Transfer Learning using vgg16

```
from keras.applications import vgg16

# 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
vgg = vgg16.VGG16(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 vgg.layers:
    layer.trainable = False

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


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


from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, GlobalAveragePooling2D
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.layers import BatchNormalization
from keras.models import Model

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

FC_Head = lw(vgg, num_classes)

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

print(model.summary())
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
```

```
block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

global_average_pooling2d_2  (None, 512)               0
 (GlobalAveragePooling2D)

dense_8 (Dense)             (None, 1024)              525312

dense_9 (Dense)             (None, 1024)              1049600

dense_10 (Dense)            (None, 512)               524800

dense_11 (Dense)            (None, 2)                 1026

=================================================================
Total params: 16815426 (64.15 MB)
Trainable params: 2100738 (8.01 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
None
```

```
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).
```

```
from keras.preprocessing.image import ImageDataGenerator

train_data_dir = '/content/drive/My Drive/Ai_dataset//train'
validation_data_dir = '/content/drive/My Drive/Ai_dataset/validate'

# 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')
```

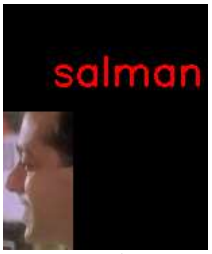
```
Found 539 images belonging to 2 classes.
Found 303 images belonging to 2 classes.
```

```
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping


checkpoint = ModelCheckpoint("face recognisation.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 = 108
nb_validation_samples = 52

# 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)
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers
<ipython-input-10-2587e57752a8>:33: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please
  history = model.fit_generator(
Epoch 1/5
6/6 [==============================] - ETA: 0s - loss: 1.4823 - accuracy: 0.4948
Epoch 1: val_loss improved from inf to 0.73278, saving model to face recognisation.h5
6/6 [==============================] - 47s 7s/step - loss: 1.4823 - accuracy: 0.4948 - val_loss: 0.7328 - val_accuracy: 0.4688
Epoch 2/5
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file vi
  saving_api.save_model(
6/6 [==============================] - ETA: 0s - loss: 0.7050 - accuracy: 0.4920
Epoch 2: val_loss did not improve from 0.73278
6/6 [==============================] - 37s 7s/step - loss: 0.7050 - accuracy: 0.4920 - val_loss: 0.7668 - val_accuracy: 0.4479
Epoch 3/5
6/6 [==============================] - ETA: 0s - loss: 0.6865 - accuracy: 0.5365
Epoch 3: val_loss improved from 0.73278 to 0.69035, saving model to face recognisation.h5
6/6 [==============================] - 16s 3s/step - loss: 0.6865 - accuracy: 0.5365 - val_loss: 0.6903 - val_accuracy: 0.4688
Epoch 4/5
6/6 [==============================] - ETA: 0s - loss: 0.7033 - accuracy: 0.4740
Epoch 4: val_loss did not improve from 0.69035
6/6 [==============================] - 10s 2s/step - loss: 0.7033 - accuracy: 0.4740 - val_loss: 0.7093 - val_accuracy: 0.5104
Epoch 5/5
6/6 [==============================] - ETA: 0s - loss: 0.6936 - accuracy: 0.5260
Epoch 5: val_loss did not improve from 0.69035
6/6 [==============================] - 9s 2s/step - loss: 0.6936 - accuracy: 0.5260 - val_loss: 0.6961 - val_accuracy: 0.5104
```

```
from keras.models import load_model

classifier = load_model('/content/face recognisation.h5')

import os
import cv2
from google.colab.patches import cv2_imshow
import numpy as np
from os import listdir
from os.path import isfile, join

monkey_breeds_dict = {"[0]": " salman ",
                      "[1]": "aamir "}

monkey_breeds_dict_n = {"n0": " salman",
                      "n1": "aamir "}

def draw_test(name, pred, im):
    monkey = monkey_breeds_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, monkey, (20, 60) , cv2.FONT_HERSHEY_SIMPLEX,1, (0,0,255), 2)
    cv2_imshow(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 - " + monkey_breeds_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("/content/drive/My Drive/Ai_dataset/validate/")
    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(0)

cv2.destroyAllWindows()
```

Class - salman

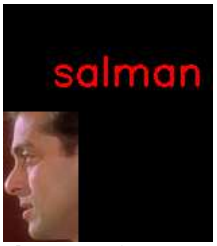


Class - aamir

Class - salman

Class - aamir

Class - salman



Class - aamir

Class - aamir