Transfer learning

Reference:

https://www.learnopencv.com/keras-tutorial-fine-tuning-using-pre-trained-models/ (https://www.learnopencv.com/keras-tutorial-fine-tuning-using-pre-trained-models/)

Build a base model

In [1]:

```
from keras.applications import VGG16
from keras import models
from keras import layers
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
from keras.preprocessing import image   # for load_image

vgg_conv = VGG16(weights='imagenet',
                  include_top=False,
                  input_shape=(224, 224, 3))
```

```
Using TensorFlow backend.
```

Freeze layers from pretrained model

In [2]:

```
# Freeze the layers except the last 4 layers
for layer in vgg_conv.layers[:-4]:
    layer.trainable = False

# Check the trainable status of the individual layers
for layer in vgg_conv.layers:
    print(layer, layer.trainable)
```

```
<keras.engine.input_layer.InputLayer object at 0x000002B129C975F8> False
<keras.layers.convolutional.Conv2D object at 0x000002B13039FEF0> False
<keras.layers.convolutional.Conv2D object at 0x000002B1302F4128> False
<keras.layers.pooling.MaxPooling2D object at 0x000002B1303F5630> False
<keras.layers.convolutional.Conv2D object at 0x000002B1303D0320> False
<keras.layers.convolutional.Conv2D object at 0x000002B1303D0390> False
<keras.layers.pooling.MaxPooling2D object at 0x000002B130422358> False
<keras.layers.convolutional.Conv2D object at 0x000002B130431128> False
<keras.layers.convolutional.Conv2D object at 0x000002B13045A748> False
<keras.layers.convolutional.Conv2D object at 0x000002B130443DD8> False
<keras.layers.pooling.MaxPooling2D object at 0x000002B13046FCF8> False
<keras.layers.convolutional.Conv2D object at 0x000002B130499EF0> False
<keras.layers.convolutional.Conv2D object at 0x000002B1304BEF28> False
<keras.layers.convolutional.Conv2D object at 0x000002B1304AE898> False
<keras.layers.pooling.MaxPooling2D object at 0x000002B1304CFD30> False
<keras.layers.convolutional.Conv2D object at 0x000002B1304FC860> True
<keras.layers.convolutional.Conv2D object at 0x000002B130522828> True
<keras.layers.convolutional.Conv2D object at 0x000002B130514AC8> True
<keras.layers.pooling.MaxPooling2D object at 0x000002B130549C50> True
```

Create a new model

In [3]:

```python
# from keras import models
# from keras import layers
# from keras import optimizers

targetClassNumber = 11

# Create the model
model = models.Sequential()

# Add the vgg convolutional base model
model.add(vgg_conv)

# Add new layers
model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(targetClassNumber, activation='softmax'))

# Show a summary of the model. Check the number of trainable parameters
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688  
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              25691136  
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 11)                11275     
=================================================================
Total params: 40,417,099
Trainable params: 32,781,835
Non-trainable params: 7,635,264
_________________________________________________________________
```

Setup the data generators

In [4]:

```
train_dir = './train'
validation_dir = './validation'
image_size = 224

# nTrain = 600
# nVal = 150

# from keras.preprocessing.image import ImageDataGenerator
# import numpy as np
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=20,
      width_shift_range=0.2,
      height_shift_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale=1./255)

# Change the batchsize according to your system RAM
train_batchsize = 20
val_batchsize = 20

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(image_size, image_size),
        batch_size=train_batchsize,
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
        validation_dir,
        target_size=(image_size, image_size),
        batch_size=val_batchsize,
        class_mode='categorical',
        shuffle=False)
```

```
Found 33 images belonging to 11 classes.
Found 22 images belonging to 11 classes.
```

Train the model

In [5]:

```
# Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])
# Train the model
history = model.fit_generator(
      train_generator,
      steps_per_epoch=train_generator.samples/train_generator.batch_size ,
      epochs=24,
      validation_data=validation_generator,
      validation_steps=validation_generator.samples/validation_generator.batch_siz
      verbose=1)

# Save the model
model.save('faceID_VGG16.h5')
```

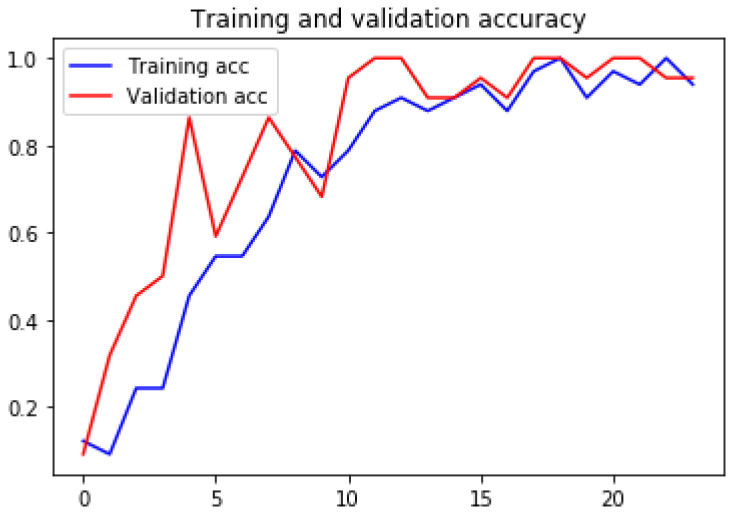
```
Epoch 1/24
2/1 [===================================] - 8s 4s/step - loss: 3.2833 - acc:
0.1340 - val_loss: 2.5770 - val_acc: 0.0909
Epoch 2/24
2/1 [====================================] - 2s 881ms/step - loss: 2.8598 - ac
c: 0.0983 - val_loss: 2.1225 - val_acc: 0.3182
Epoch 3/24
2/1 [====================================] - 2s 889ms/step - loss: 2.3815 - ac
c: 0.2411 - val_loss: 1.8904 - val_acc: 0.4545
Epoch 4/24
2/1 [====================================] - 2s 875ms/step - loss: 2.2975 - ac
c: 0.2411 - val_loss: 1.4564 - val_acc: 0.5000
Epoch 5/24
2/1 [===================================] - 2s 879ms/step - loss: 1.9713 - ac
c: 0.4374 - val_loss: 1.0498 - val_acc: 0.8636
Epoch 6/24
2/1 [====================================] - 2s 881ms/step - loss: 1.4149 - ac
c: 0.5357 - val_loss: 1.3710 - val_acc: 0.5909
Epoch 7/24
2/1 [====================================] - 2s 900ms/step - loss: 1.3616 - ac
c: 0.5357 - val_loss: 0.7851 - val_acc: 0.7273
Epoch 8/24
2/1 [====================================] - 2s 890ms/step - loss: 0.9548 - ac
c: 0.6519 - val_loss: 0.6277 - val_acc: 0.8636
Epoch 9/24
2/1 [===================================] - 2s 882ms/step - loss: 0.8553 - ac
c: 0.7857 - val_loss: 0.6197 - val_acc: 0.7727
Epoch 10/24
2/1 [===================================] - 2s 885ms/step - loss: 0.8097 - ac
c: 0.7232 - val_loss: 0.9519 - val_acc: 0.6818
Epoch 11/24
2/1 [===================================] - 2s 882ms/step - loss: 0.7908 - ac
c: 0.7947 - val_loss: 0.3036 - val_acc: 0.9545
Epoch 12/24
2/1 [====================================] - 2s 885ms/step - loss: 0.4097 - ac
c: 0.8750 - val_loss: 0.1982 - val_acc: 1.0000
Epoch 13/24
2/1 [====================================] - 2s 882ms/step - loss: 0.3287 - ac
c: 0.9197 - val_loss: 0.2137 - val_acc: 1.0000
Epoch 14/24
```

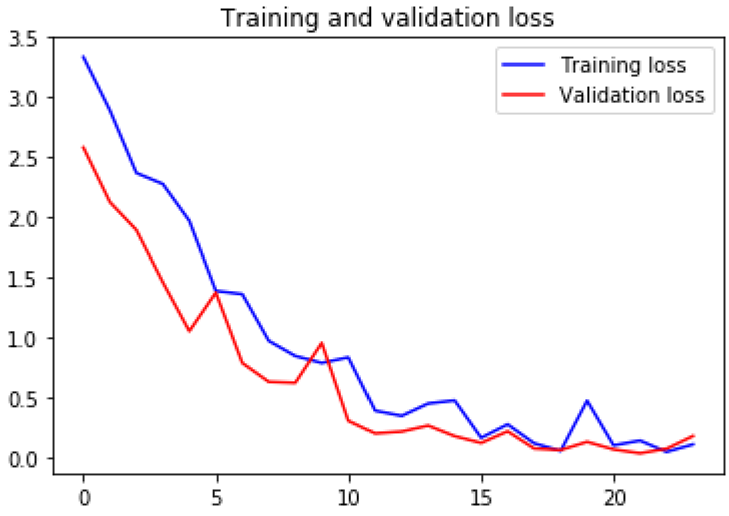
```
2/1 [====================================] - 2s 884ms/step - loss: 0.4788 - ac
c: 0.8660 - val_loss: 0.2635 - val_acc: 0.9091
Epoch 15/24
2/1 [====================================] - 2s 886ms/step - loss: 0.4345 - ac
c: 0.9197 - val_loss: 0.1753 - val_acc: 0.9091
Epoch 16/24
2/1 [====================================] - 2s 886ms/step - loss: 0.1585 - ac
c: 0.9375 - val_loss: 0.1188 - val_acc: 0.9545
Epoch 17/24
2/1 [====================================] - 2s 882ms/step - loss: 0.2798 - ac
c: 0.8750 - val_loss: 0.2156 - val_acc: 0.9091
Epoch 18/24
2/1 [====================================] - 2s 885ms/step - loss: 0.1135 - ac
c: 0.9732 - val_loss: 0.0724 - val_acc: 1.0000
Epoch 19/24
2/1 [====================================] - 2s 888ms/step - loss: 0.0536 - ac
c: 1.0000 - val_loss: 0.0597 - val_acc: 1.0000
Epoch 20/24
2/1 [====================================] - 2s 882ms/step - loss: 0.4554 - ac
c: 0.9107 - val_loss: 0.1277 - val_acc: 0.9545
Epoch 21/24
2/1 [====================================] - 2s 883ms/step - loss: 0.0906 - ac
c: 0.9732 - val_loss: 0.0648 - val_acc: 1.0000
Epoch 22/24
2/1 [====================================] - 2s 887ms/step - loss: 0.1309 - ac
c: 0.9465 - val_loss: 0.0325 - val_acc: 1.0000
Epoch 23/24
2/1 [====================================] - 2s 883ms/step - loss: 0.0432 - ac
c: 1.0000 - val_loss: 0.0703 - val_acc: 0.9545
Epoch 24/24
2/1 [====================================] - 2s 881ms/step - loss: 0.0984 - ac
c: 0.9465 - val_loss: 0.1764 - val_acc: 0.9545
```

Check Performance

In [6]:

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

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'b', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





In [7]:

```
# from keras.preprocessing import image   # for load_image

# Create a generator for prediction
validation_generator = validation_datagen.flow_from_directory(
        validation_dir,
        target_size=(image_size, image_size),
        batch_size=val_batchsize,
        class_mode='categorical',
        shuffle=False)

# Get the filenames from the generator
fnames = validation_generator.filenames

# Get the ground truth from generator
ground_truth = validation_generator.classes

# Get the label to class mapping from the generator
label2index = validation_generator.class_indices

# Getting the mapping from class index to class label
idx2label = dict((v,k) for k,v in label2index.items())

# Get the predictions from the model using the generator
predictions = model.predict_generator(validation_generator,
                                      steps=validation_generator.samples/validatio
                                      verbose=1)
predicted_classes = np.argmax(predictions,axis=1)

errors = np.where(predicted_classes != ground_truth)[0]
print("No of errors = {}/{}".format(len(errors),validation_generator.samples))

# Show the errors
for i in range(len(errors)):
    pred_class = np.argmax(predictions[errors[i]])
    pred_label = idx2label[pred_class]

    title = 'Original label:{}, Prediction :{}, confidence : {:.3f}'.format(
        fnames[errors[i]].split('/')[0],
        pred_label,
        predictions[errors[i]][pred_class])

    original = image.load_img('{}/{}'.format(validation_dir,fnames[errors[i]]))
    plt.figure(figsize=[7,7])
    plt.axis('off')
    plt.title(title)
    plt.imshow(original)
    plt.show()
```

```
Found 22 images belonging to 11 classes.
2/1 [======================================================] - 1s 401ms/step
No of errors = 1/22
```

Original label:albert\aa4.PNG, Prediction :Frank, confidence : 0.581

```
In [ ]:
```