# Transfer learning

Reference:

Keras Documentation: https://keras.io/applications/#usage-examples-for-image-classification-models (https://keras.io/applications/#usage-examples-for-image-classification-models)

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

Environment:

- Keras-gpu 2.2.0
- matplotlib 2.2.2
- pillow 5.1.0

# Build a base model

In [1]:
```
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing import image
# from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras import backend as K
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


# create the base pre-trained model
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(224,
```

```
Using TensorFlow backend.
```

# Create a new model

In [2]:
```
# 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(base_model)

# 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'))
```

# Freeze layers from pretrained model

In [3]:
```
# Freeze the layers except the last 4 layers
for layer in model.layers[:-4]:
    layer.trainable = False

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

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 5, 5, 2048)        21802784
_________________________________________________________________
flatten_1 (Flatten)          (None, 51200)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              52429824
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 11)                11275
=================================================================
Total params: 74,243,883
Trainable params: 52,441,099
Non-trainable params: 21,802,784
_________________________________________________________________
```

# 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 = 33
val_batchsize = 22

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'])

# we need to recompile the model for these modifications to take effect
# we use SGD with a low learning rate
# from keras.optimizers import SGD
# model.compile(loss='categorical_crossentropy',
#               optimizer=SGD(lr=0.0001, momentum=0.9),
#               metrics=['acc'])


# Train the model
history = model.fit_generator(
      train_generator,
      steps_per_epoch=train_generator.samples/train_generator.batch_size ,
      epochs=40,
      validation_data=validation_generator,
      validation_steps=validation_generator.samples/validation_generator.batch_siz
      verbose=1)

# Save the model
model.save('faceID_InceptionV3_gpu.h5')
```

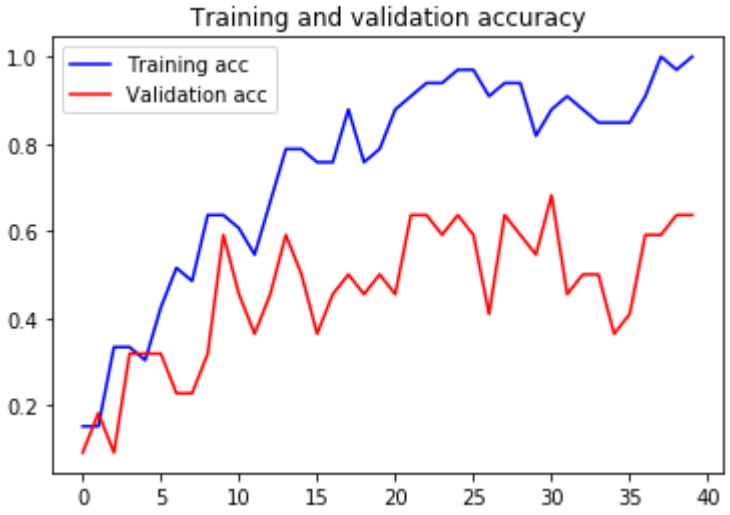
```
Epoch 1/40
1/1 [==============================] - 10s 10s/step - loss: 2.9864 - acc: 0.15
15 - val_loss: 9.4325 - val_acc: 0.0909
Epoch 2/40
1/1 [==============================] - 1s 1s/step - loss: 7.5655 - acc: 0.1515
- val_loss: 8.7859 - val_acc: 0.1818
Epoch 3/40
1/1 [==============================] - 1s 1s/step - loss: 6.6334 - acc: 0.3333
- val_loss: 7.2885 - val_acc: 0.0909
Epoch 4/40
1/1 [==============================] - 1s 1s/step - loss: 5.6504 - acc: 0.3333
- val_loss: 7.9209 - val_acc: 0.3182
Epoch 5/40
1/1 [==============================] - 1s 1s/step - loss: 5.8875 - acc: 0.3030
- val_loss: 5.9943 - val_acc: 0.3182
Epoch 6/40
1/1 [==============================] - 1s 1s/step - loss: 4.3802 - acc: 0.4242
- val_loss: 6.3837 - val_acc: 0.3182
Epoch 7/40
1/1 [==============================] - 1s 994ms/step - loss: 3.9174 - acc: 0.5
152 - val_loss: 7.8179 - val_acc: 0.2273
Epoch 8/40
1/1 [==============================] - 1s 1s/step - loss: 3.7902 - acc: 0.4848
- val_loss: 6.7774 - val_acc: 0.2273
Epoch 9/40
1/1 [==============================] - 1s 1s/step - loss: 2.2917 - acc: 0.6364
- val_loss: 6.4406 - val_acc: 0.3182
Epoch 10/40
1/1 [==============================] - 1s 1s/step - loss: 2.5741 - acc: 0.6364
- val_loss: 4.1383 - val_acc: 0.5909
Epoch 11/40
```

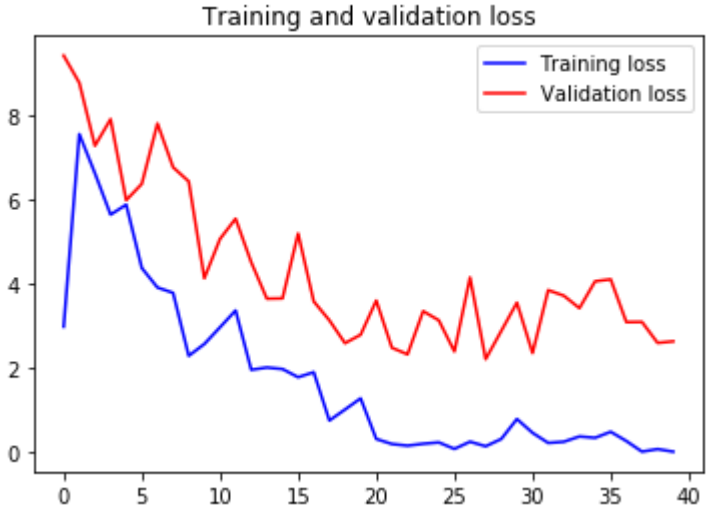
```
1/1 [==============================] - 1s 1s/step - loss: 2.9706 - acc: 0.6061
- val_loss: 5.0725 - val_acc: 0.4545
Epoch 12/40
1/1 [==============================] - 1s 1s/step - loss: 3.3723 - acc: 0.5455
- val_loss: 5.5546 - val_acc: 0.3636
Epoch 13/40
1/1 [==============================] - 1s 1s/step - loss: 1.9605 - acc: 0.6667
- val_loss: 4.5175 - val_acc: 0.4545
Epoch 14/40
1/1 [==============================] - 1s 1s/step - loss: 2.0145 - acc: 0.7879
- val_loss: 3.6515 - val_acc: 0.5909
Epoch 15/40
1/1 [==============================] - 1s 990ms/step - loss: 1.9785 - acc: 0.7
879 - val_loss: 3.6582 - val_acc: 0.5000
Epoch 16/40
1/1 [==============================] - 1s 1s/step - loss: 1.7841 - acc: 0.7576
- val_loss: 5.2007 - val_acc: 0.3636
Epoch 17/40
1/1 [==============================] - 1s 1s/step - loss: 1.9025 - acc: 0.7576
- val_loss: 3.5860 - val_acc: 0.4545
Epoch 18/40
1/1 [==============================] - 1s 1s/step - loss: 0.7544 - acc: 0.8788
- val_loss: 3.1374 - val_acc: 0.5000
Epoch 19/40
1/1 [==============================] - 1s 1s/step - loss: 1.0159 - acc: 0.7576
- val_loss: 2.5933 - val_acc: 0.4545
Epoch 20/40
1/1 [==============================] - 1s 1s/step - loss: 1.2811 - acc: 0.7879
- val_loss: 2.7944 - val_acc: 0.5000
Epoch 21/40
1/1 [==============================] - 1s 1s/step - loss: 0.3153 - acc: 0.8788
- val_loss: 3.6067 - val_acc: 0.4545
Epoch 22/40
1/1 [==============================] - 1s 1s/step - loss: 0.1956 - acc: 0.9091
- val_loss: 2.4848 - val_acc: 0.6364
Epoch 23/40
1/1 [==============================] - 1s 1s/step - loss: 0.1562 - acc: 0.9394
- val_loss: 2.3280 - val_acc: 0.6364
Epoch 24/40
1/1 [==============================] - 1s 1s/step - loss: 0.2011 - acc: 0.9394
- val_loss: 3.3546 - val_acc: 0.5909
Epoch 25/40
1/1 [==============================] - 1s 1s/step - loss: 0.2336 - acc: 0.9697
- val_loss: 3.1373 - val_acc: 0.6364
Epoch 26/40
1/1 [==============================] - 1s 994ms/step - loss: 0.0793 - acc: 0.9
697 - val_loss: 2.3966 - val_acc: 0.5909
Epoch 27/40
1/1 [==============================] - 1s 1s/step - loss: 0.2503 - acc: 0.9091
- val_loss: 4.1566 - val_acc: 0.4091
Epoch 28/40
1/1 [==============================] - 1s 1s/step - loss: 0.1418 - acc: 0.9394
- val_loss: 2.2141 - val_acc: 0.6364
Epoch 29/40
1/1 [==============================] - 1s 995ms/step - loss: 0.3156 - acc: 0.9
394 - val_loss: 2.8861 - val_acc: 0.5909
Epoch 30/40
```

```
1/1 [==============================] - 1s 1s/step - loss: 0.7897 - acc: 0.8182
- val_loss: 3.5567 - val_acc: 0.5455
Epoch 31/40
1/1 [==============================] - 1s 977ms/step - loss: 0.4613 - acc: 0.8
788 - val_loss: 2.3595 - val_acc: 0.6818
Epoch 32/40
1/1 [==============================] - 1s 1s/step - loss: 0.2216 - acc: 0.9091
- val_loss: 3.8548 - val_acc: 0.4545
Epoch 33/40
1/1 [==============================] - 1s 1s/step - loss: 0.2488 - acc: 0.8788
- val_loss: 3.7229 - val_acc: 0.5000
Epoch 34/40
1/1 [==============================] - 1s 1s/step - loss: 0.3749 - acc: 0.8485
- val_loss: 3.4263 - val_acc: 0.5000
Epoch 35/40
1/1 [==============================] - 1s 1s/step - loss: 0.3426 - acc: 0.8485
- val_loss: 4.0650 - val_acc: 0.3636
Epoch 36/40
1/1 [==============================] - 1s 1s/step - loss: 0.4862 - acc: 0.8485
- val_loss: 4.1119 - val_acc: 0.4091
Epoch 37/40
1/1 [==============================] - 1s 1s/step - loss: 0.2665 - acc: 0.9091
- val_loss: 3.0951 - val_acc: 0.5909
Epoch 38/40
1/1 [==============================] - 1s 973ms/step - loss: 0.0142 - acc: 1.0
000 - val_loss: 3.0988 - val_acc: 0.5909
Epoch 39/40
1/1 [==============================] - 1s 1s/step - loss: 0.0744 - acc: 0.9697
- val_loss: 2.6003 - val_acc: 0.6364
Epoch 40/40
1/1 [==============================] - 1s 1s/step - loss: 0.0129 - acc: 1.0000
- val_loss: 2.6367 - val_acc: 0.6364
```

# 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()
```



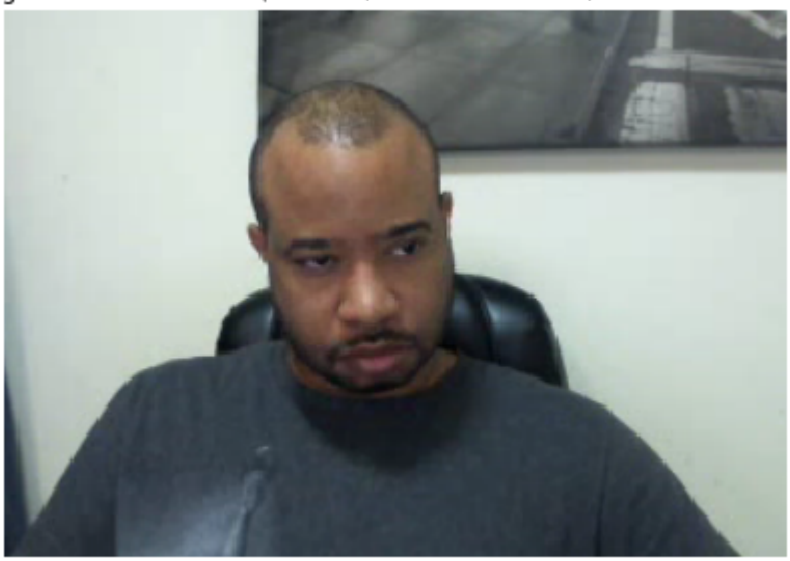



In [7]:
```
# from keras.preprocessing import image   # for load_image

# 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_gener
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()
```

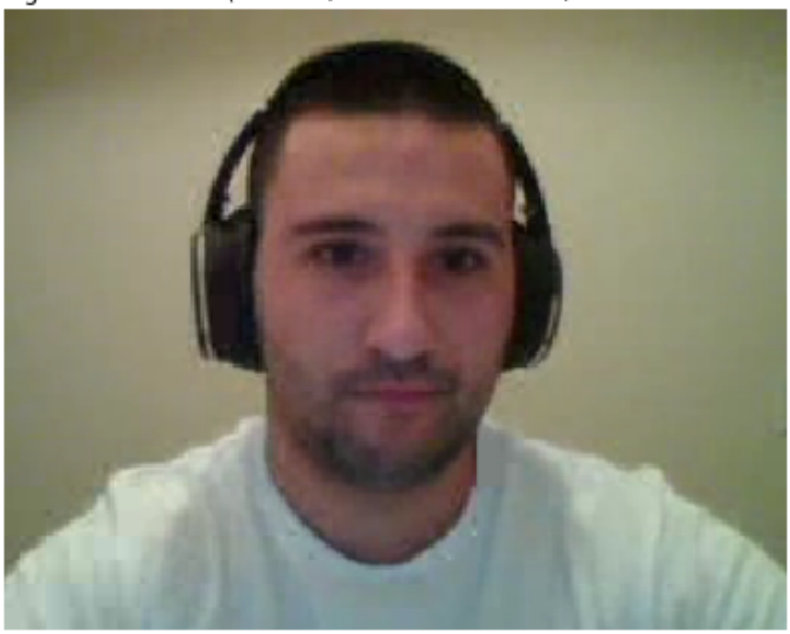
```
Found 22 images belonging to 11 classes.
1/1 [==============================] - 2s 2s/step
No of errors = 8/22
```

Original label:Damarcus\dt1.PNG, Prediction :Lokesh, confidence : 0.999

Original label:Damarcus\dt3.PNG, Prediction :albert, confidence : 0.971



Original label:Frank\fs1.PNG, Prediction :Lokesh, confidence : 0.478

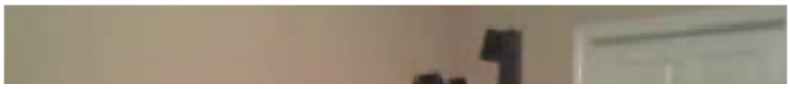


Original label:Frank\fs3.PNG, Prediction :Lokesh, confidence : 0.548

Original label:Misael\ms3.PNG, Prediction :albert, confidence : 0.424

Original label:albert\aa4.PNG, Prediction :Lokesh, confidence : 0.583

Original label:brian\bl.PNG, Prediction :Lokesh, confidence : 0.467



Original label:mooyoung-lee\lee.PNG, Prediction :Lokesh, confidence : 0.875

In [ ]: