Using MobileNet

Loading the MobileNet Model

Freeze all layers except the top 4, as we'll only be training the top 4

In [1]:
```
from keras.applications import MobileNet

# MobileNet was designed to work on 224 x 224 pixel input images sizes
img_rows, img_cols = 224, 224

# Re-loads the MobileNet model without the top or FC layers
MobileNet = MobileNet(weights = 'imagenet',
                      include_top = False,
                      input_shape = (img_rows, img_cols, 3))

# Here we freeze the last 4 layers
# Layers are set to trainable as True by default
for layer in MobileNet.layers:
    layer.trainable = False

# Let's print our layers
for (i,layer) in enumerate(MobileNet.layers):
    print(str(i) + " "+ layer.__class__.__name__, layer.trainable)
```

```
Using TensorFlow backend.
```

```
0 InputLayer False
1 ZeroPadding2D False
2 Conv2D False
3 BatchNormalization False
4 ReLU False
5 DepthwiseConv2D False
6 BatchNormalization False
```

```
4 ReLU False
5 DepthwiseConv2D False
6 BatchNormalization False
7 ReLU False
8 Conv2D False
9 BatchNormalization False
10 ReLU False
11 ZeroPadding2D False
12 DepthwiseConv2D False
13 BatchNormalization False
14 ReLU False
15 Conv2D False
16 BatchNormalization False
17 ReLU False
18 DepthwiseConv2D False
19 BatchNormalization False
20 ReLU False
21 Conv2D False
22 BatchNormalization False
23 ReLU False
24 ZeroPadding2D False
25 DepthwiseConv2D False
26 BatchNormalization False
27 ReLU False
28 Conv2D False
29 BatchNormalization False
30 ReLU False
31 DepthwiseConv2D False
32 BatchNormalization False
33 ReLU False
34 Conv2D False
35 BatchNormalization False
36 ReLU False
37 ZeroPadding2D False
38 DepthwiseConv2D False
39 BatchNormalization False
40 ReLU False
```

```
44 DepthwiseConv2D False
45 BatchNormalization False
46 ReLU False
47 Conv2D False
48 BatchNormalization False
49 ReLU False
50 DepthwiseConv2D False
51 BatchNormalization False
52 ReLU False
53 Conv2D False
54 BatchNormalization False
55 ReLU False
56 DepthwiseConv2D False
57 BatchNormalization False
58 ReLU False
59 Conv2D False
60 BatchNormalization False
61 ReLU False
62 DepthwiseConv2D False
63 BatchNormalization False
64 ReLU False
65 Conv2D False
66 BatchNormalization False
67 ReLU False
68 DepthwiseConv2D False
69 BatchNormalization False
70 ReLU False
71 Conv2D False
72 BatchNormalization False
73 ReLU False
74 ZeroPadding2D False
75 DepthwiseConv2D False
76 BatchNormalization False
77 ReLU False
78 Conv2D False
79 BatchNormalization False
80 ReLU False
81 DepthwiseConv2D False
82 BatchNormalization False
```

Let's make a function that returns our FC Head

In [2]:
```
def lw(bottom_model, num_classes):
    """creates the top or head of the model that will be
    placed ontop of the bottom layers"""

    top_model = bottom_model.output
    top_model = GlobalAveragePooling2D()(top_model)
    top_model = Dense(1024,activation='relu')(top_model)
    top_model = Dense(1024,activation='relu')(top_model)
    top_model = Dense(512,activation='relu')(top_model)
    top_model = Dense(num_classes,activation='softmax')(top_model)
    return top_model
```

Let's add our FC Head back onto MobileNet

In [3]:
```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, GlobalAveragePooling2D
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.layers.normalization import BatchNormalization
from keras.models import Model

# Set our class number to 3 (Young, Middle, Old)
num_classes = 2

FC_Head = lw(MobileNet, num_classes)

model = Model(inputs = MobileNet.input, outputs = FC_Head)

print(model.summary())
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 225, 225, 3)       0
_________________________________________________________________
conv1 (Conv2D)               (None, 112, 112, 32)      864
_________________________________________________________________
conv1_bn (BatchNormalization (None, 112, 112, 32)      128
_________________________________________________________________
conv1_relu (ReLU)            (None, 112, 112, 32)      0
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)      288
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, 112, 112, 32)      128
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, 112, 112, 32)      0
```

Loading our Own Dataset

In [4]:
```
from keras.preprocessing.image import ImageDataGenerator

train_data_dir = 'mydata/train/'
validation_data_dir = 'mydata/validation/'

# Let's use some data augmentaiton
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=45,
      width_shift_range=0.3,
      height_shift_range=0.3,
      horizontal_flip=True,
      fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale=1./255)

# set our batch size (typically on most mid tier systems we'll use 16-32)
batch_size = 32

train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_rows, img_cols),
        batch_size=batch_size,
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_rows, img_cols),
        batch_size=batch_size,
        class_mode='categorical')
```

```
Found 108 images belonging to 2 classes.
Found 26 images belonging to 2 classes.
```

Training out Model

- Note we're using checkpointing and early stopping

In [5]:
```
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping


checkpoint = ModelCheckpoint("human_dataset.h5",
                             monitor="val_loss",
                             mode="min",

                             save_best_only = True,
                             verbose=1)

earlystop = EarlyStopping(monitor = 'val_loss',
                          min_delta = 0,
                          patience = 3,
                          verbose = 1,
                          restore_best_weights = True)

# we put our call backs into a callback list
callbacks = [earlystop, checkpoint]

# We use a very small learning rate
model.compile(loss = 'categorical_crossentropy',
              optimizer = RMSprop(lr = 0.001),
              metrics = ['accuracy'])

# Enter the number of training and validation samples here
nb_train_samples = 200
nb_validation_samples = 50

# We only train 5 EPOCHS
epochs = 5
batch_size = 16
```

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch = nb_train_samples // batch_size,
    epochs = epochs,
    callbacks = callbacks,
    validation_data = validation_generator,
    validation_steps = nb_validation_samples // batch_size)
```

```
Epoch 1/5
12/12 [==============================] - 42s 3s/step - loss: 6.4891 - accuracy: 0.5123 - val_loss: 0.7477 - val_accur
acy: 0.3846

Epoch 00001: val_loss improved from inf to 0.74767, saving model to human_dataset.h5
Epoch 2/5
12/12 [==============================] - 38s 3s/step - loss: 0.6976 - accuracy: 0.5617 - val_loss: 0.6807 - val_accur
acy: 0.6538

Epoch 00002: val_loss improved from 0.74767 to 0.68067, saving model to human_dataset.h5
Epoch 3/5
12/12 [==============================] - 38s 3s/step - loss: 0.7015 - accuracy: 0.5833 - val_loss: 4.3557 - val_accur
acy: 0.3462

Epoch 00003: val_loss did not improve from 0.68067
Epoch 4/5
12/12 [==============================] - 38s 3s/step - loss: 0.8123 - accuracy: 0.6265 - val_loss: 0.8293 - val_accur
acy: 0.5385

Epoch 00004: val_loss did not improve from 0.68067
Epoch 5/5
12/12 [==============================] - 37s 3s/step - loss: 0.6606 - accuracy: 0.7068 - val_loss: 1.1875 - val_accur
acy: 0.6538
Restoring model weights from the end of the best epoch

Epoch 00005: val_loss did not improve from 0.68067
Epoch 00005: early stopping
```

Loading our classifer

In [6]:
```
from keras.models import load_model

classifier = load_model('human_dataset.h5')
```

Testing our classifer on some test images

In [7]:
```
import os
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join

human_dict = { "[0]": "Akshay Kumar",
                       "[1]": "Salman Khan" }

human_dict_n = { "akshaykumar": "Akshay Kumar",
                         "salmankhan": "Salman Khan" }

def draw_test(name, pred, im):
    person = human_dict[str(pred)]
    BLACK = [0,0,0]
    expanded_image = cv2.copyMakeBorder(im, 80, 0, 0, 100 ,cv2.BORDER_CONSTANT,value=BLACK)
    cv2.putText(expanded_image, person, (20, 60) , cv2.FONT_HERSHEY_SIMPLEX,1, (0,0,255), 2)
    cv2.imshow(name, expanded_image)

def getRandomImage(path):
    """function loads a random images from a random folder in our test path """
    folders = list(filter(lambda x: os.path.isdir(os.path.join(path, x)), os.listdir(path)))
    random_directory = np.random.randint(0,len(folders))
    path_class = folders[random_directory]
    print("Class - " + human_dict_n[str(path_class)])
    file_path = path + path_class
```

```
    path_class = folders[random_directory]
    print("Class - " + human_dict_n[str(path_class)])
    file_path = path + path_class
    file_names = [f for f in listdir(file_path) if isfile(join(file_path, f))]
    random_file_index = np.random.randint(0,len(file_names))
    image_name = file_names[random_file_index]
    return cv2.imread(file_path+"/"+image_name)

for i in range(0,10):
    input_im = getRandomImage("mydata/validation/")
    input_original = input_im.copy()
    input_original = cv2.resize(input_original, None, fx=0.5, fy=0.5, interpolation = cv2.INTER_LINEAR)

    input_im = cv2.resize(input_im, (224, 224), interpolation = cv2.INTER_LINEAR)
    input_im = input_im / 255.
    input_im = input_im.reshape(1,224,224,3)

    # Get Prediction
    res = np.argmax(classifier.predict(input_im, 1, verbose = 0), axis=1)

    # Show image with predicted class
    draw_test("Prediction", res, input_original)
    cv2.waitKey(1000)

cv2.destroyAllWindows()
```

```
Class - Salman Khan
Class - Akshay Kumar
Class - Akshay Kumar
Class - Salman Khan
Class - Salman Khan
Class - Akshay Kumar
Class - Akshay Kumar
Class - Salman Khan
Class - Akshay Kumar
Class - Akshay Kumar
```