Using MobileNet for our Monkey Classifer

Loading the MobileNet Model

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

In [8]:
```python
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)
```

```
0 InputLayer False
1 ZeroPadding2D False
2 Conv2D False
3 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
41 Conv2D False
42 BatchNormalization False
43 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
83 ReLU False
84 Conv2D False
85 BatchNormalization False
86 ReLU False
```

Let's make a function that returns our FC Head

In [9]:
```python
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(512,activation='relu')(top_model)
    top_model = Dense(256,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 [10]:
```python
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 = 3

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         
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, 112, 112, 64)      2048      
_________________________________________________________________
conv_pw_1_bn (BatchNormaliza (None, 112, 112, 64)      256       
_________________________________________________________________
conv_pw_1_relu (ReLU)        (None, 112, 112, 64)      0         
_________________________________________________________________
conv_pad_2 (ZeroPadding2D)   (None, 113, 113, 64)      0         
_________________________________________________________________
conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 64)        576       
_________________________________________________________________
conv_dw_2_bn (BatchNormaliza (None, 56, 56, 64)        256       
_________________________________________________________________
conv_dw_2_relu (ReLU)        (None, 56, 56, 64)        0         
_________________________________________________________________
conv_pw_2 (Conv2D)           (None, 56, 56, 128)       8192      
_________________________________________________________________
conv_pw_2_bn (BatchNormaliza (None, 56, 56, 128)       512       
_________________________________________________________________
conv_pw_2_relu (ReLU)        (None, 56, 56, 128)       0         
_________________________________________________________________
conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)       1152      
_________________________________________________________________
conv_dw_3_bn (BatchNormaliza (None, 56, 56, 128)       512       
_________________________________________________________________
conv_dw_3_relu (ReLU)        (None, 56, 56, 128)       0         
_________________________________________________________________
conv_pw_3 (Conv2D)           (None, 56, 56, 128)       16384     
_________________________________________________________________
conv_pw_3_bn (BatchNormaliza (None, 56, 56, 128)       512       
_________________________________________________________________
conv_pw_3_relu (ReLU)        (None, 56, 56, 128)       0         
_________________________________________________________________
conv_pad_4 (ZeroPadding2D)   (None, 57, 57, 128)       0         
_________________________________________________________________
conv_dw_4 (DepthwiseConv2D)  (None, 28, 28, 128)       1152      
_________________________________________________________________
conv_dw_4_bn (BatchNormaliza (None, 28, 28, 128)       512       
_________________________________________________________________
conv_dw_4_relu (ReLU)        (None, 28, 28, 128)       0         
_________________________________________________________________
conv_pw_4 (Conv2D)           (None, 28, 28, 256)       32768     
_________________________________________________________________
conv_pw_4_bn (BatchNormaliza (None, 28, 28, 256)       1024      
_________________________________________________________________
conv_pw_4_relu (ReLU)        (None, 28, 28, 256)       0         
_________________________________________________________________
conv_dw_5 (DepthwiseConv2D)  (None, 28, 28, 256)       2304      
_________________________________________________________________
conv_dw_5_bn (BatchNormaliza (None, 28, 28, 256)       1024      
_________________________________________________________________
conv_dw_5_relu (ReLU)        (None, 28, 28, 256)       0         
_________________________________________________________________
conv_pw_5 (Conv2D)           (None, 28, 28, 256)       65536     
_________________________________________________________________
conv_pw_5_bn (BatchNormaliza (None, 28, 28, 256)       1024      
_________________________________________________________________
conv_pw_5_relu (ReLU)        (None, 28, 28, 256)       0         
_________________________________________________________________
conv_pad_6 (ZeroPadding2D)   (None, 29, 29, 256)       0         
_________________________________________________________________
conv_dw_6 (DepthwiseConv2D)  (None, 14, 14, 256)       2304      
_________________________________________________________________
conv_dw_6_bn (BatchNormaliza (None, 14, 14, 256)       1024      
_________________________________________________________________
conv_dw_6_relu (ReLU)        (None, 14, 14, 256)       0         
_________________________________________________________________
conv_pw_6 (Conv2D)           (None, 14, 14, 512)       131072    
_________________________________________________________________
conv_pw_6_bn (BatchNormaliza (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_pw_6_relu (ReLU)        (None, 14, 14, 512)       0         
_________________________________________________________________
conv_dw_7 (DepthwiseConv2D)  (None, 14, 14, 512)       4608      
_________________________________________________________________
conv_dw_7_bn (BatchNormaliza (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_dw_7_relu (ReLU)        (None, 14, 14, 512)       0         
_________________________________________________________________
conv_pw_7 (Conv2D)           (None, 14, 14, 512)       262144    
_________________________________________________________________
conv_pw_7_bn (BatchNormaliza (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_pw_7_relu (ReLU)        (None, 14, 14, 512)       0         
_________________________________________________________________
conv_dw_8 (DepthwiseConv2D)  (None, 14, 14, 512)       4608      
_________________________________________________________________
conv_dw_8_bn (BatchNormaliza (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_dw_8_relu (ReLU)        (None, 14, 14, 512)       0         
_________________________________________________________________
conv_pw_8 (Conv2D)           (None, 14, 14, 512)       262144    
_________________________________________________________________
conv_pw_8_bn (BatchNormaliza (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_pw_8_relu (ReLU)        (None, 14, 14, 512)       0         
_________________________________________________________________
conv_dw_9 (DepthwiseConv2D)  (None, 14, 14, 512)       4608      
_________________________________________________________________
conv_dw_9_bn (BatchNormaliza (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_dw_9_relu (ReLU)        (None, 14, 14, 512)       0         
_________________________________________________________________
conv_pw_9 (Conv2D)           (None, 14, 14, 512)       262144    
_________________________________________________________________
conv_pw_9_bn (BatchNormaliza (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_pw_9_relu (ReLU)        (None, 14, 14, 512)       0         
_________________________________________________________________
conv_dw_10 (DepthwiseConv2D) (None, 14, 14, 512)       4608      
_________________________________________________________________
conv_dw_10_bn (BatchNormaliz (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_dw_10_relu (ReLU)       (None, 14, 14, 512)       0         
_________________________________________________________________
conv_pw_10 (Conv2D)          (None, 14, 14, 512)       262144    
_________________________________________________________________
conv_pw_10_bn (BatchNormaliz (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_pw_10_relu (ReLU)       (None, 14, 14, 512)       0         
_________________________________________________________________
conv_dw_11 (DepthwiseConv2D) (None, 14, 14, 512)       4608      
_________________________________________________________________
conv_dw_11_bn (BatchNormaliz (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_dw_11_relu (ReLU)       (None, 14, 14, 512)       0         
_________________________________________________________________
conv_pw_11 (Conv2D)          (None, 14, 14, 512)       262144    
_________________________________________________________________
conv_pw_11_bn (BatchNormaliz (None, 14, 14, 512)       2048      
_________________________________________________________________
conv_pw_11_relu (ReLU)       (None, 14, 14, 512)       0         
_________________________________________________________________
conv_pad_12 (ZeroPadding2D)  (None, 15, 15, 512)       0         
_________________________________________________________________
conv_dw_12 (DepthwiseConv2D) (None, 7, 7, 512)         4608      
_________________________________________________________________
conv_dw_12_bn (BatchNormaliz (None, 7, 7, 512)         2048      
_________________________________________________________________
conv_dw_12_relu (ReLU)       (None, 7, 7, 512)         0         
_________________________________________________________________
conv_pw_12 (Conv2D)          (None, 7, 7, 1024)        524288    
_________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, 7, 7, 1024)        4096      
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 7, 7, 1024)        0         
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 1024)        9216      
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096      
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 7, 7, 1024)        0         
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 7, 7, 1024)        1048576   
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096      
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 7, 7, 1024)        0         
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1024)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800    
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328    
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 771       
=================================================================
Total params: 3,885,763
Trainable params: 656,899
Non-trainable params: 3,228,864
_________________________________________________________________
None
```

Loading our Monkey Breed Dataset

In [11]:
```python
from keras.preprocessing.image import ImageDataGenerator

train_data_dir = 'face_recognition/face_recognition/train/'
validation_data_dir = 'face_recognition/face_recognition/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 30 images belonging to 3 classes.
Found 20 images belonging to 3 classes.
```

Training out Model

- Note we're using checkpointing and early stopping

In [12]:
```python
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping


checkpoint = ModelCheckpoint("face_recognition_mobileNet.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 = 1097
nb_validation_samples = 272

# We only train 5 EPOCHS
epochs = 3
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 1/3
68/68 [==============================] - 374s 6s/step - loss: 0.2156 - accuracy: 0.9657 - val
_loss: 1.8606 - val_accuracy: 0.6500

Epoch 00001: val_loss improved from inf to 1.86064, saving model to face_recognition_mobileNe
t.h5
Epoch 2/3
31/68 [============>.................] - ETA: 2:37 - loss: 2.2037e-05 - accuracy: 1.0000

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-12-1440d4573dac> in <module>
     37     callbacks = callbacks,
     38     validation_data = validation_generator,
---> 39     validation_steps = nb_validation_samples // batch_size)

E:\python\python\lib\site-packages\keras\legacy\interfaces.py in wrapper(*args, **kwargs)
     89                 warnings.warn('Update your `' + object_name + '` call to the ' +
     90                               'Keras 2 API: ' + signature, stacklevel=2)
---> 91             return func(*args, **kwargs)
     92         wrapper._original_function = func
     93         return wrapper

E:\python\python\lib\site-packages\keras\engine\training.py in fit_generator(self, generator,
steps_per_epoch, epochs, verbose, callbacks, validation_data, validation_steps, validation_fr
eq, class_weight, max_queue_size, workers, use_multiprocessing, shuffle, initial_epoch)
   1730             use_multiprocessing=use_multiprocessing,
   1731             shuffle=shuffle,
-> 1732             initial_epoch=initial_epoch)
   1733 
   1734     @interfaces.legacy_generator_methods_support

E:\python\python\lib\site-packages\keras\engine\training_generator.py in fit_generator(model,
generator, steps_per_epoch, epochs, verbose, callbacks, validation_data, validation_steps, va
lidation_freq, class_weight, max_queue_size, workers, use_multiprocessing, shuffle, initial_e
poch)
    218                                             sample_weight=sample_weight,
    219                                             class_weight=class_weight,
--> 220                                             reset_metrics=False)
    221 
    222                 outs = to_list(outs)

E:\python\python\lib\site-packages\keras\engine\training.py in train_on_batch(self, x, y, sam
ple_weight, class_weight, reset_metrics)
   1512             ins = x + y + sample_weights
   1513         self._make_train_function()
-> 1514         outputs = self.train_function(ins)
   1515 
   1516         if reset_metrics:

E:\python\python\lib\site-packages\tensorflow_core\python\keras\backend.py in __call__(self,
 inputs)
   3725         value = math_ops.cast(value, tensor.dtype)
   3726       converted_inputs.append(value)
-> 3727     outputs = self._graph_fn(*converted_inputs)
   3728 
   3729     # EagerTensor.numpy() will often make a copy to ensure memory safety.

E:\python\python\lib\site-packages\tensorflow_core\python\eager\function.py in __call__(self,
*args, **kwargs)
   1549       TypeError: For invalid positional/keyword argument combinations.
   1550     """
-> 1551     return self._call_impl(args, kwargs)
   1552 
   1553   def _call_impl(self, args, kwargs, cancellation_manager=None):

E:\python\python\lib\site-packages\tensorflow_core\python\eager\function.py in _call_impl(sel
f, args, kwargs, cancellation_manager)
   1589       raise TypeError("Keyword arguments {} unknown. Expected {}.".format(
   1590           list(kwargs.keys()), list(self._arg_keywords)))
-> 1591     return self._call_flat(args, self.captured_inputs, cancellation_manager)
   1592 
   1593   def _filtered_call(self, args, kwargs):

E:\python\python\lib\site-packages\tensorflow_core\python\eager\function.py in _call_flat(sel
f, args, captured_inputs, cancellation_manager)
   1690       # No tape is watching; skip to running the function.
   1691       return self._build_call_outputs(self._inference_function.call(
-> 1692           ctx, args, cancellation_manager=cancellation_manager))
   1693     forward_backward = self._select_forward_and_backward_functions(
   1694         args,

E:\python\python\lib\site-packages\tensorflow_core\python\eager\function.py in call(self, ct
x, args, cancellation_manager)
    543               inputs=args,
    544               attrs=("executor_type", executor_type, "config_proto", config),
--> 545               ctx=ctx)
    546         else:
    547           outputs = execute.execute_with_cancellation(

E:\python\python\lib\site-packages\tensorflow_core\python\eager\execute.py in quick_execute(o
p_name, num_outputs, inputs, attrs, ctx, name)
     59     tensors = pywrap_tensorflow.TFE_Py_Execute(ctx._handle, device_name,
     60                                                op_name, inputs, attrs,
---> 61                                                num_outputs)
     62   except core._NotOkStatusException as e:
     63     if name is not None:

KeyboardInterrupt: 
```

Loading our classifer

In [13]:
```python
from keras.models import load_model

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

Testing our classifer on some test images

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

face_recognition_dict = {"[0]": "Abhinay",
                         "[1]": "Abhishek",
                         "[2]": "Vikas"}

face_recognition_dict_n = {"Abhinay": "Abhinay",
                      "Abhishek": "Abhishek",
                      "Vikas": "Vikas"}




def draw_test(name, pred, im):
    face = face_recognition_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, face, (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 - " + face_recognition_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(0,3):
    input_im = getRandomImage("face_recognition/face_recognition/validation/")
    input_original = input_im.copy()
    input_original = cv2.resize(input_original, None, fx=0.5, fy=0.5, interpolation = cv2.IN
TER_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(0)

cv2.destroyAllWindows()
```

```
Class - Vikas
```

In [ ]: