Recognize ingredients using Transfer Learning

```
from __future__ import absolute_import, division, print_function, unicode_litera
# from google_images_download import google_images_download

try:
  # The %tensorflow_version magic only works in colab.
  %tensorflow_version 2.x
except Exception:
  pass
import tensorflow as tf

import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

```
tf.__version__
```

```
'2.1.0'
```

```
import os
from tqdm import tqdm
from tensorflow.keras import models, layers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization, Activation, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import Dense,AveragePooling2D,BatchNormalization,Co
from keras.preprocessing.image import ImageDataGenerator
from time import time
from datetime import datetime
from tensorflow.python.keras.callbacks import TensorBoard
```

```
Using TensorFlow backend.
```

Setup Input Pipeline

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_

Enter your authorization code:
··········
Mounted at /content/drive
```

```
!pwd
```

```
/content
```

```
path = 'drive/My Drive/Project ideas/Recipe_Finder/'

```

```
os.chdir(path)
```

```
base_dir = os.path.join('/content/drive/My Drive/Project ideas/Recipe_Finder/',
```

```
base_dir
```

```
'/content/drive/My Drive/Project ideas/Recipe_Finder/downloads'
```

```
os.chdir(base_dir)
!ls
```

```
 almonds           cucumber           lemon          prawn
 asparagus         dates              lettuce        pumpkin
 bananas           egg                mango          raisins
'bay leaf'         eggplant           mint          'raw banana'
'bell pepper'     'fenugreek leaves' 'mint leaf'    'raw mango'
 broccoli         'fenugreek seed'    mushrooms     'raw masoor dal'
 carrot            fish               mussels       'raw moong dal'
'cashew nut'      'fruit apple'       okra           rice
 cauliflower       garlic             onion          rosemary
'cayenne pepper'   ginger             orange         spinach
 cherry            gourd              paneer         strawberries
 chicken          'granny smith'      papaya         tamarind
 chickpeas         grape              peanut         tomato
 chilli            guava              pear           turnip
 cinnamon          hazelnut           peas           vanilla
 coconut           jackfruit          pineapple     'vegetable drumstick'
'coriander leaf'  'kidney bean'       pistachio      watermelon
 corn              kiwis              pomegranate    yam
 cranberries       labels.txt         potatoes
```

```

```

```

```

```

```

Method-1

Train and test Image set preparation

Use `ImageDataGenerator` to rescale the images.

Create the train generator and specify where the train dataset directory, image size, batch size.

Create the validation generator with similar approach as the train generator with the flow_from_dire

```
IMAGE_SIZE = 224
BATCH_SIZE = 64

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2)

# rotation_range=40,
#     width_shift_range=0.2,
#     height_shift_range=0.2,
#     shear_range=0.2,
#     zoom_range=0.2,
#     horizontal_flip=True,
#     fill_mode='nearest',


train_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='training')

val_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='validation')
```

```
Found 3202 images belonging to 74 classes.
Found 769 images belonging to 74 classes.
```

```
datagen
```

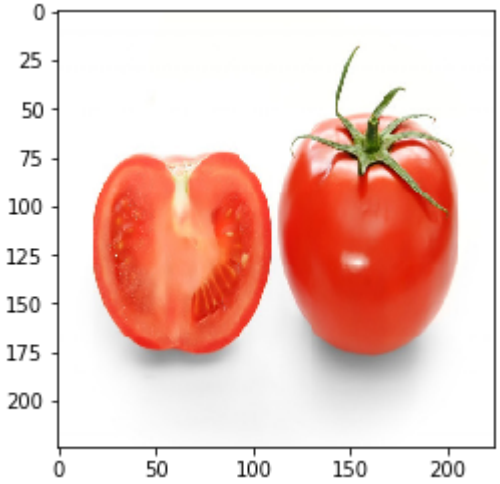
```
<tensorflow.python.keras.preprocessing.image.ImageDataGenerator at 0x7fc4887ae
```

```
# len_len()
```

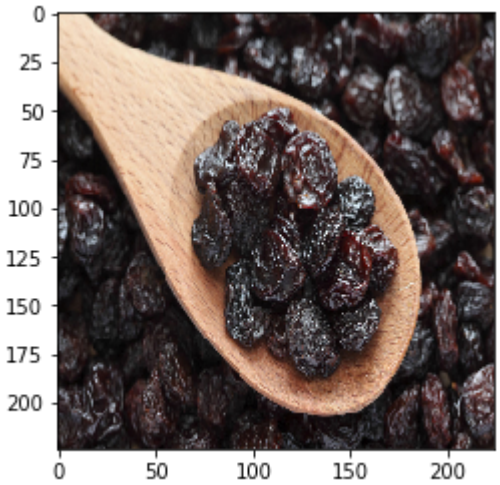
```
x,y = train_generator.next()
for i in range(0,3):
    image = x[i]
    label = y[i]
    print (label)
    plt.imshow(image)
    plt.show()

```

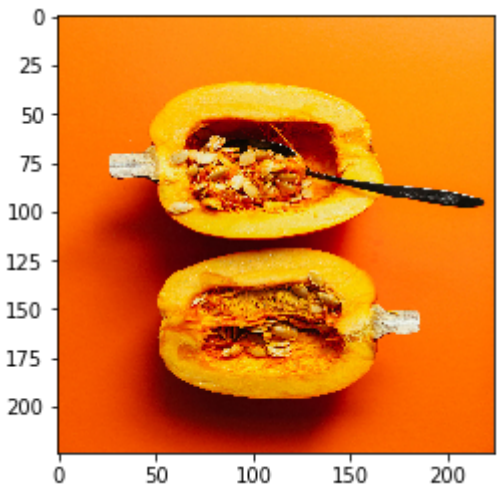
```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 0. 0.]
```




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




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




```
1 sbase_dir
```

```
'/content/drive/My Drive/Project ideas/Recipe_Finder/downloads'
```

```
for image_batch, label_batch in train_generator:
  break
image_batch.shape, label_batch.shape
```

```
/usr/local/lib/python3.6/dist-packages/PIL/Image.py:989: UserWarning: Palette i
  "Palette images with Transparency expressed in bytes should be "
((64, 224, 224, 3), (64, 74))
```

Save the labels in a file which will be downloaded later.

```
print (train_generator.class_indices)

labels = '\n'.join(sorted(train_generator.class_indices.keys()))

with open('labels.txt', 'w') as f:
  f.write(labels)
```

```
{'almonds': 0, 'asparagus': 1, 'bananas': 2, 'bay leaf': 3, 'bell pepper': 4,
```

```
print(len(train_generator.class_indices))
```

```
74
```

```
!cat labels.txt
```

```
almonds
asparagus
bananas
bay leaf
bell pepper
broccoli
carrot
cashew nut
cauliflower
cayenne pepper
cherry
chicken
chickpeas
chilli
cinnamon
coconut
coriander leaf
corn
cranberries
cucumber
dates
egg
eggplant
fenugreek leaves
fenugreek seed
fish
fruit apple
garlic
ginger
gourd
granny smith
grape
guava
hazelnut
jackfruit
kidney bean
kiwis
lemon
lettuce
mango
mint
mint leaf
mushrooms
mussels
okra
onion
orange
paneer
papaya
peanut
pear
peas
pineapple
pistachio
pomegranate
potatoes
prawn
pumpkin
raisins
raw banana
raw mango
```

```
raw masoor dal
raw moong dal
rice
rosemary
spinach
strawberries
tamarind
tomato
turnip
vanilla
vegetable drumstick
watermelon
yam
```

```
input_shape_img = (image_batch.shape[1],image_batch.shape[2],image_batch.shape[3
print(input_shape_img)
```

```
(224, 224, 3)
```

Create the base model from the pre-trained convnets

Create the base model from the **MobileNet V2** model developed at Google, and pre-trained on the I images and 1000 classes of web images.

First, pick which intermediate layer of MobileNet V2 will be used for feature extraction. A common layer before the flatten operation, the so-called "bottleneck layer". The reasoning here is that the fo specialized to the task the network was trained on, and thus the features learned by these layers w bottleneck features, however, retain much generality.

Let's instantiate an MobileNet V2 model pre-loaded with weights trained on ImageNet. By specifyir load a network that doesn't include the classification layers at the top, which is ideal for feature ext

```
IMG_SHAPE = (IMAGE_SIZE, IMAGE_SIZE, 3)


# Create the base model from the pre-trained model MobileNet V2
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

```
base_model.trainable = True

# Let's take a look to see how many layers are in the base model
print("Number of layers in the base model: ", len(base_model.layers))

# Fine tune from this layer onwards
fine_tune_at = 100 #20 # 100 thila

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
  layer.trainable =  False
```

```
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D
from keras.layers import Activation, Dense, Flatten, Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint
from keras import backend as K
```

```
from keras.regularizers import l2,l1
```

```
base_model.trainable = False


model = tf.keras.Sequential([
  base_model,
  tf.keras.layers.Conv2D(filters = 128, kernel_size = (3,3), activation= 'relu')
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Conv2D(filters = 256, kernel_size = (3,3), activation= 'relu')
#   tf.keras.layers.Dropout(0.5),
#   tf.keras.layers.Conv2D(filters = 512, kernel_size = (3,3), activation= 'relu
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(150, activation= 'relu'),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.GlobalAveragePooling2D(),
  tf.keras.layers.Dense(len(train_generator.class_indices), activation='softmax'
])


```

```
model.summary()
```

```

# sgd = tf.keras.optimizers.SGD(lr = 0.1,momentum = 0.7, nesterov = True)
rmsprop = tf.keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9)
adam = tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, a
adam_0 = tf.keras.optimizers.Adam(1e-5)

model.compile(loss='categorical_crossentropy',
              optimizer=rmsprop, #adam_0,#adam #rmsprop
              metrics=['accuracy'])
# print('Compiled!')
```

```
# model.summary()

# Call Backs
filepath = "weights.{epoch:02d}-{val_loss:.2f}.hdf5"
history = tf.keras.callbacks.History()
tensorboard = TensorBoard(log_dir="logs/{}".format(time()))
filepath = "weights.{epoch:02d}-{val_accuracy:.2f}.hdf5"
path = os.path.abspath('__model_save/')
```

```

filepath = os.path.join(path, filepath)
learning_rate_reduction = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accu
checkpoint_save = tf.keras.callbacks.ModelCheckpoint(filepath, monitor='val_accu
early_stoping = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patienc
callbacks_list = [checkpoint_save,history,tensorboard,learning_rate_reduction,ea

```

```
filepath
```

```
os.getcwd()
```

```
epochs = 300

model.load_weights(os.path.join(path, 'weights.28-0.65.hdf5')) # weights.52-1.54

history = model.fit_generator(train_generator, epochs=epochs, callbacks = callba

```

```
  #Loss plot
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1.2])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,3.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```

```

```

Convert to TFLite

↳ 10 cells hidden

CNN 4 layer Own Model

↳ 6 cells hidden

Inception Model

↳ 6 cells hidden

VGG16 Model

```
IMG_SHAPE = (IMAGE_SIZE, IMAGE_SIZE, 3)

vgg16_base_model = tf.keras.applications.VGG16(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')

```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases
58892288/58889256 [==============================] - 1s 0us/step
```

```

for layer in vgg16_base_model.layers[:]: # add [:50] for next time
    layer.trainable = False
```

```


vgg16_model = tf.keras.Sequential([
  vgg16_base_model,
  tf.keras.layers.Conv2D(filters = 128, kernel_size = 2, activation= 'relu'),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Conv2D(filters = 256, kernel_size = 2, activation= 'relu'),
#   tf.keras.layers.Dropout(0.5),
#   tf.keras.layers.Conv2D(filters = 512, kernel_size = (3,3), activation= 'relu
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(150, activation= 'relu'),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.GlobalAveragePooling2D(),
  tf.keras.layers.Dense(len(train_generator.class_indices), activation='softmax'
])


vgg16_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
conv2d (Conv2D)              (None, 6, 6, 128)         262272
_________________________________________________________________
dropout (Dropout)            (None, 6, 6, 128)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 5, 5, 256)         131328
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 5, 256)         0
_________________________________________________________________
dense (Dense)                (None, 5, 5, 150)         38550
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 150)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 150)               0
_________________________________________________________________
dense_1 (Dense)              (None, 74)                11174
=================================================================
Total params: 15,158,012
Trainable params: 443,324
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
# filepath = "weights.{epoch:02d}-{val_loss:.2f}.hdf5"
history = tf.keras.callbacks.History()
tensorboard = TensorBoard(log_dir="logs/{}".format(time()))
filepath = "vgg_weights-epoch:{epoch:02d}-train_acc:_{accuracy:.2f}-test_acc:_{v
path = os.path.abspath('__model_save/')
filepath = os.path.join(path, filepath)
learning_rate_reduction = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accu
checkpoint_save = tf.keras.callbacks.ModelCheckpoint(filepath, monitor='val_accu
early_stoping = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patienc
callbacks_list = [checkpoint_save,history,tensorboard,learning_rate_reduction,ea

```

```
rmsprop = tf.keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9)
adam = tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, a
adam_0 = tf.keras.optimizers.Adam(1e-5)

vgg16_model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=['accuracy'])
print('Compiled!')
```

```
Compiled!
```

```
                                          #vgg_weights-epoch_31-train_acc__0.9
vgg16_model.load_weights(os.path.join(path, 'vgg_weights-epoch_31-train_acc__0.9

history = vgg16_model.fit_generator(train_generator, epochs=100, callbacks = cal
```

```
WARNING:tensorflow:From <ipython-input-28-b5ab640f1648>:3: Model.fit_generator
Instructions for updating:
Please use Model.fit, which supports generators.
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 51 steps, validate for 13 steps
Epoch 1/100
 1/51 [..............................] - ETA: 23:47 - loss: 0.0456 - accuracy:
  "Palette images with Transparency expressed in bytes should be "
15/51 [=======>......................] - ETA: 11:16 - loss: 0.2105 - accuracy:
  warnings.warn(str(msg))
30/51 [================>.............] - ETA: 6:33 - loss: 0.2107 - accuracy: 
  warnings.warn(str(msg))
50/51 [============================>.] - ETA: 18s - loss: 0.1928 - accuracy: 0
Epoch 00001: val_accuracy improved from -inf to 0.72172, saving model to /conte
51/51 [==============================] - 1158s 23s/step - loss: 0.1928 - accura
Epoch 2/100
50/51 [============================>.] - ETA: 1s - loss: 0.1031 - accuracy: 0.9
Epoch 00002: val_accuracy improved from 0.72172 to 0.72432, saving model to /co
51/51 [==============================] - 107s 2s/step - loss: 0.1049 - accuracy
Epoch 3/100
50/51 [============================>.] - ETA: 1s - loss: 0.1204 - accuracy: 0.9
Epoch 00003: val_accuracy did not improve from 0.72432
51/51 [==============================] - 106s 2s/step - loss: 0.1211 - accuracy
Epoch 4/100
50/51 [============================>.] - ETA: 1s - loss: 0.1248 - accuracy: 0.9
Epoch 00004: val_accuracy did not improve from 0.72432
51/51 [==============================] - 107s 2s/step - loss: 0.1258 - accuracy
Epoch 5/100
50/51 [============================>.] - ETA: 1s - loss: 0.1262 - accuracy: 0.9
Epoch 00005: val_accuracy did not improve from 0.72432
51/51 [==============================] - 108s 2s/step - loss: 0.1283 - accuracy
Epoch 6/100
50/51 [============================>.] - ETA: 1s - loss: 0.1188 - accuracy: 0.9
Epoch 00006: val_accuracy did not improve from 0.72432
51/51 [==============================] - 108s 2s/step - loss: 0.1201 - accuracy
Epoch 7/100
50/51 [============================>.] - ETA: 1s - loss: 0.1675 - accuracy: 0.9
Epoch 00007: val_accuracy did not improve from 0.72432

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257
51/51 [==============================] - 108s 2s/step - loss: 0.1660 - accuracy
Epoch 8/100
50/51 [============================>.] - ETA: 1s - loss: 0.0721 - accuracy: 0.9
Epoch 00008: val_accuracy improved from 0.72432 to 0.75423, saving model to /co
51/51 [==============================] - 108s 2s/step - loss: 0.0713 - accuracy
Epoch 9/100
50/51 [============================>.] - ETA: 1s - loss: 0.0398 - accuracy: 0.9
Epoch 00009: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0393 - accuracy
Epoch 10/100
50/51 [============================>.] - ETA: 1s - loss: 0.0395 - accuracy: 0.9
Epoch 00010: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0397 - accuracy
Epoch 11/100
```

```
50/51 [============================>.] - ETA: 1s - loss: 0.0366 - accuracy: 0.9
Epoch 00011: val_accuracy did not improve from 0.75423
51/51 [==============================] - 108s 2s/step - loss: 0.0363 - accuracy
Epoch 12/100
50/51 [============================>.] - ETA: 1s - loss: 0.0374 - accuracy: 0.9
Epoch 00012: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0374 - accuracy
Epoch 13/100
50/51 [============================>.] - ETA: 1s - loss: 0.0431 - accuracy: 0.9
Epoch 00013: val_accuracy did not improve from 0.75423

Epoch 00013: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628
51/51 [==============================] - 109s 2s/step - loss: 0.0426 - accuracy
Epoch 14/100
50/51 [============================>.] - ETA: 1s - loss: 0.0261 - accuracy: 0.9
Epoch 00014: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0260 - accuracy
Epoch 15/100
50/51 [============================>.] - ETA: 1s - loss: 0.0224 - accuracy: 0.9
Epoch 00015: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0230 - accuracy
Epoch 16/100
50/51 [============================>.] - ETA: 1s - loss: 0.0251 - accuracy: 0.9
Epoch 00016: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0248 - accuracy
Epoch 17/100
50/51 [============================>.] - ETA: 1s - loss: 0.0236 - accuracy: 0.9
Epoch 00017: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0234 - accuracy
Epoch 18/100
50/51 [============================>.] - ETA: 1s - loss: 0.0179 - accuracy: 0.9
Epoch 00018: val_accuracy did not improve from 0.75423

Epoch 00018: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814
51/51 [==============================] - 108s 2s/step - loss: 0.0179 - accuracy
Epoch 19/100
49/51 [============================>..] - ETA: 3s - loss: 0.0170 - accuracy: 0.9
Epoch 00019: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0166 - accuracy
Epoch 20/100
50/51 [============================>.] - ETA: 1s - loss: 0.0149 - accuracy: 0.9
Epoch 00020: val_accuracy did not improve from 0.75423
51/51 [==============================] - 108s 2s/step - loss: 0.0166 - accuracy
Epoch 21/100
50/51 [============================>.] - ETA: 1s - loss: 0.0151 - accuracy: 0.9
Epoch 00021: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0150 - accuracy
Epoch 22/100
49/51 [============================>..] - ETA: 3s - loss: 0.0154 - accuracy: 0.9
Epoch 00022: val_accuracy did not improve from 0.75423
51/51 [==============================] - 111s 2s/step - loss: 0.0150 - accuracy
Epoch 23/100
50/51 [============================>.] - ETA: 1s - loss: 0.0128 - accuracy: 0.9
Epoch 00023: val_accuracy did not improve from 0.75423

Epoch 00023: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
51/51 [==============================] - 111s 2s/step - loss: 0.0127 - accuracy
Epoch 24/100
50/51 [============================>.] - ETA: 1s - loss: 0.0110 - accuracy: 0.9
Epoch 00024: val_accuracy did not improve from 0.75423
51/51 [==============================] - 110s 2s/step - loss: 0.0112 - accuracy
Epoch 25/100
```

```
50/51 [============================>.] - ETA: 1s - loss: 0.0124 - accuracy: 0.9
Epoch 00025: val_accuracy did not improve from 0.75423
51/51 [==============================] - 110s 2s/step - loss: 0.0129 - accuracy
Epoch 26/100
50/51 [============================>.] - ETA: 1s - loss: 0.0121 - accuracy: 0.9
Epoch 00026: val_accuracy did not improve from 0.75423
51/51 [==============================] - 109s 2s/step - loss: 0.0120 - accuracy
Epoch 27/100
50/51 [============================>.] - ETA: 1s - loss: 0.0140 - accuracy: 0.9
Epoch 00027: val_accuracy did not improve from 0.75423
51/51 [==============================] - 108s 2s/step - loss: 0.0138 - accuracy
Epoch 28/100
50/51 [============================>.] - ETA: 1s - loss: 0.0111 - accuracy: 0.9
Epoch 00028: val_accuracy did not improve from 0.75423

Epoch 00028: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05
51/51 [==============================] - 108s 2s/step - loss: 0.0111 - accuracy
Epoch 29/100
50/51 [============================>.] - ETA: 1s - loss: 0.0120 - accuracy: 0.9
Epoch 00029: val_accuracy did not improve from 0.75423
51/51 [==============================] - 107s 2s/step - loss: 0.0121 - accuracy
Epoch 30/100
50/51 [============================>.] - ETA: 1s - loss: 0.0108 - accuracy: 0.9
Epoch 00030: val_accuracy did not improve from 0.75423
51/51 [==============================] - 107s 2s/step - loss: 0.0106 - accuracy
Epoch 31/100
50/51 [============================>.] - ETA: 1s - loss: 0.0122 - accuracy: 0.9
Epoch 00031: val_accuracy did not improve from 0.75423
51/51 [==============================] - 106s 2s/step - loss: 0.0120 - accuracy
Epoch 32/100
50/51 [============================>.] - ETA: 1s - loss: 0.0110 - accuracy: 0.9
Epoch 00032: val_accuracy did not improve from 0.75423
51/51 [==============================] - 107s 2s/step - loss: 0.0108 - accuracy
Epoch 33/100
50/51 [============================>.] - ETA: 1s - loss: 0.0110 - accuracy: 0.9
Epoch 00033: val_accuracy did not improve from 0.75423

Epoch 00033: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05
51/51 [==============================] - 108s 2s/step - loss: 0.0109 - accuracy
```

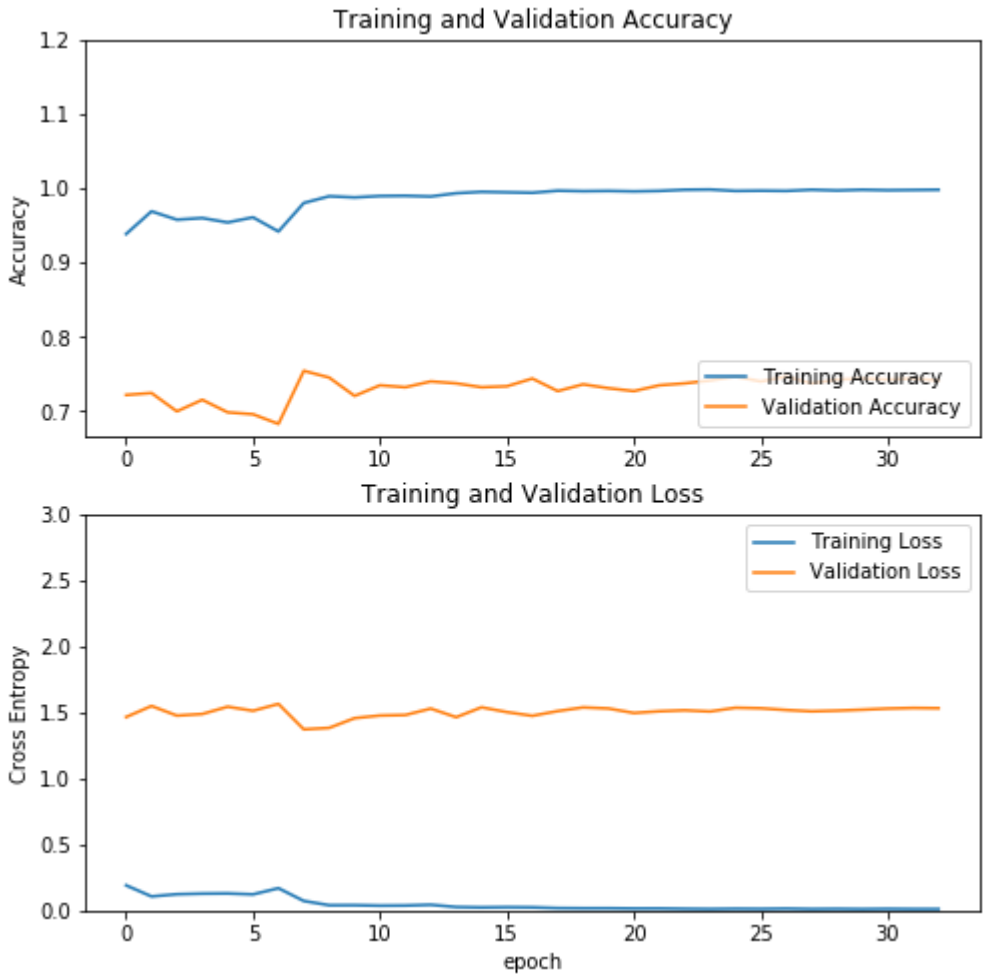
```
#Loss plot
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1.2])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
```

```
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,3.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```



Saving to tflite

```

```

```
saved_model_dir = 'save/fine_tuning'
tf.saved_model.save(vgg16_model, saved_model_dir)

converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()

with open('model_75.tflite', 'wb') as f:
  f.write(tflite_model)
```

```
WARNING:tensorflow:From /tensorflow-2.1.0/python3.6/tensorflow_core/python/ops
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
INFO:tensorflow:Assets written to: save/fine_tuning/assets
```

```
from google.colab import files

files.download('model_75.tflite')
files.download('labels.txt')
```

Observation

- Make model with 4 diffrent Transfer learning based model those are Inceptionv3, VGG16, Mo
- From all this i found out that for this specific problem set with these images vgg16 worked be

```
from prettytable import PrettyTable

```

```
x = PrettyTable()

x.field_names = ["Model Name", "Hidden Layer", "Accuracy"]

x.add_row(["Inceptionv3", 3, 60.5])
x.add_row(["VGG16", 2, 75.6])
x.add_row(["Mobilenet", 2, 55.2])
x.add_row(["Own Model", 4, 51.0])

print(x)

```

```
+-------------+--------------+----------+
|  Model Name | Hidden Layer | Accuracy |
+-------------+--------------+----------+
| Inceptionv3 |      3       |   60.5   |
|    VGG16    |      2       |   75.6   |
|  Mobilenet  |      2       |   55.2   |
|  Own Model  |      4       |   51.0   |
+-------------+--------------+----------+
```

```

```

```

```