

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
7 ReLU False
8 Conv2D False
9 BatchNormalization False
10 ReLU False
11 ZeroPadding2D 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 atop 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 augmentation
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(
```

```
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 [=====] - 41s 3s/step - loss: 3.6638 - accuracy: 0.4784 - val_loss: 0.8307 - val_accu  
acy: 0.4231
```

```
Epoch 00001: val_loss improved from inf to 0.83072, saving model to human_dataset.h5
```

```
Epoch 2/5  
12/12 [=====] - 39s 3s/step - loss: 0.7442 - accuracy: 0.5247 - val_loss: 0.7975 - val_accu  
acy: 0.3846
```

```
Epoch 00002: val_loss improved from 0.83072 to 0.79748, saving model to human_dataset.h5
```

```
Epoch 3/5  
12/12 [=====] - 39s 3s/step - loss: 0.7469 - accuracy: 0.6451 - val_loss: 1.0525 - val_accu  
acy: 0.3846
```

```
Epoch 00003: val_loss did not improve from 0.79748
```

```
Epoch 4/5  
12/12 [=====] - 40s 3s/step - loss: 0.5180 - accuracy: 0.7562 - val_loss: 5.9635 - val_accu
```

Loading our classifier

```
In [1]: from keras.models import load_model  
  
classifier = load_model('human_dataset.h5')
```

Using TensorFlow backend.

Testing our classifier on some test images

```
In [2]: 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
    file_names = [f for f in listdir(file_path) if isfile(join(file_path, f))]
```



```

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

```