```
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_remount=True).
```

```
import os
from glob import glob

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical

from keras.preprocessing.image import ImageDataGenerator
from keras import layers
from keras import Model
from keras.applications.inception_v3 import InceptionV3, preprocess_input
from keras.optimizers import Adam
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras import backend as K

%matplotlib inline
import matplotlib.pyplot as plt
```

```
X_train = np.load("/content/drive/MyDrive/Colab Notebooks/Copy of 256_192_train.npy")
```

```
y_train = np.load("/content/drive/MyDrive/Colab Notebooks/train_labels.npy")
```

```
X_val = np.load("/content/drive/MyDrive/Colab Notebooks/256_192_val.npy")
```

```
y_val = np.load("/content/drive/MyDrive/Colab Notebooks/val_labels.npy")
```

```
X_train.shape, X_val.shape
```

```
((4596, 192, 256, 3), (511, 192, 256, 3))
```

```
y_train.shape, y_val.shape
```

```
((4596,), (511,))
```

```
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
```

```
pre_trained_model = InceptionV3(input_shape=(192, 256, 3), include_top=False, weights="imagenet")
```

```
for layer in pre_trained_model.layers:
    print(layer.name)
    if hasattr(layer, 'moving_mean') and hasattr(layer, 'moving_variance'):
        layer.trainable = True
        K.eval(K.update(layer.moving_mean, K.zeros_like(layer.moving_mean)))
        K.eval(K.update(layer.moving_variance, K.zeros_like(layer.moving_variance)))
    else:
        layer.trainable = False

print(len(pre_trained_model.layers))
```

```
activation_79
activation_82
activation_83
batch_normalization_84
activation_76
mixed9_0
concatenate
activation_84
mixed9
conv2d_89
batch_normalization_89
activation_89
conv2d_86
conv2d_90
batch_normalization_86
batch_normalization_90
activation_86
activation_90
conv2d_87
conv2d_88
conv2d_91
conv2d_92
average_pooling2d_8
conv2d_85
batch_normalization_87
batch_normalization_88
batch_normalization_91
batch_normalization_92
conv2d_93
batch_normalization_85
activation_87
activation_88
activation_91
activation_92
batch_normalization_93
activation_85
mixed9_1
concatenate_1
activation_93
mixed10
311
```

```
last_layer = pre_trained_model.get_layer('mixed10')
print('last layer output shape:', last_layer.output_shape)
last_output = last_layer.output
```

```
last layer output shape: (None, 4, 6, 2048)
```

```
import tensorflow
# Flatten the output layer to 1 dimension
x = layers.GlobalMaxPooling2D()(last_output)
# Add a fully connected layer with 512 hidden units and ReLU activation
x = layers.Dense(512, activation='relu')(x)
# Add a dropout rate of 0.5
x = layers.Dropout(0.5)(x)
# Add a final sigmoid layer for classification
x = layers.Dense(7, activation='softmax')(x)

# Configure and compile the model

model = Model(pre_trained_model.input, x)
#optimizer = tensorflow.keras.optimizers.legacy.SGD(lr=0.0001, decay=0.0)
model.compile(loss='categorical_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

```
model.summary()
```

```
activation_92 (Activation)  (None, 4, 6, 384)           0         ['batch_normalization_92[0][0]
                                                                  ']

batch_normalization_93 (Ba  (None, 4, 6, 192)           576       ['conv2d_93[0][0]']
tchNormalization)

activation_85 (Activation)  (None, 4, 6, 320)           0         ['batch_normalization_85[0][0]
                                                                  ']

mixed9_1 (Concatenate)      (None, 4, 6, 768)           0         ['activation_87[0][0]',
                                                                   'activation_88[0][0]']

concatenate_1 (Concatenate  (None, 4, 6, 768)           0         ['activation_91[0][0]',
)                                                                  'activation_92[0][0]']

activation_93 (Activation)  (None, 4, 6, 192)           0         ['batch_normalization_93[0][0]
                                                                  ']

mixed10 (Concatenate)       (None, 4, 6, 2048)          0         ['activation_85[0][0]',
                                                                   'mixed9_1[0][0]',
                                                                   'concatenate_1[0][0]',
                                                                   'activation_93[0][0]']

global_max_pooling2d_1 (Gl  (None, 2048)                0         ['mixed10[0][0]']
obalMaxPooling2D)

dense_2 (Dense)             (None, 512)                 1049088   ['global_max_pooling2d_1[0][0]
                                                                  ']

dropout_1 (Dropout)         (None, 512)                 0         ['dense_2[0][0]']

dense_3 (Dense)             (None, 7)                   3591      ['dropout_1[0][0]']

==================================================================================================
Total params: 22855463 (87.19 MB)
Trainable params: 1069895 (4.08 MB)
Non-trainable params: 21785568 (83.11 MB)
__________________________________________________________________________________________________
```

```
train_datagen = ImageDataGenerator(rotation_range=60, width_shift_range=0.2, height_shift_range=0.2,
                                   shear_range=0.2, zoom_range=0.2, fill_mode='nearest')

train_datagen.fit(X_train)

val_datagen = ImageDataGenerator()
val_datagen.fit(X_val)
```

```
batch_size = 64
epochs = 1
history = model.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),
                              epochs = epochs, validation_data = val_datagen.flow(X_val, y_val),
                              verbose = 1, steps_per_epoch=(X_train.shape[0] // batch_size),
                              validation_steps=(X_val.shape[0] // batch_size))
```

```
<ipython-input-29-f5452c349cd2>:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please
  history = model.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),
71/71 [==============================] - 95s 1s/step - loss: 2.7143 - accuracy: 0.3802 - val_loss: 4914.6997 - val_accuracy: 0.6116
```

```
for layer in pre_trained_model.layers[249:]:
    layer.trainable = True
```

```
#optimizer = tensorflow.keras.optimizers.legacy.SGD(lr=0.0001, decay=0.0)
model.compile(loss='categorical_crossentropy',
              optimizer="adam",
              metrics=['acc'])
```

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc', patience=3, verbose=1, factor=0.5,
                                            min_lr=0.000001, cooldown=2)
```

```
model.summary()
```

```
 activation_87 (Activation)  (None, 4, 6, 384)           0         ['batch_normalization_87[0][0]
                                                                   ']

 activation_88 (Activation)  (None, 4, 6, 384)           0         ['batch_normalization_88[0][0]
                                                                   ']

 activation_91 (Activation)  (None, 4, 6, 384)           0         ['batch_normalization_91[0][0]
                                                                   ']

 activation_92 (Activation)  (None, 4, 6, 384)           0         ['batch_normalization_92[0][0]
                                                                   ']

 batch_normalization_93 (Ba  (None, 4, 6, 192)           576       ['conv2d_93[0][0]']
 tchNormalization)

 activation_85 (Activation)  (None, 4, 6, 320)           0         ['batch_normalization_85[0][0]
                                                                   ']

 mixed9_1 (Concatenate)      (None, 4, 6, 768)           0         ['activation_87[0][0]',
                                                                    'activation_88[0][0]']

 concatenate_1 (Concatenate  (None, 4, 6, 768)           0         ['activation_91[0][0]',
 )                                                                  'activation_92[0][0]']

 activation_93 (Activation)  (None, 4, 6, 192)           0         ['batch_normalization_93[0][0]
                                                                   ']

 mixed10 (Concatenate)       (None, 4, 6, 2048)          0         ['activation_85[0][0]',
                                                                    'mixed9_1[0][0]',
                                                                    'concatenate_1[0][0]',
                                                                    'activation_93[0][0]']

 global_max_pooling2d_1 (Gl  (None, 2048)                0         ['mixed10[0][0]']
 obalMaxPooling2D)

 dense_2 (Dense)             (None, 512)                 1049088   ['global_max_pooling2d_1[0][0]
                                                                   ']

 dropout_1 (Dropout)         (None, 512)                 0         ['dense_2[0][0]']

 dense_3 (Dense)             (None, 7)                   3591      ['dropout_1[0][0]']

==================================================================================================
Total params: 22855463 (87.19 MB)
Trainable params: 1069895 (4.08 MB)
Non-trainable params: 21785568 (83.11 MB)
__________________________________________________________________________________________________
```

```
batch_size = 64
epochs = 30
history = model.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),
                              epochs = epochs, validation_data = val_datagen.flow(X_val, y_val),
                              verbose = 1, steps_per_epoch=(X_train.shape[0] // batch_size),
                              validation_steps=(X_val.shape[0] // batch_size),
                              callbacks=[learning_rate_reduction])
```

```
t-25-1f224cf570ef>:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model
odel.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),

========================] - 64s 902ms/step - loss: 0.8408 - acc: 0.6946 - val_loss: 0.8485 - val_acc: 0.6696 - lr: 2.5000e-04

========================] - 65s 911ms/step - loss: 0.8428 - acc: 0.6977 - val_loss: 0.9050 - val_acc: 0.6518 - lr: 2.5000e-04

========================] - 66s 925ms/step - loss: 0.8542 - acc: 0.7012 - val_loss: 0.7667 - val_acc: 0.7366 - lr: 2.5000e-04

========================] - 65s 907ms/step - loss: 0.8456 - acc: 0.7048 - val_loss: 0.8052 - val_acc: 0.7366 - lr: 2.5000e-04

========================] - 66s 915ms/step - loss: 0.8403 - acc: 0.6988 - val_loss: 0.8654 - val_acc: 0.6920 - lr: 2.5000e-04

========================] - ETA: 0s - loss: 0.8437 - acc: 0.6981
ceLROnPlateau reducing learning rate to 0.0001250000059371814.
========================] - 63s 882ms/step - loss: 0.8437 - acc: 0.6981 - val_loss: 0.9428 - val_acc: 0.6562 - lr: 2.5000e-04

========================] - 65s 917ms/step - loss: 0.8256 - acc: 0.7001 - val_loss: 0.8719 - val_acc: 0.6652 - lr: 1.2500e-04

========================] - 62s 873ms/step - loss: 0.8250 - acc: 0.7039 - val_loss: 0.8032 - val_acc: 0.7009 - lr: 1.2500e-04

========================] - 66s 913ms/step - loss: 0.8251 - acc: 0.7109 - val_loss: 0.7904 - val_acc: 0.7054 - lr: 1.2500e-04

========================] - ETA: 0s - loss: 0.8206 - acc: 0.7052
uceLROnPlateau reducing learning rate to 6.25000029685907e-05.
========================] - 64s 886ms/step - loss: 0.8206 - acc: 0.7052 - val_loss: 0.7813 - val_acc: 0.7054 - lr: 1.2500e-04

========================] - 64s 905ms/step - loss: 0.8154 - acc: 0.6992 - val_loss: 0.9157 - val_acc: 0.6830 - lr: 6.2500e-05

========================] - 67s 945ms/step - loss: 0.8095 - acc: 0.7063 - val_loss: 0.9237 - val_acc: 0.6473 - lr: 6.2500e-05

========================] - 65s 911ms/step - loss: 0.8189 - acc: 0.7032 - val_loss: 0.8888 - val_acc: 0.6652 - lr: 6.2500e-05
```

```
========================] - ETA: 0s - loss: 0.7980 - acc: 0.7101
uceLROnPlateau reducing learning rate to 3.1250001484295350e-05.
========================] - 66s 925ms/step - loss: 0.7980 - acc: 0.7101 - val_loss: 0.7656 - val_acc: 0.7009 - lr: 6.2500e-05

========================] - 65s 906ms/step - loss: 0.8101 - acc: 0.7087 - val_loss: 0.7916 - val_acc: 0.6920 - lr: 3.1250e-05

========================] - 65s 913ms/step - loss: 0.8244 - acc: 0.7028 - val_loss: 0.9440 - val_acc: 0.6830 - lr: 3.1250e-05

========================] - 67s 940ms/step - loss: 0.8032 - acc: 0.7056 - val_loss: 0.8129 - val_acc: 0.7411 - lr: 3.1250e-05

========================] - 63s 887ms/step - loss: 0.8093 - acc: 0.7030 - val_loss: 0.9083 - val_acc: 0.6652 - lr: 3.1250e-05

========================] - 66s 922ms/step - loss: 0.8199 - acc: 0.7096 - val_loss: 0.8205 - val_acc: 0.7232 - lr: 3.1250e-05

========================] - ETA: 0s - loss: 0.8141 - acc: 0.7068
uceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
========================] - 63s 885ms/step - loss: 0.8141 - acc: 0.7068 - val_loss: 0.8434 - val_acc: 0.6920 - lr: 3.1250e-05

========================] - 67s 936ms/step - loss: 0.8081 - acc: 0.7021 - val_loss: 0.8003 - val_acc: 0.7054 - lr: 1.5625e-05

========================] - 64s 904ms/step - loss: 0.8025 - acc: 0.7052 - val_loss: 0.8053 - val_acc: 0.7143 - lr: 1.5625e-05

========================] - 65s 912ms/step - loss: 0.7982 - acc: 0.7092 - val_loss: 0.8272 - val_acc: 0.7054 - lr: 1.5625e-05
```

```
loss_val, acc_val = model.evaluate(X_val, y_val, verbose=1)
print("Validation: accuracy = %f  ;  loss_v = %f" % (acc_val, loss_val))
```

```
16/16 [==============================] - 4s 250ms/step - loss: 0.8304 - acc: 0.6888
Validation: accuracy = 0.688845  ;  loss_v = 0.830358
```

```
X_test = np.load("/content/drive/MyDrive/Colab Notebooks/256_192_test.npy")
```

```
y_test = np.load("/content/drive/MyDrive/Colab Notebooks/test_labels.npy")
y_test = to_categorical(y_test)
```

```
loss_test, acc_test = model.evaluate(X_test, y_test, verbose=1)
print("Test: accuracy = %f  ;  loss = %f" % (acc_test, loss_test))
```

```
model.save("InceptionV3FT.h5")
```

```
C:\Users\dell6\anaconda3\lib\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving your model as an HDF5 fil
  saving_api.save_model(
```

```
# Retrieve a list of accuracy results on training and test data
# sets for each training epoch
acc = history.history['acc']
val_acc = history.history['val_acc']

# Retrieve a list of list results on training and test data
# sets for each training epoch
loss = history.history['loss']
val_loss = history.history['val_loss']

# Get number of epochs
epochs = range(len(acc))

# Plot training and validation accuracy per epoch
plt.plot(epochs, acc, label = "training")
plt.plot(epochs, val_acc, label = "validation")
plt.legend(loc="upper left")
plt.title('Training and validation accuracy')

plt.figure()

# Plot training and validation loss per epoch
plt.plot(epochs, loss, label = "training")
plt.plot(epochs, val_loss, label = "validation")
plt.legend(loc="upper right")
plt.title('Training and validation loss')
```

```
Text(0.5, 1.0, 'Training and validation loss')
```