```
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 # convert to one-hot-encoding

from keras.preprocessing.image import ImageDataGenerator
from keras import layers
from keras import Model
from keras.applications.inception_resnet_v2 import InceptionResNetV2
from keras.optimizers import Adam
from keras.callbacks import ReduceLROnPlateau
import keras.backend as K

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

## Load in the Dataset

```
X_train = np.load("/content/drive/MyDrive/Colab Notebooks/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)
```

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

```
((4596, 7), (511, 7))
```

## Load Pretrained Model

```
pre_trained_model = InceptionResNetV2(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))
```

```
conv2d_192
batch_normalization_192
activation_192
conv2d_193
batch_normalization_193
activation_193
conv2d_191
conv2d_194
batch_normalization_191
batch_normalization_194
activation_191
activation_194
block8_8_mixed
block8_8_conv
custom_scale_layer_37
block8_8_ac
conv2d_196
batch_normalization_196
activation_196
conv2d_197
batch_normalization_197
activation_197
conv2d_195
conv2d_198
batch_normalization_195
batch_normalization_198
activation_195
activation_198
block8_9_mixed
block8_9_conv
custom_scale_layer_38
block8_9_ac
conv2d_200
batch_normalization_200
activation_200
conv2d_201
batch_normalization_201
activation_201
conv2d_199
conv2d_202
batch_normalization_199
batch_normalization_202
activation_199
activation_202
block8_10_mixed
block8_10_conv
custom_scale_layer_39
conv_7b
conv_7b_bn
conv_7b_ac
780
```

```
last_layer = pre_trained_model.get_layer('conv_7b_ac')
print('last layer output shape:', last_layer.output_shape)
last_output = last_layer.output
```

```
last layer output shape: (None, 4, 6, 1536)
```

## Define the Model

```
# 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 0.7
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 = Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=True)
model.compile(loss='categorical_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])


model.summary()
```

```
activation_200 (Activation  (None, 4, 6, 192)           0         ['batch_normalization_200[0][0
)                                                                  ]']

conv2d_201 (Conv2D)         (None, 4, 6, 224)           129024    ['activation_200[0][0]']

batch_normalization_201 (B  (None, 4, 6, 224)           672       ['conv2d_201[0][0]']
atchNormalization)

activation_201 (Activation  (None, 4, 6, 224)           0         ['batch_normalization_201[0][0
)                                                                  ]']

conv2d_199 (Conv2D)         (None, 4, 6, 192)           399360    ['block8_9_ac[0][0]']

conv2d_202 (Conv2D)         (None, 4, 6, 256)           172032    ['activation_201[0][0]']

batch_normalization_199 (B  (None, 4, 6, 192)           576       ['conv2d_199[0][0]']
atchNormalization)

batch_normalization_202 (B  (None, 4, 6, 256)           768       ['conv2d_202[0][0]']
atchNormalization)

activation_199 (Activation  (None, 4, 6, 192)           0         ['batch_normalization_199[0][0
)                                                                  ]']

activation_202 (Activation  (None, 4, 6, 256)           0         ['batch_normalization_202[0][0
)                                                                  ]']

block8_10_mixed (Concatena  (None, 4, 6, 448)           0         ['activation_199[0][0]',
te)                                                                'activation_202[0][0]']

block8_10_conv (Conv2D)     (None, 4, 6, 2080)          933920    ['block8_10_mixed[0][0]']

custom_scale_layer_39 (Cus  (None, 4, 6, 2080)          0         ['block8_9_ac[0][0]',
tomScaleLayer)                                                     'block8_10_conv[0][0]']

conv_7b (Conv2D)            (None, 4, 6, 1536)          3194880   ['custom_scale_layer_39[0][0]'
                                                                   ]

conv_7b_bn (BatchNormaliza  (None, 4, 6, 1536)          4608      ['conv_7b[0][0]']
tion)

conv_7b_ac (Activation)     (None, 4, 6, 1536)          0         ['conv_7b_bn[0][0]']

global_max_pooling2d (Glob  (None, 1536)                0         ['conv_7b_ac[0][0]']
alMaxPooling2D)

dense (Dense)               (None, 512)                 786944    ['global_max_pooling2d[0][0]']

dropout (Dropout)           (None, 512)                 0         ['dense[0][0]']

dense_1 (Dense)             (None, 7)                   3591      ['dropout[0][0]']

==================================================================================================
Total params: 55127271 (210.29 MB)
Trainable params: 820807 (3.13 MB)
Non-trainable params: 54306464 (207.16 MB)
__________________________________________________________________________________________________
```

## Training

### Feature Extraction

```
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-17-f5452c349cd2>:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
  history = model.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),
71/71 [==============================] - 151s 1s/step - loss: 1.9539 - accuracy: 0.6158 - val_loss: 647903.6250 - val_accuracy: 0.6607
```

## Fine Tuning

```
pre_trained_model.layers[617].name
```

```
'mixed_7a'
```

```
for layer in pre_trained_model.layers[618:]:
    layer.trainable = True
```

```
model.compile(loss='categorical_crossentropy',
              metrics=['acc'])
```

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

```
model.summary()
```

```
 dense (Dense)               (None, 512)               786944    ['global_max_pooling2d[0][0]']

 dropout (Dropout)           (None, 512)               0         ['dense[0][0]']

 dense_1 (Dense)             (None, 7)                 3591      ['dropout[0][0]']

==================================================================================================
Total params: 55127271 (210.29 MB)
Trainable params: 24352647 (92.90 MB)
Non-trainable params: 30774624 (117.40 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])
```

```
<ipython-input-23-1f224cf570ef>:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please u
  history = model.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),
Epoch 1/30
71/71 [==============================] - 146s 1s/step - loss: 1.4744 - acc: 0.6602 - val_loss: 1368668032.0000 - val_acc: 0.6741 - lr
Epoch 2/30
71/71 [==============================] - 79s 1s/step - loss: 1.0009 - acc: 0.7045 - val_loss: 2322.5171 - val_acc: 0.6295 - lr: 0.001
Epoch 3/30
71/71 [==============================] - 81s 1s/step - loss: 0.8553 - acc: 0.7231 - val_loss: 25198.4590 - val_acc: 0.6295 - lr: 0.00
Epoch 4/30
71/71 [==============================] - 79s 1s/step - loss: 0.7220 - acc: 0.7493 - val_loss: 0.8327 - val_acc: 0.6964 - lr: 0.0010
Epoch 5/30
71/71 [==============================] - 80s 1s/step - loss: 0.6657 - acc: 0.7723 - val_loss: 0.7998 - val_acc: 0.7143 - lr: 0.0010
Epoch 6/30
71/71 [==============================] - 78s 1s/step - loss: 0.6701 - acc: 0.7895 - val_loss: 84.3628 - val_acc: 0.7143 - lr: 0.0010
Epoch 7/30
71/71 [==============================] - 79s 1s/step - loss: 0.6251 - acc: 0.7950 - val_loss: 2.6259 - val_acc: 0.7902 - lr: 0.0010
Epoch 8/30
71/71 [==============================] - 79s 1s/step - loss: 0.5490 - acc: 0.8197 - val_loss: 0.8071 - val_acc: 0.7679 - lr: 0.0010
Epoch 9/30
71/71 [==============================] - 80s 1s/step - loss: 0.5172 - acc: 0.8248 - val_loss: 0.5987 - val_acc: 0.7946 - lr: 0.0010
Epoch 10/30
71/71 [==============================] - 78s 1s/step - loss: 0.4973 - acc: 0.8255 - val_loss: 0.8157 - val_acc: 0.7679 - lr: 0.0010
Epoch 11/30
71/71 [==============================] - 80s 1s/step - loss: 0.4548 - acc: 0.8438 - val_loss: 0.7964 - val_acc: 0.7634 - lr: 0.0010
Epoch 12/30
71/71 [==============================] - ETA: 0s - loss: 0.4489 - acc: 0.8475
Epoch 12: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
71/71 [==============================] - 77s 1s/step - loss: 0.4489 - acc: 0.8475 - val_loss: 0.7357 - val_acc: 0.7723 - lr: 0.0010
Epoch 13/30
71/71 [==============================] - 78s 1s/step - loss: 0.3734 - acc: 0.8795 - val_loss: 0.6655 - val_acc: 0.8080 - lr: 5.0000e-
Epoch 14/30
71/71 [==============================] - 79s 1s/step - loss: 0.3266 - acc: 0.8861 - val_loss: 0.7063 - val_acc: 0.7723 - lr: 5.0000e-
Epoch 15/30
71/71 [==============================] - 78s 1s/step - loss: 0.3068 - acc: 0.8947 - val_loss: 0.6631 - val_acc: 0.7946 - lr: 5.0000e-
Epoch 16/30
71/71 [==============================] - 80s 1s/step - loss: 0.2986 - acc: 0.8932 - val_loss: 0.5284 - val_acc: 0.8259 - lr: 5.0000e-
Epoch 17/30
71/71 [==============================] - 78s 1s/step - loss: 0.2781 - acc: 0.9027 - val_loss: 0.7888 - val_acc: 0.8125 - lr: 5.0000e-
Epoch 18/30
71/71 [==============================] - 79s 1s/step - loss: 0.2502 - acc: 0.9078 - val_loss: 0.7443 - val_acc: 0.7902 - lr: 5.0000e-
Epoch 19/30
71/71 [==============================] - ETA: 0s - loss: 0.2515 - acc: 0.9148
Epoch 19: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
71/71 [==============================] - 76s 1s/step - loss: 0.2515 - acc: 0.9148 - val_loss: 0.9081 - val_acc: 0.8036 - lr: 5.0000e-
Epoch 20/30
71/71 [==============================] - 79s 1s/step - loss: 0.2123 - acc: 0.9230 - val_loss: 0.6840 - val_acc: 0.7991 - lr: 2.5000e-
Epoch 21/30
71/71 [==============================] - 78s 1s/step - loss: 0.1937 - acc: 0.9318 - val_loss: 0.9520 - val_acc: 0.7500 - lr: 2.5000e-
Epoch 22/30
71/71 [==============================] - 78s 1s/step - loss: 0.2028 - acc: 0.9316 - val_loss: 0.8878 - val_acc: 0.7857 - lr: 2.5000e-
Epoch 23/30
71/71 [==============================] - ETA: 0s - loss: 0.1765 - acc: 0.9389
Epoch 23: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
71/71 [==============================] - 78s 1s/step - loss: 0.1765 - acc: 0.9389 - val_loss: 0.8290 - val_acc: 0.8036 - lr: 2.5000e-
Epoch 24/30
71/71 [==============================] - 78s 1s/step - loss: 0.1599 - acc: 0.9444 - val_loss: 0.8549 - val_acc: 0.8125 - lr: 1.2500e-
Epoch 25/30
```

```
model.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),
                    epochs = 30, 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])
```

```
<ipython-input-20-fba5f059e7e1>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
  model.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),
Epoch 1/10
58/71 [=======================>......] - ETA: 16s - loss: 2.5644 - accuracy: 0.5846
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-20-fba5f059e7e1> in <cell line: 1>()
----> 1 model.fit_generator(train_datagen.flow(X_train,y_train, batch_size=batch_size),
      2                     epochs = 10, validation_data = val_datagen.flow(X_val, y_val),
      3                     verbose = 1, steps_per_epoch=(X_train.shape[0] // batch_size),
      4                     validation_steps=(X_val.shape[0] // batch_size),
      5                     callbacks=[learning_rate_reduction])

16 frames
/usr/local/lib/python3.10/dist-packages/tensorflow/python/framework/ops.py in _numpy(self)
    358   def _numpy(self):
    359     try:
--> 360       return self._numpy_internal()
    361     except core._NotOkStatusException as e:  # pylint: disable=protected-access
    362       raise core._status_to_exception(e) from None  # pylint: disable=protected-access

KeyboardInterrupt:
```

```
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 [==============================] - 6s 365ms/step - loss: 0.9095 - acc: 0.8141
Validation: accuracy = 0.814090  ;  loss_v = 0.909516
```

## Testing

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-52e2a8a18b3d> in <cell line: 1>()
----> 1 X_test = np.load("/content/drive/MyDrive/Colab Notebooks/256_192_train.npy")

NameError: name 'np' is not defined
```

Next steps: Explain error

```
y_test = np.load("/content/drive/MyDrive/Colab Notebooks/256_192_train/test_labels.npy")
y_test = to_categorical(y_test)
```

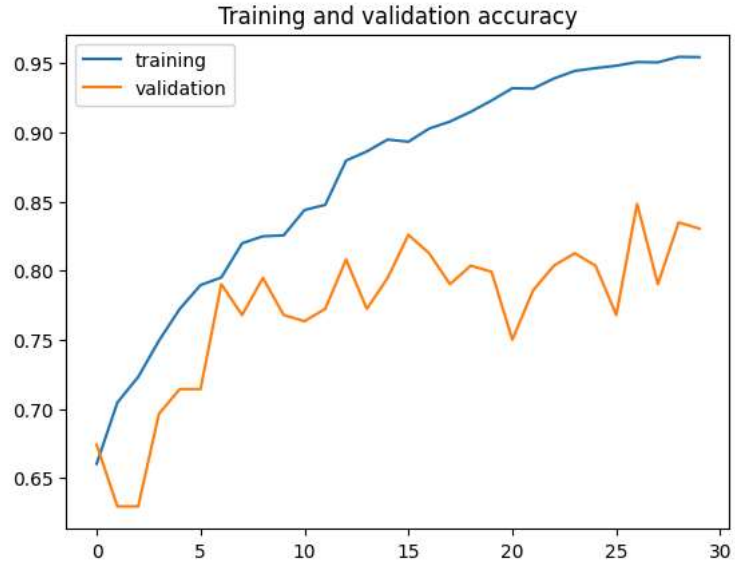
```
loss_test, acc_test = model.evaluate(X_test, y_test, verbose=1)
print("Test: accuracy = %f  ;  loss = %f" % (acc_test, loss_test))
```

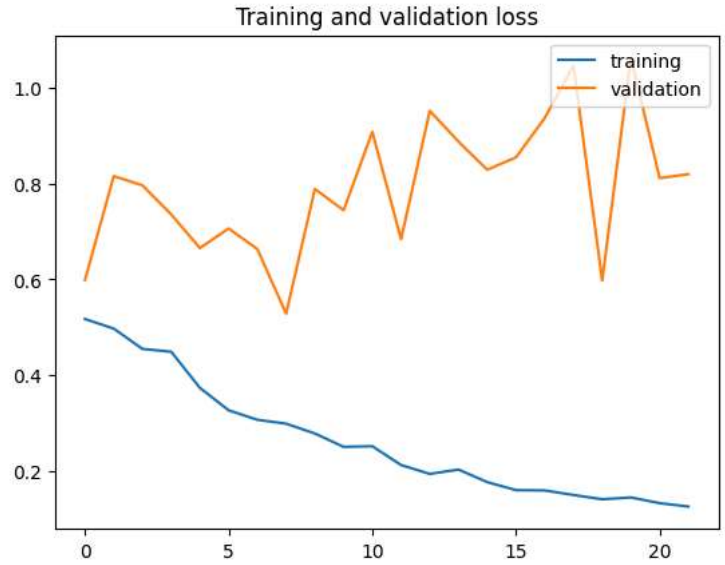
```
model.save("InceptionResNet.h5")
```

```
# 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()
epochs2 = range(len(lose_s))

# Plot training and validation loss per epoch
plt.plot(epochs2, lose_s, label = "training")
plt.plot(epochs2, val_lose_s, label = "validation")
plt.legend(loc="upper right")
plt.title('Training and validation loss')
```

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




```
type(val_loss)
```

```
list
```

```
lose_s=loss[8:]
val_lose_s=val_loss[8:]
```