, 4, 4, 112)            0
['block5b_project_bn[0][0]']
 )



 block5b_add (Add)           (None, 4, 4, 112)            0
['block5b_drop[0][0]',

'block5a_project_bn[0][0]']


 block5c_expand_conv (Conv2  (None, 4, 4, 672)            75264
['block5b_add[0][0]']
 D)



 block5c_expand_bn (BatchNo  (None, 4, 4, 672)            2688
['block5c_expand_conv[0][0]']
 rmalization)



 block5c_expand_activation   (None, 4, 4, 672)            0
['block5c_expand_bn[0][0]']
 (Activation)



 block5c_dwconv (DepthwiseC  (None, 4, 4, 672)            16800
```

```
['block5c_expand_activation[0]
 onv2D)
[0]']


 block5c_bn (BatchNormaliza  (None, 4, 4, 672)            2688
['block5c_dwconv[0][0]']
 tion)



 block5c_activation (Activa  (None, 4, 4, 672)            0
['block5c_bn[0][0]']
 tion)



 block5c_se_squeeze (Global  (None, 672)                  0
['block5c_activation[0][0]']
 AveragePooling2D)



 block5c_se_reshape (Reshap  (None, 1, 1, 672)            0
['block5c_se_squeeze[0][0]']
 e)



 block5c_se_reduce (Conv2D)  (None, 1, 1, 28)             18844
['block5c_se_reshape[0][0]']


 block5c_se_expand (Conv2D)  (None, 1, 1, 672)            19488
['block5c_se_reduce[0][0]']


 block5c_se_excite (Multipl  (None, 4, 4, 672)            0
['block5c_activation[0][0]',
 y)
'block5c_se_expand[0][0]']


 block5c_project_conv (Conv  (None, 4, 4, 112)            75264
['block5c_se_excite[0][0]']
 2D)



 block5c_project_bn (BatchN  (None, 4, 4, 112)            448
['block5c_project_conv[0][0]']
```

```
 ormalization)


 block5c_drop (FixedDropout  (None, 4, 4, 112)             0
['block5c_project_bn[0][0]']
 )


 block5c_add (Add)           (None, 4, 4, 112)             0
['block5c_drop[0][0]',

'block5b_add[0][0]']


 block6a_expand_conv (Conv2  (None, 4, 4, 672)             75264
['block5c_add[0][0]']
 D)


 block6a_expand_bn (BatchNo  (None, 4, 4, 672)             2688
['block6a_expand_conv[0][0]']
 rmalization)


 block6a_expand_activation   (None, 4, 4, 672)             0
['block6a_expand_bn[0][0]']
 (Activation)


 block6a_dwconv (DepthwiseC  (None, 2, 2, 672)             16800
['block6a_expand_activation[0]
 onv2D)
[0]']


 block6a_bn (BatchNormaliza  (None, 2, 2, 672)             2688
['block6a_dwconv[0][0]']
 tion)


 block6a_activation (Activa  (None, 2, 2, 672)             0
['block6a_bn[0][0]']
 tion)
```

```
 block6a_se_squeeze (Global  (None, 672)                  0
['block6a_activation[0][0]']
 AveragePooling2D)


 block6a_se_reshape (Reshap  (None, 1, 1, 672)            0
['block6a_se_squeeze[0][0]']
 e)


 block6a_se_reduce (Conv2D)  (None, 1, 1, 28)             18844
['block6a_se_reshape[0][0]']


 block6a_se_expand (Conv2D)  (None, 1, 1, 672)            19488
['block6a_se_reduce[0][0]']


 block6a_se_excite (Multipl  (None, 2, 2, 672)            0
['block6a_activation[0][0]',
 y)
'block6a_se_expand[0][0]']


 block6a_project_conv (Conv  (None, 2, 2, 192)            129024
['block6a_se_excite[0][0]']
 2D)


 block6a_project_bn (BatchN  (None, 2, 2, 192)            768
['block6a_project_conv[0][0]']
 ormalization)


 block6b_expand_conv (Conv2  (None, 2, 2, 1152)           221184
['block6a_project_bn[0][0]']
 D)


 block6b_expand_bn (BatchNo  (None, 2, 2, 1152)           4608
['block6b_expand_conv[0][0]']
 rmalization)
```

```
 block6b_expand_activation   (None, 2, 2, 1152)           0
['block6b_expand_bn[0][0]']
 (Activation)


 block6b_dwconv (DepthwiseC  (None, 2, 2, 1152)           28800
['block6b_expand_activation[0]
 onv2D)
[0]']


 block6b_bn (BatchNormaliza  (None, 2, 2, 1152)           4608
['block6b_dwconv[0][0]']
 tion)


 block6b_activation (Activa  (None, 2, 2, 1152)           0
['block6b_bn[0][0]']
 tion)


 block6b_se_squeeze (Global  (None, 1152)                 0
['block6b_activation[0][0]']
 AveragePooling2D)


 block6b_se_reshape (Reshap  (None, 1, 1, 1152)           0
['block6b_se_squeeze[0][0]']
 e)


 block6b_se_reduce (Conv2D)  (None, 1, 1, 48)             55344
['block6b_se_reshape[0][0]']


 block6b_se_expand (Conv2D)  (None, 1, 1, 1152)           56448
['block6b_se_reduce[0][0]']


 block6b_se_excite (Multipl  (None, 2, 2, 1152)           0
['block6b_activation[0][0]',
 y)
'block6b_se_expand[0][0]']


 block6b_project_conv (Conv  (None, 2, 2, 192)            221184
```

```
['block6b_se_excite[0][0]']
 2D)



 block6b_project_bn (BatchN  (None, 2, 2, 192)             768
['block6b_project_conv[0][0]']
 ormalization)



 block6b_drop (FixedDropout  (None, 2, 2, 192)             0
['block6b_project_bn[0][0]']
 )



 block6b_add (Add)           (None, 2, 2, 192)             0
['block6b_drop[0][0]',

'block6a_project_bn[0][0]']


 block6c_expand_conv (Conv2  (None, 2, 2, 1152)            221184
['block6b_add[0][0]']
 D)



 block6c_expand_bn (BatchNo  (None, 2, 2, 1152)            4608
['block6c_expand_conv[0][0]']
 rmalization)



 block6c_expand_activation   (None, 2, 2, 1152)            0
['block6c_expand_bn[0][0]']
 (Activation)



 block6c_dwconv (DepthwiseC  (None, 2, 2, 1152)            28800
['block6c_expand_activation[0]
 onv2D)
[0]']


 block6c_bn (BatchNormaliza  (None, 2, 2, 1152)            4608
['block6c_dwconv[0][0]']
 tion)
```

```
 block6c_activation (Activa  (None, 2, 2, 1152)           0
['block6c_bn[0][0]']
 tion)


 block6c_se_squeeze (Global  (None, 1152)                 0
['block6c_activation[0][0]']
 AveragePooling2D)


 block6c_se_reshape (Reshap  (None, 1, 1, 1152)           0
['block6c_se_squeeze[0][0]']
 e)


 block6c_se_reduce (Conv2D)  (None, 1, 1, 48)             55344
['block6c_se_reshape[0][0]']


 block6c_se_expand (Conv2D)  (None, 1, 1, 1152)           56448
['block6c_se_reduce[0][0]']


 block6c_se_excite (Multipl  (None, 2, 2, 1152)           0
['block6c_activation[0][0]',
 y)
'block6c_se_expand[0][0]']


 block6c_project_conv (Conv  (None, 2, 2, 192)            221184
['block6c_se_excite[0][0]']
 2D)


 block6c_project_bn (BatchN  (None, 2, 2, 192)            768
['block6c_project_conv[0][0]']
 ormalization)


 block6c_drop (FixedDropout  (None, 2, 2, 192)            0
['block6c_project_bn[0][0]']
 )
```

```
 block6c_add (Add)           (None, 2, 2, 192)            0
['block6c_drop[0][0]',

'block6b_add[0][0]']


 block6d_expand_conv (Conv2  (None, 2, 2, 1152)           221184
['block6c_add[0][0]']
 D)



 block6d_expand_bn (BatchNo  (None, 2, 2, 1152)           4608
['block6d_expand_conv[0][0]']
 rmalization)



 block6d_expand_activation   (None, 2, 2, 1152)           0
['block6d_expand_bn[0][0]']
 (Activation)



 block6d_dwconv (DepthwiseC  (None, 2, 2, 1152)           28800
['block6d_expand_activation[0]
 onv2D)
[0]']


 block6d_bn (BatchNormaliza  (None, 2, 2, 1152)           4608
['block6d_dwconv[0][0]']
 tion)



 block6d_activation (Activa  (None, 2, 2, 1152)           0
['block6d_bn[0][0]']
 tion)



 block6d_se_squeeze (Global  (None, 1152)                 0
['block6d_activation[0][0]']
 AveragePooling2D)



 block6d_se_reshape (Reshap  (None, 1, 1, 1152)           0
['block6d_se_squeeze[0][0]']
```

```
 e)



 block6d_se_reduce (Conv2D)  (None, 1, 1, 48)              55344
['block6d_se_reshape[0][0]']


 block6d_se_expand (Conv2D)  (None, 1, 1, 1152)            56448
['block6d_se_reduce[0][0]']


 block6d_se_excite (Multipl  (None, 2, 2, 1152)            0
['block6d_activation[0][0]',
 y)
'block6d_se_expand[0][0]']


 block6d_project_conv (Conv  (None, 2, 2, 192)             221184
['block6d_se_excite[0][0]']
 2D)



 block6d_project_bn (BatchN  (None, 2, 2, 192)             768
['block6d_project_conv[0][0]']
 ormalization)



 block6d_drop (FixedDropout  (None, 2, 2, 192)             0
['block6d_project_bn[0][0]']
 )



 block6d_add (Add)           (None, 2, 2, 192)             0
['block6d_drop[0][0]',

'block6c_add[0][0]']


 block7a_expand_conv (Conv2  (None, 2, 2, 1152)            221184
['block6d_add[0][0]']
 D)



 block7a_expand_bn (BatchNo  (None, 2, 2, 1152)            4608
['block7a_expand_conv[0][0]']
 rmalization)
```

```
 block7a_expand_activation   (None, 2, 2, 1152)           0
['block7a_expand_bn[0][0]']
 (Activation)


 block7a_dwconv (DepthwiseC  (None, 2, 2, 1152)           10368
['block7a_expand_activation[0]
 onv2D)
[0]']


 block7a_bn (BatchNormaliza  (None, 2, 2, 1152)           4608
['block7a_dwconv[0][0]']
 tion)


 block7a_activation (Activa  (None, 2, 2, 1152)           0
['block7a_bn[0][0]']
 tion)


 block7a_se_squeeze (Global  (None, 1152)                 0
['block7a_activation[0][0]']
 AveragePooling2D)


 block7a_se_reshape (Reshap  (None, 1, 1, 1152)           0
['block7a_se_squeeze[0][0]']
 e)


 block7a_se_reduce (Conv2D)  (None, 1, 1, 48)             55344
['block7a_se_reshape[0][0]']


 block7a_se_expand (Conv2D)  (None, 1, 1, 1152)           56448
['block7a_se_reduce[0][0]']


 block7a_se_excite (Multipl  (None, 2, 2, 1152)           0
['block7a_activation[0][0]',
 y)
'block7a_se_expand[0][0]']
```

```
 block7a_project_conv (Conv  (None, 2, 2, 320)            368640
['block7a_se_excite[0][0]']
 2D)


 block7a_project_bn (BatchN  (None, 2, 2, 320)            1280
['block7a_project_conv[0][0]']
 ormalization)


 top_conv (Conv2D)           (None, 2, 2, 1280)           409600
['block7a_project_bn[0][0]']


 top_bn (BatchNormalization  (None, 2, 2, 1280)           5120
['top_conv[0][0]']
 )


 top_activation (Activation  (None, 2, 2, 1280)           0
['top_bn[0][0]']
 )


 flatten (Flatten)           (None, 5120)                 0
['top_activation[0][0]']


 dense (Dense)               (None, 1024)                 5243904
['flatten[0][0]']


 dropout (Dropout)           (None, 1024)                 0
['dense[0][0]']


 dense_1 (Dense)             (None, 201)                  206025
['dropout[0][0]']


==================================================================
============================
Total params: 9499493 (36.24 MB)
Trainable params: 6231369 (23.77 MB)
Non-trainable params: 3268124 (12.47 MB)
```

```
_________________________________________________________________
```

Phase 2 end

Phase 3 : Training phase

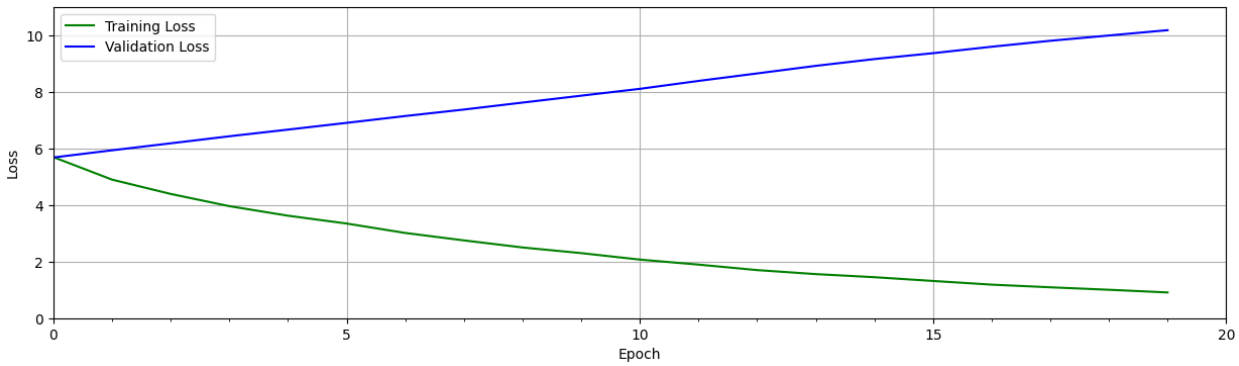
```
epoch = 20
eff_history = model.fit(x_train,
                        y_train,
                        batch_size=256,
                        epochs=epoch,
                        verbose=1,
                        validation_split=.25)

model.save('efficientnet_WB')
```

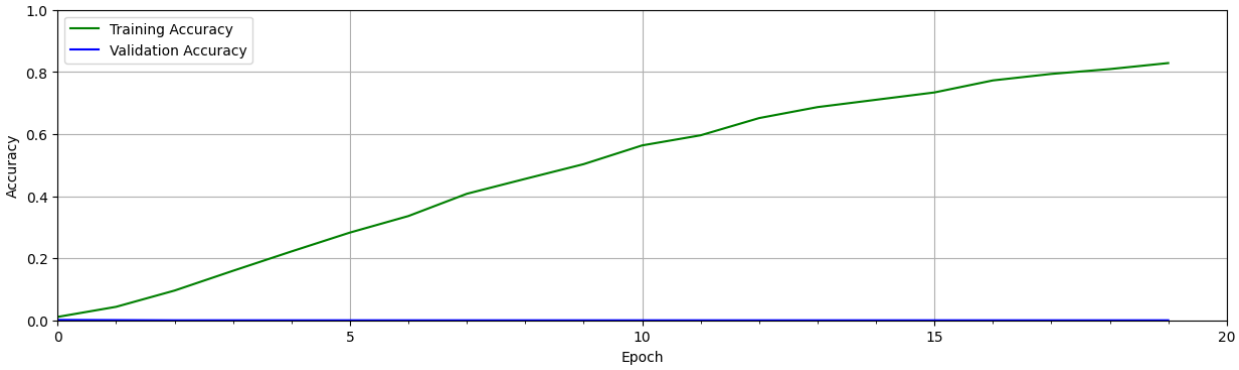
```
Epoch 1/20
18/18 [==============================] - 36s 1s/step - loss: 5.6896 -
acc: 0.0109 - val_loss: 5.6771 - val_acc: 0.0013
Epoch 2/20
18/18 [==============================] - 17s 953ms/step - loss: 4.8947
- acc: 0.0434 - val_loss: 5.9307 - val_acc: 6.6800e-04
Epoch 3/20
18/18 [==============================] - 17s 959ms/step - loss: 4.3890
- acc: 0.0957 - val_loss: 6.1776 - val_acc: 0.0000e+00
Epoch 4/20
18/18 [==============================] - 20s 1s/step - loss: 3.9569 -
acc: 0.1594 - val_loss: 6.4262 - val_acc: 0.0000e+00
Epoch 5/20
18/18 [==============================] - 20s 1s/step - loss: 3.6198 -
acc: 0.2218 - val_loss: 6.6618 - val_acc: 0.0000e+00
Epoch 6/20
18/18 [==============================] - 20s 1s/step - loss: 3.3431 -
acc: 0.2826 - val_loss: 6.9001 - val_acc: 0.0000e+00
Epoch 7/20
18/18 [==============================] - 18s 1s/step - loss: 3.0101 -
acc: 0.3358 - val_loss: 7.1416 - val_acc: 0.0000e+00
Epoch 8/20
18/18 [==============================] - 20s 1s/step - loss: 2.7473 -
acc: 0.4075 - val_loss: 7.3710 - val_acc: 0.0000e+00
Epoch 9/20
18/18 [==============================] - 19s 1s/step - loss: 2.4974 -
acc: 0.4556 - val_loss: 7.6165 - val_acc: 0.0000e+00
Epoch 10/20
18/18 [==============================] - 19s 1s/step - loss: 2.2977 -
acc: 0.5030 - val_loss: 7.8601 - val_acc: 0.0000e+00
Epoch 11/20
18/18 [==============================] - 18s 1s/step - loss: 2.0691 -
```

```
acc: 0.5633 - val_loss: 8.1022 - val_acc: 0.0000e+00
Epoch 12/20
18/18 [==============================] - 20s 1s/step - loss: 1.8921 -
acc: 0.5959 - val_loss: 8.3819 - val_acc: 0.0000e+00
Epoch 13/20
18/18 [==============================] - 19s 1s/step - loss: 1.6979 -
acc: 0.6513 - val_loss: 8.6472 - val_acc: 0.0000e+00
Epoch 14/20
18/18 [==============================] - 18s 1s/step - loss: 1.5563 -
acc: 0.6867 - val_loss: 8.9147 - val_acc: 0.0000e+00
Epoch 15/20
18/18 [==============================] - 18s 1s/step - loss: 1.4469 -
acc: 0.7103 - val_loss: 9.1541 - val_acc: 0.0000e+00
Epoch 16/20
18/18 [==============================] - 19s 1s/step - loss: 1.3142 -
acc: 0.7339 - val_loss: 9.3621 - val_acc: 0.0000e+00
Epoch 17/20
18/18 [==============================] - 18s 998ms/step - loss: 1.1841
- acc: 0.7727 - val_loss: 9.5904 - val_acc: 0.0000e+00
Epoch 18/20
18/18 [==============================] - 18s 1s/step - loss: 1.0902 -
acc: 0.7936 - val_loss: 9.8006 - val_acc: 0.0000e+00
Epoch 19/20
18/18 [==============================] - 18s 1s/step - loss: 1.0026 -
acc: 0.8092 - val_loss: 9.9872 - val_acc: 0.0000e+00
Epoch 20/20
18/18 [==============================] - 18s 1s/step - loss: 0.9080 -
acc: 0.8285 - val_loss: 10.1751 - val_acc: 0.0000e+00
INFO:tensorflow:Assets written to: efficientnet_WB\assets

INFO:tensorflow:Assets written to: efficientnet_WB\assets
```

```
def plot_results(metrics, title=None, ylabel=None, ylim=None,
metric_name=None, color=None):

    fig, ax = plt.subplots(figsize=(15, 4))

    if not (isinstance(metric_name, list) or isinstance(metric_name,
tuple)):
        metrics = [metrics,]
        metric_name = [metric_name,]

    for idx, metric in enumerate(metrics):
        ax.plot(metric, color=color[idx])

    plt.xlabel("Epoch")
    plt.ylabel(ylabel)
    plt.title(title)
    plt.xlim([0, epoch])
    plt.ylim(ylim)
```

```
    # Tailor x-axis tick marks
    ax.xaxis.set_major_locator(MultipleLocator(5))
    ax.xaxis.set_major_formatter(FormatStrFormatter('%d'))
    ax.xaxis.set_minor_locator(MultipleLocator(1))
    plt.grid(True)
    plt.legend(metric_name)
    plt.show()
    plt.close()

#Retrieve training results.
train_loss = eff_history.history["loss"]
train_acc  = eff_history.history["acc"]
valid_loss = eff_history.history["val_loss"]
valid_acc  = eff_history.history["val_acc"]

plot_results([ train_loss, valid_loss ],
            ylabel="Loss",
            ylim = [0.0, 11],
            metric_name=["Training Loss", "Validation Loss"],
            color=["g", "b"],
            );

plot_results([ train_acc, valid_acc ],
            ylabel="Accuracy",
            ylim = [0.0, 1.0],
            metric_name=["Training Accuracy", "Validation Accuracy"],
            color=["g", "b"])
```





Phase 4 : Testing

```
reloaded_model = models.load_model('efficientnet_WB')
test_loss, test_acc = reloaded_model.evaluate(x_test, y_test)
print(f"Test accuracy : {test_acc*100:.3f}")
```

```
WARNING:tensorflow:From C:\Users\HP\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\keras\src\saving\legacy\saved_model\
load.py:107: The name tf.gfile.Exists is deprecated. Please use
tf.io.gfile.exists instead.


WARNING:tensorflow:From C:\Users\HP\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\keras\src\saving\legacy\saved_model\
load.py:107: The name tf.gfile.Exists is deprecated. Please use
tf.io.gfile.exists instead.


181/181 [==============================] - 29s 146ms/step - loss:
5.5903 - acc: 0.1012
Test accuracy : 10.117
```

We found that the test accuracy is very low.

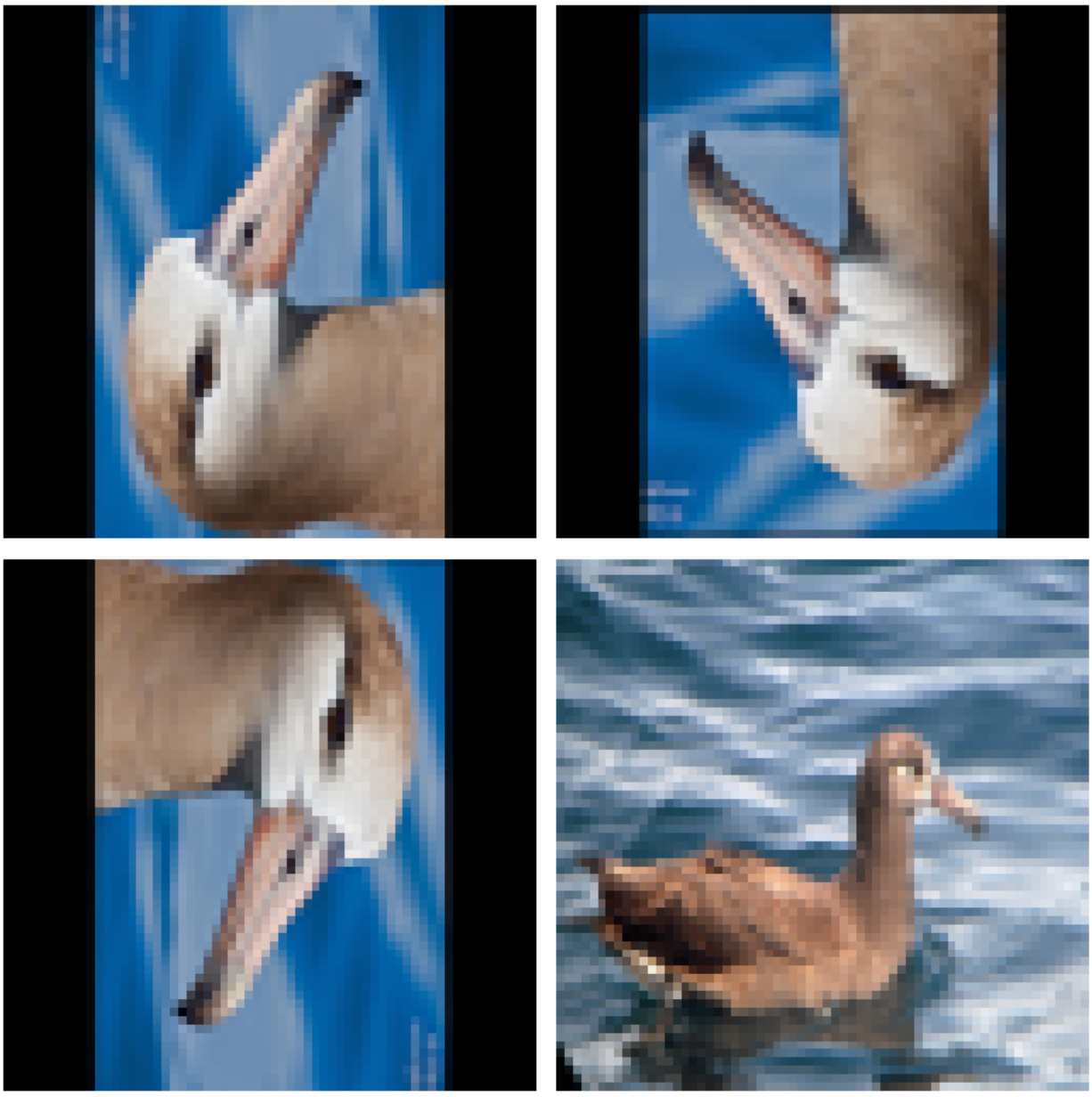
Trying out methods like data augmentation

Phase 5 : Data augmentation

```
# Creating x train and x test datasets
#This are augmented datasets

k =3
with open("CUB_200_2011/images.txt", 'r') as file:
    lines = file.readlines()

data = []
# creating dataset of file locations of images
for line in lines:
```

```
    parts = line.split()
    num = int(parts[0])
    link = ' '.join(parts[1:])
    link= ".//CUB_200_2011//images//"+link
    data.append([num, link])

data_array = np.array(data)

from PIL import Image
import numpy as np



# Creating two list ie x_train and x_test containing array of image
matrix
x_train = []
x_test = []

for i, item in enumerate(data_array, start=1):
    link = item[1]
    try:
        image = Image.open(link)
        resized_image = image.resize((t, t))  # Resize the image to
txt pixels
        resized_image_array = np.array(resized_image)   # Convert the
resized image to a NumPy array
        if i in train_index:
            x_train.append(resized_image_array)
            for j in range(k):
                image = image.rotate(90)
                resized_image = image.resize((t, t))  # Resize the
image to txt pixels
                resized_image_array = np.array(resized_image)   #
Convert the resized image to a NumPy array
                x_train.append(resized_image_array)
        else:
            x_test.append(resized_image_array)
    except Exception as e:
        print(f"Error loading image from {link}: {e}")

# Now x_train contains the resized images loaded from the links
corresponding to indices present in train_index
# and x_test contains the resized images loaded from the links
corresponding to indices not present in train_index

import matplotlib.image as mpimg
images = []
for i in range(1, 5):
    img = x_train[i]
    images.append(img)
```

```
# Create a subplot grid
fig, axes = plt.subplots(2, 2, figsize=(8, 8))

# Loop through images and plot each one
for i, img in enumerate(images):
    axes[i//2, i%2].imshow(img)
    axes[i//2, i%2].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```



Images obtained by rotating original image by 90, 180 and 270 degrees

Phase 2 : Data Preprocessing for rotated images

```
with open("CUB_200_2011/image_class_labels.txt", "r") as file:
    data = file.readlines()

# Convert data to list of floats
data = [list(map(int, line.strip().split())) for line in data]

# Convert list to tensor
class_labels_num = torch.tensor(data)

y_train = []
y_test = []
for i in range(0, N):
    if i in train_index:
        for j in range(k+1):
            y_train.append(int(class_labels_num[i][1]))
    else:
        y_test.append(int(class_labels_num[i][1]))

# TO drop images whoes dimensions are not consistant with requirnments
of model
deleted_test_sample_index = [] #this index is with respect test sample

deleted_train_sample_index = [] #This index is with respect to train
sample

# finding index of samples having no RGB dimensions (Number of such
images are very less. so deleting them seems to be a good option)
for i in range(len(x_test)):
    if (x_test[i].shape)!=(t,t,3):
        deleted_test_sample_index.append(i)
        print(f'{i}th element of x test is found inhomogenous')
for j in range(len(x_train)):
    if (x_train[j].shape)!=(t,t,3):
        deleted_train_sample_index.append(j)
        print(f'{j}th element of x train is found inhomogenous')

print(deleted_test_sample_index)
print(deleted_train_sample_index)

# Droping thoes datapoints from dataset
for i in range(len(deleted_test_sample_index)):
    x_test.pop(deleted_test_sample_index[i]-i)
    y_test.pop(deleted_test_sample_index[i]-i)
    print(deleted_test_sample_index[i])

for i in range(len(deleted_train_sample_index)):
    x_train.pop(deleted_train_sample_index[i]-i)
```

```
    y_train.pop(deleted_train_sample_index[i]-i)
    print(deleted_train_sample_index[i])
```

```
195th element of x test is found inhomogenous
3081th element of x test is found inhomogenous
2996th element of x train is found inhomogenous
2997th element of x train is found inhomogenous
2998th element of x train is found inhomogenous
2999th element of x train is found inhomogenous
7472th element of x train is found inhomogenous
7473th element of x train is found inhomogenous
7474th element of x train is found inhomogenous
7475th element of x train is found inhomogenous
7476th element of x train is found inhomogenous
7477th element of x train is found inhomogenous
7478th element of x train is found inhomogenous
7479th element of x train is found inhomogenous
7804th element of x train is found inhomogenous
7805th element of x train is found inhomogenous
7806th element of x train is found inhomogenous
7807th element of x train is found inhomogenous
10320th element of x train is found inhomogenous
10321th element of x train is found inhomogenous
10322th element of x train is found inhomogenous
10323th element of x train is found inhomogenous
11048th element of x train is found inhomogenous
11049th element of x train is found inhomogenous
11050th element of x train is found inhomogenous
11051th element of x train is found inhomogenous
[195, 3081]
[2996, 2997, 2998, 2999, 7472, 7473, 7474, 7475, 7476, 7477, 7478,
7479, 7804, 7805, 7806, 7807, 10320, 10321, 10322, 10323, 11048,
11049, 11050, 11051]
195
3081
2996
2997
2998
2999
7472
7473
7474
7475
7476
7477
7478
7479
7804
7805
7806
```

```
7807
10320
10321
10322
10323
11048
11049
11050
11051
```

```
x_train = np.array(x_train)
x_test = np.array(x_test)

# Normalize images to the range [0, 1].
x_train = x_train.astype("float32") / 255
x_test  = x_test.astype("float32") / 255

#TO convert y train and test set into categorical format
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
print(y_train)
```

```
[[0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]]
```

```
print(x_train.shape)
print(len(y_train))
print(x_test.shape)
print(len(y_test))
```

```
(23952, 64, 64, 3)
23952
(5792, 64, 64, 3)
5792
```

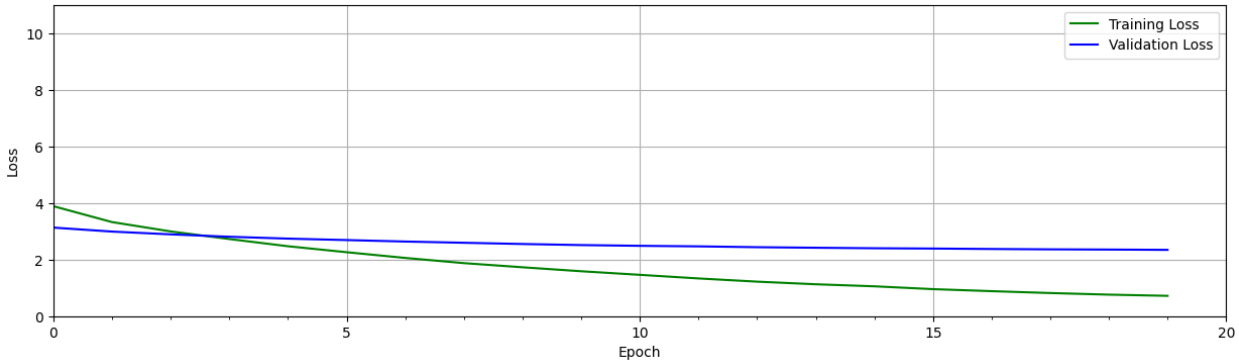
```
num_samples_train = len(x_train)
shuffled_indices_train = np.random.permutation(num_samples_train)

# Shuffle both arrays using the same shuffled indices
x_train = x_train[shuffled_indices_train]
y_train = y_train[shuffled_indices_train]

num_samples_test = len(x_test)
shuffled_indices_test = np.random.permutation(num_samples_test)

# Shuffle both arrays using the same shuffled indices
```

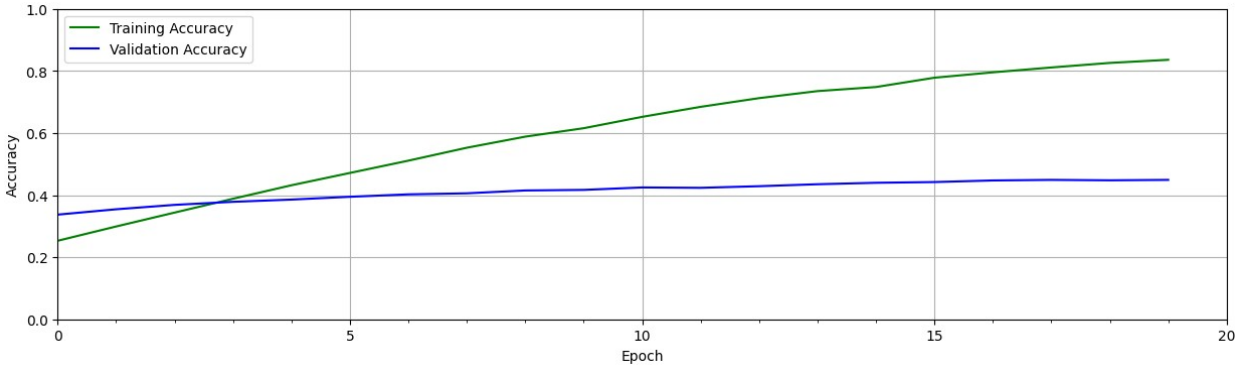
```
x_test = x_test[shuffled_indices_test]
y_test = y_test[shuffled_indices_test]

model = models.load_model('efficientnet_WB_aug')
epoch = 20
eff_history = model.fit(x_train,
                        y_train,
                        batch_size=256,
                        epochs=epoch,
                        verbose=1,
                        validation_split=.25)

model.save('efficientnet_WB_aug')
```

```
Epoch 1/20
71/71 [==============================] - 112s 1s/step - loss: 3.8848 -
acc: 0.2529 - val_loss: 3.1285 - val_acc: 0.3372
Epoch 2/20
71/71 [==============================] - 90s 1s/step - loss: 3.3235 -
acc: 0.2989 - val_loss: 2.9881 - val_acc: 0.3544
Epoch 3/20
71/71 [==============================] - 90s 1s/step - loss: 2.9934 -
acc: 0.3437 - val_loss: 2.8904 - val_acc: 0.3686
Epoch 4/20
71/71 [==============================] - 91s 1s/step - loss: 2.7244 -
acc: 0.3883 - val_loss: 2.8050 - val_acc: 0.3784
Epoch 5/20
71/71 [==============================] - 99s 1s/step - loss: 2.4702 -
acc: 0.4318 - val_loss: 2.7391 - val_acc: 0.3856
Epoch 6/20
71/71 [==============================] - 84s 1s/step - loss: 2.2592 -
acc: 0.4714 - val_loss: 2.6885 - val_acc: 0.3946
Epoch 7/20
71/71 [==============================] - 83s 1s/step - loss: 2.0571 -
acc: 0.5110 - val_loss: 2.6361 - val_acc: 0.4025
Epoch 8/20
71/71 [==============================] - 81s 1s/step - loss: 1.8732 -
acc: 0.5526 - val_loss: 2.5927 - val_acc: 0.4058
Epoch 9/20
71/71 [==============================] - 82s 1s/step - loss: 1.7267 -
acc: 0.5883 - val_loss: 2.5493 - val_acc: 0.4150
Epoch 10/20
71/71 [==============================] - 85s 1s/step - loss: 1.5863 -
acc: 0.6155 - val_loss: 2.5116 - val_acc: 0.4168
Epoch 11/20
71/71 [==============================] - 83s 1s/step - loss: 1.4601 -
acc: 0.6521 - val_loss: 2.4849 - val_acc: 0.4247
Epoch 12/20
71/71 [==============================] - 85s 1s/step - loss: 1.3324 -
acc: 0.6843 - val_loss: 2.4657 - val_acc: 0.4235
```

```
Epoch 13/20
71/71 [==============================] - 87s 1s/step - loss: 1.2190 -
acc: 0.7124 - val_loss: 2.4364 - val_acc: 0.4287
Epoch 14/20
71/71 [==============================] - 89s 1s/step - loss: 1.1266 -
acc: 0.7350 - val_loss: 2.4168 - val_acc: 0.4350
Epoch 15/20
71/71 [==============================] - 83s 1s/step - loss: 1.0543 -
acc: 0.7482 - val_loss: 2.3990 - val_acc: 0.4397
Epoch 16/20
71/71 [==============================] - 89s 1s/step - loss: 0.9544 -
acc: 0.7782 - val_loss: 2.3883 - val_acc: 0.4421
Epoch 17/20
71/71 [==============================] - 92s 1s/step - loss: 0.8831 -
acc: 0.7955 - val_loss: 2.3730 - val_acc: 0.4472
Epoch 18/20
71/71 [==============================] - 87s 1s/step - loss: 0.8164 -
acc: 0.8111 - val_loss: 2.3606 - val_acc: 0.4491
Epoch 19/20
71/71 [==============================] - 80s 1s/step - loss: 0.7599 -
acc: 0.8258 - val_loss: 2.3529 - val_acc: 0.4477
Epoch 20/20
71/71 [==============================] - 91s 1s/step - loss: 0.7178 -
acc: 0.8358 - val_loss: 2.3420 - val_acc: 0.4491
INFO:tensorflow:Assets written to: efficientnet_WB_aug\assets

INFO:tensorflow:Assets written to: efficientnet_WB_aug\assets
```

```
#Retrieve training results.
train_loss = eff_history.history["loss"]
train_acc  = eff_history.history["acc"]
valid_loss = eff_history.history["val_loss"]
valid_acc  = eff_history.history["val_acc"]

plot_results([ train_loss, valid_loss ],
            ylabel="Loss",
            ylim = [0.0, 11],
            metric_name=["Training Loss", "Validation Loss"],
            color=["g", "b"],
            );

plot_results([ train_acc, valid_acc ],
            ylabel="Accuracy",
            ylim = [0.0, 1.0],
            metric_name=["Training Accuracy", "Validation Accuracy"],
            color=["g", "b"])
```






```
reloaded_model = models.load_model('efficientnet_WB_aug')
test_loss, test_acc = reloaded_model.evaluate(x_test, y_test)
print(f"Test accuracy : {test_acc*100:.3f}")

181/181 [==============================] - 39s 197ms/step - loss:
4.8689 - acc: 0.1165
Test accuracy : 11.654
```

we are not able to increase accuracy beyond 12% even by augmentation and random shuffling

Model developement

We tried using efficientnetB0 as pretrianed model. On top of it we added 1 flat layer and two dense layer to get final output. To avoid overfitting we tried to use dropout data augmentation and random shuffling. We used softmax at the end to get probability profile. We tried both RMSprop and Adam activation. We found that Adam is performing better in training as well as testing. For finetuing we unfreezed last few layers of efficientnet. First training happened with 20 epoch with 20s for each epoch. Total training time found to 40s on my device.(Training time can varry device to device)

In conclusion, We firstly tried to build our own CNN model test accuracy of which was around 4%.

After that, we tried to explore few pretrained models like VGG, inceptionnet and efficientnet.

Accuracy of inceptionnet was found to be around 22% but number of parameters were around 24M. Hence, we rejected that model.

Our final submission includes trained model using efficient net having 9.1M parameters with test accuracy 11.6%