Six key points for effective transfer learning

I would like to introduce the key points to build your own model based on MobileNet. The model is lightweight and fast but it has enough performance.

Import libraries

Import the liblraries as usual.

In [22]:

```
import os
import time
import math

import cv2
import coremltools
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

from keras import optimizers
from keras.layers import Input, Flatten, Dense, Dropout, GlobalAveragePooling2
D, BatchNormalization
from keras.models import Model
from keras.utils.generic_utils import CustomObjectScope
from keras.applications import mobilenet
from keras.applications.mobilenet import MobileNet
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

Setting

I assign 25% of training dataset as validation dataset this time.

In [23]:

```
movie_path = '../dataset/movies'
train_data_dir = '../dataset/training'
result_dir = '../result'
frame_interval = 3
batch_size = 64
epoch_count = 100
validation_data_split_rate = 0.25
```

Class Labels

I load the class labels from the "classes.txt" file, and then setup a list with removing the line separator.

In [24]:

```
# Set up the class labels
with open('classes.txt', 'r') as fp:
    lines = fp.readlines()
class_labels = []
for line in lines:
# The line separator will be; Win='\r\n', Mac='\r', Linux='\n'
    line = line.rstrip('\n')
    class_labels.append(line)
```

Key Point 1: Create training image and annotation automatically

I am going to pick out a frame every interval specified in "frame_interval". The destination directory name is same as the name of movie file and it is the class label. For instance, if you have "deep_learning.mov", "deep_learning" directory will be made under the "../dataset/training" directory and the name of the image file will be set from 0.jpg to nnn.jpg.

In [25]:

```
# Loop until the end of the file list under the movie directory.
file_list = [f for f in os.listdir(movie_path) if os.path.isfile(os.path.join(
movie_path, f)) and  f.endswith(".mov")]
for filename in file_list:
    source_file = os.path.join(movie_path, filename)
    destination_dir = os.path.join(train_data_dir, os.path.splitext(filename)[
0])

    # Make destination directory if not exist
    if not os.path.exists(destination_dir):
        os.mkdir(destination_dir)

    # Open video file to capture the jpeg images
    capture = cv2.VideoCapture(source_file)

    img_count = 0
    frame_count = 0

    # Loop until  EOF
    while(capture.isOpened()):

        # Read one video frame
        ret, frame = capture.read()
        if ret == False:
            break

        # Pick up the frame if its position is the specified frame interval
        if frame_count % frame_interval == 0:
            img_file_name = os.path.join(destination_dir, str(img_count) + '.j
pg')
            cv2.imwrite(img_file_name, frame)
            img_count += 1
        frame_count += 1
    # Close the current movie file
    capture.release()
    print('# of images & path:', img_count, 'images under' + destination_dir)
```

```
# of images & path: 59 images under../dataset/training/five
# of images & path: 56 images under../dataset/training/one
# of images & path: 61 images under../dataset/training/three
# of images & path: 56 images under../dataset/training/two
# of images & path: 59 images under../dataset/training/four
```

Key Point 2: Configure output layers to meet your task

I configure my own output layers to connect to the last layer of the base model. I have several options to configure this layers but I should remember the MobileNet is lightweight and fast model.

1. Which do I use Flatten or GlobalAveragePooling2D as the input layer of my model?
2. Which do I use relu or tanh as an activation as the fully connected layer?
3. Do I use batch normarization?
4. Do I use Dropout? What ratio do I set?

Very large neural network like ResNet50 or VGG16 places the Flatten as the input layer of my own output layers but I do not think MobileNet should have the same function there since it has fully connected Dense layer next to the input layer. The parameter size that Flatten layer passes to the Dense layer will be 50176 (7 x 7 x 1024), and it makes the size of model around 115MB even though it is only 15MB if I place the GlobalAveragePooling2D function there.

This configuration is just an example but my model shows good performance to resolve my task. You will need trial and error to understand whether BatchNormarization and Dropout works for your task as you expected or not. It will take longer time to try several conbination of configuration and parameters but you need to do that to get better model for your computer vision task.

In [26]:

```
input_tensor = Input(shape=(224, 224, 3))
base_model = MobileNet(include_top=False, weights='imagenet', input_tensor=inp
ut_tensor)

# Make our own output layers
x = base_model.output
# Flatten layer is not suitable for MobileNet based model as it makes file siz
e larger
# x = Flatten(input_shape=base_model.output_shape[1:])(x)
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu', name='fc1')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
output_tensor = Dense(len(class_labels), activation='softmax', name='predictio
ns')(x)

# Connect base model to our own output layers
mymodel = Model(inputs=base_model.input, outputs=output_tensor)
```

Key Point 3: Train batch normarization layers in the base model

MobileNet and ResNet50 has batch normarization layers and I need to update these layers even though I want to use transfer learning without updating any layers of the base model. The batch normarization layer passes the output data to the next layer with normarizing the batch input data to avoid overfitting.

However, I have never seen that the traing loss getting converge if I set the trainable flag of the batch normarization layers to false since the layers no longer calcurate the average, standard deviation or variance. The layers use the original value to normarize the new training data.

In [27]:

```
# Set the trainable flag to true if the layer is batch normalization layer.
freeze_layer_counts = 0
for layer in mymodel.layers[:len(base_model.layers)]:

    # MobileNet need to update its batch nomalization layer to converge the lo
ss.
    if layer.name.endswith('_bn'):
        # You can understand the training loss never converge when you set thi
s to false.
        layer.trainable = True
    else:
        freeze_layer_counts = freeze_layer_counts + 1
        layer.trainable = False

# I need to update the paraemters for my additional layers.
for layer in mymodel.layers[len(base_model.layers):]:
    layer.trainable = True

print('Model Structure:')
mymodel.summary()
print('Total Layers:  ' + str(len(mymodel.layers)))
print('Freezed Layers:' + str(freeze_layer_counts))
```

```
Model Structure:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 226, 226, 3)       0
_________________________________________________________________
conv1 (Conv2D)               (None, 112, 112, 32)      864
_________________________________________________________________
conv1_bn (BatchNormalization (None, 112, 112, 32)      128
_________________________________________________________________
conv1_relu (Activation)      (None, 112, 112, 32)      0
_________________________________________________________________
conv_pad_1 (ZeroPadding2D)   (None, 114, 114, 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 (Activation)	(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 (Activation)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 114, 114, 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 (Activation)	(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 (Activation)	(None, 56, 56, 128)	0
conv_pad_3 (ZeroPadding2D)	(None, 58, 58, 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 (Activation)	(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 (Activation)	(None, 56, 56, 128)	0
conv_pad_4 (ZeroPadding2D)	(None, 58, 58, 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 (Activation)	(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 (Activation)	(None, 28, 28, 256)	0
conv_pad_5 (ZeroPadding2D)	(None, 30, 30, 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 (Activation)	(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 (Activation)	(None, 28, 28, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, 30, 30, 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 (Activation)	(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 (Activation)	(None, 14, 14, 512)	0
conv_pad_7 (ZeroPadding2D)	(None, 16, 16, 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 (Activation)	(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 (Activation)	(None, 14, 14, 512)	0
conv_pad_8 (ZeroPadding2D)	(None, 16, 16, 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 (Activation)	(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 (Activation)	(None, 14, 14, 512)	0
conv_pad_9 (ZeroPadding2D)	(None, 16, 16, 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 (Activation)	(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 (Activation)	(None, 14, 14, 512)	0
conv_pad_10 (ZeroPadding2D)	(None, 16, 16, 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 (Activation)	(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 (Activation)	(None, 14, 14, 512)	0
conv_pad_11 (ZeroPadding2D)	(None, 16, 16, 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 (Activation)	(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 (Activation)	(None, 14, 14, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 16, 16, 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 (Activation)	(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 (Activation)	(None, 7, 7, 1024)	0
conv_pad_13 (ZeroPadding2D)	(None, 9, 9, 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 (Activation) (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 (Activation) (None, 7, 7, 1024)        0         
_________________________________________________________________
global_average_pooling2d_3 ( (None, 1024)              0         
_________________________________________________________________
fc1 (Dense)                  (None, 256)               262400    
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024      
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
predictions (Dense)          (None, 5)                 1285      
=================================================================
Total params: 3,493,573
Trainable params: 286,085
Non-trainable params: 3,207,488
_________________________________________________________________
Total Layers:  101
Freezed Layers:69
```

Key Point 4: Use Keras augmentation to avoid overfitting

I have very limited volume of training dataset. It is less than 300 images for 5 classes. Basically it is quite small volume to learn, and the small dataset often causes the overfitting issue as you know. However, Keras has very strong data augmentation features. I can generate addtional images from my own training dataset. I can train the model with enough volume of the training dataset to avoid overfitting.

Refer to [Image Preprocessing - ImageDataGenerator class (https://keras.io/preprocessing/image/)](https://keras.io/preprocessing/image/) for more attribute of the ImageDataGenerator.

In [28]:

```
train_data_generator = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=45,
    zoom_range=0.3,
    brightness_range=[0.1, 0.9],
    channel_shift_range=50.0,
    width_shift_range=0.2,
    height_shift_range=0.2,
    validation_split=validation_data_split_rate)

train_generator = train_data_generator.flow_from_directory(
    train_data_dir,
    target_size=(224, 224),
    color_mode='rgb',
    classes=class_labels,
    class_mode='categorical',
    batch_size=batch_size,
    subset='training',
    shuffle=True)

validation_generator = train_data_generator.flow_from_directory(
    train_data_dir,
    target_size=(224, 224),
    color_mode='rgb',
    classes=class_labels,
    class_mode='categorical',
    batch_size=batch_size,
    subset='validation',
    shuffle=True)
```

```
Found 220 images belonging to 5 classes.
Found 71 images belonging to 5 classes.
```

Key Point 5: Save the best performance model

The last epoch does not always give me the best performance model. I often found the best performance model is generated on the way to the last epoch. I will miss the best model if I only save the model after the last epoch has ended. So, I save the model when it update its best "loss" at the end of each epoch. I save the model not only configuration but also its weight.

In addtion, I tried early stopping but I commented out the code as it does not works for my model. My model sometimes get worth performance than previous epoch but a few epochs later it start improving the performance again.

In [29]:

```
print('Start training:')

# Compile the model
mymodel.compile(loss='categorical_crossentropy', optimizer=optimizers.SGD(
    lr=1e-3, momentum=0.9), metrics=['accuracy'])

# Save the model if it has the best loss after an epoch training.
checkpoint_cb = ModelCheckpoint(filepath=os.path.join(result_dir, 'mobilenet.h
5'),
                                monitor='loss',
                                verbose=1,
                                save_best_only=True,
                                save_weights_only=False,
                                mode='auto',
                                period=1)

# Stop earlier when the training has stopped improving val_loss.
# earystop_cb = EarlyStopping(monitor='val_loss',
#                                 patience=0,
#                                 verbose=1,
#                                 mode='auto')

# Transfer learning start
start = time.time()
history = mymodel.fit_generator(
    train_generator,
    steps_per_epoch=math.ceil(train_generator.samples / batch_size),
    epochs=epoch_count,
    callbacks=[checkpoint_cb],
    validation_data = validation_generator,
    validation_steps=math.ceil(validation_generator.samples / batch_size))

# End learning
process_time = (time.time() - start) / 60
print('Learning Time: ', '{:.2f}'.format(process_time), 'min.')
```

```
Start training:
Epoch 1/100
4/4 [==============================] - 128s 32s/step - loss: 2.683
8 - acc: 0.2460 - val_loss: 2.2330 - val_acc: 0.1831

Epoch 00001: loss improved from inf to 2.63226, saving model to ..
/result/mobilenet.h5
Epoch 2/100
4/4 [==============================] - 132s 33s/step - loss: 2.547
6 - acc: 0.1653 - val_loss: 2.3284 - val_acc: 0.1690

Epoch 00002: loss improved from 2.63226 to 2.48569, saving model t
o ../result/mobilenet.h5
Epoch 3/100
4/4 [==============================] - 130s 32s/step - loss: 2.354
1 - acc: 0.2702 - val_loss: 1.7162 - val_acc: 0.3521

Epoch 00003: loss improved from 2.48569 to 2.36953, saving model t
```

```
o ../result/mobilenet.h5
Epoch 4/100
4/4 [==============================] - 124s 31s/step - loss: 1.958
9 - acc: 0.3669 - val_loss: 1.4392 - val_acc: 0.4648

Epoch 00004: loss improved from 2.36953 to 2.00425, saving model t
o ../result/mobilenet.h5
Epoch 5/100
4/4 [==============================] - 125s 31s/step - loss: 1.680
8 - acc: 0.4355 - val_loss: 1.1178 - val_acc: 0.5070

Epoch 00005: loss improved from 2.00425 to 1.71914, saving model t
o ../result/mobilenet.h5
Epoch 6/100
4/4 [==============================] - 128s 32s/step - loss: 1.513
8 - acc: 0.4798 - val_loss: 1.3574 - val_acc: 0.5634

Epoch 00006: loss improved from 1.71914 to 1.53824, saving model t
o ../result/mobilenet.h5
Epoch 7/100
4/4 [==============================] - 122s 30s/step - loss: 1.231
3 - acc: 0.5443 - val_loss: 1.0452 - val_acc: 0.5634

Epoch 00007: loss improved from 1.53824 to 1.25788, saving model t
o ../result/mobilenet.h5
Epoch 8/100
4/4 [==============================] - 123s 31s/step - loss: 1.231
5 - acc: 0.5727 - val_loss: 1.1087 - val_acc: 0.6479

Epoch 00008: loss improved from 1.25788 to 1.14232, saving model t
o ../result/mobilenet.h5
Epoch 9/100
4/4 [==============================] - 123s 31s/step - loss: 0.982
3 - acc: 0.6572 - val_loss: 0.7854 - val_acc: 0.6620

Epoch 00009: loss improved from 1.14232 to 1.00751, saving model t
o ../result/mobilenet.h5
Epoch 10/100
4/4 [==============================] - 127s 32s/step - loss: 0.975
0 - acc: 0.6451 - val_loss: 0.7856 - val_acc: 0.7183

Epoch 00010: loss did not improve from 1.00751
Epoch 11/100
4/4 [==============================] - 120s 30s/step - loss: 1.018
3 - acc: 0.6250 - val_loss: 0.7211 - val_acc: 0.7042

Epoch 00011: loss improved from 1.00751 to 1.00332, saving model t
o ../result/mobilenet.h5
Epoch 12/100
4/4 [==============================] - 119s 30s/step - loss: 0.930
8 - acc: 0.6209 - val_loss: 0.7575 - val_acc: 0.7606

Epoch 00012: loss improved from 1.00332 to 0.97064, saving model t
o ../result/mobilenet.h5
Epoch 13/100
4/4 [==============================] - 118s 29s/step - loss: 0.872
5 - acc: 0.6896 - val_loss: 0.8707 - val_acc: 0.6761
```

```
Epoch 00013: loss improved from 0.97064 to 0.83042, saving model t
o ../result/mobilenet.h5
Epoch 14/100
4/4 [==============================] - 125s 31s/step - loss: 0.885
8 - acc: 0.6895 - val_loss: 0.8728 - val_acc: 0.7042

Epoch 00014: loss did not improve from 0.83042
Epoch 15/100
4/4 [==============================] - 108s 27s/step - loss: 0.756
0 - acc: 0.7500 - val_loss: 0.7146 - val_acc: 0.6901

Epoch 00015: loss improved from 0.83042 to 0.73630, saving model t
o ../result/mobilenet.h5
Epoch 16/100
4/4 [==============================] - 107s 27s/step - loss: 0.766
6 - acc: 0.7097 - val_loss: 0.7223 - val_acc: 0.7465

Epoch 00016: loss did not improve from 0.73630
Epoch 17/100
4/4 [==============================] - 106s 27s/step - loss: 0.690
9 - acc: 0.7339 - val_loss: 0.5852 - val_acc: 0.7887

Epoch 00017: loss improved from 0.73630 to 0.68300, saving model t
o ../result/mobilenet.h5
Epoch 18/100
4/4 [==============================] - 106s 26s/step - loss: 0.679
3 - acc: 0.7379 - val_loss: 0.4792 - val_acc: 0.8169

Epoch 00018: loss improved from 0.68300 to 0.68194, saving model t
o ../result/mobilenet.h5
Epoch 19/100
4/4 [==============================] - 106s 26s/step - loss: 0.599
0 - acc: 0.7782 - val_loss: 0.5705 - val_acc: 0.8310

Epoch 00019: loss improved from 0.68194 to 0.59836, saving model t
o ../result/mobilenet.h5
Epoch 20/100
4/4 [==============================] - 105s 26s/step - loss: 0.689
5 - acc: 0.7621 - val_loss: 0.3698 - val_acc: 0.8592

Epoch 00020: loss did not improve from 0.59836
Epoch 21/100
4/4 [==============================] - 106s 26s/step - loss: 0.685
7 - acc: 0.7661 - val_loss: 0.4024 - val_acc: 0.8451

Epoch 00021: loss did not improve from 0.59836
Epoch 22/100
4/4 [==============================] - 106s 26s/step - loss: 0.482
2 - acc: 0.8306 - val_loss: 0.5349 - val_acc: 0.7746

Epoch 00022: loss improved from 0.59836 to 0.48483, saving model t
o ../result/mobilenet.h5
Epoch 23/100
4/4 [==============================] - 105s 26s/step - loss: 0.539
6 - acc: 0.8064 - val_loss: 0.3609 - val_acc: 0.8732
```

```
Epoch 00023: loss did not improve from 0.48483
Epoch 24/100
4/4 [==============================] - 105s 26s/step - loss: 0.547
0 - acc: 0.8387 - val_loss: 0.3987 - val_acc: 0.8310

Epoch 00024: loss did not improve from 0.48483
Epoch 25/100
4/4 [==============================] - 106s 26s/step - loss: 0.575
7 - acc: 0.7742 - val_loss: 0.4296 - val_acc: 0.8169

Epoch 00025: loss did not improve from 0.48483
Epoch 26/100
4/4 [==============================] - 106s 26s/step - loss: 0.554
8 - acc: 0.8226 - val_loss: 0.2908 - val_acc: 0.9155

Epoch 00026: loss did not improve from 0.48483
Epoch 27/100
4/4 [==============================] - 109s 27s/step - loss: 0.526
6 - acc: 0.7944 - val_loss: 0.4217 - val_acc: 0.8732

Epoch 00027: loss did not improve from 0.48483
Epoch 28/100
4/4 [==============================] - 122s 31s/step - loss: 0.449
4 - acc: 0.8024 - val_loss: 0.4002 - val_acc: 0.8873

Epoch 00028: loss improved from 0.48483 to 0.45627, saving model t
o ../result/mobilenet.h5
Epoch 29/100
4/4 [==============================] - 126s 32s/step - loss: 0.354
7 - acc: 0.8588 - val_loss: 0.3725 - val_acc: 0.8732

Epoch 00029: loss improved from 0.45627 to 0.37014, saving model t
o ../result/mobilenet.h5
Epoch 30/100
4/4 [==============================] - 120s 30s/step - loss: 0.453
1 - acc: 0.8387 - val_loss: 0.3520 - val_acc: 0.8451

Epoch 00030: loss did not improve from 0.37014
Epoch 31/100
4/4 [==============================] - 109s 27s/step - loss: 0.417
4 - acc: 0.8589 - val_loss: 0.4378 - val_acc: 0.8592

Epoch 00031: loss did not improve from 0.37014
Epoch 32/100
4/4 [==============================] - 119s 30s/step - loss: 0.430
4 - acc: 0.8306 - val_loss: 0.4428 - val_acc: 0.8732

Epoch 00032: loss did not improve from 0.37014
Epoch 33/100
4/4 [==============================] - 128s 32s/step - loss: 0.486
0 - acc: 0.8146 - val_loss: 0.3264 - val_acc: 0.8732

Epoch 00033: loss did not improve from 0.37014
Epoch 34/100
4/4 [==============================] - 128s 32s/step - loss: 0.441
9 - acc: 0.8669 - val_loss: 0.2959 - val_acc: 0.9296
```

```
Epoch 00034: loss did not improve from 0.37014
Epoch 35/100
4/4 [==============================] - 120s 30s/step - loss: 0.340
3 - acc: 0.8911 - val_loss: 0.2860 - val_acc: 0.8873

Epoch 00035: loss improved from 0.37014 to 0.35286, saving model t
o ../result/mobilenet.h5
Epoch 36/100
4/4 [==============================] - 124s 31s/step - loss: 0.374
4 - acc: 0.8589 - val_loss: 0.4458 - val_acc: 0.8451

Epoch 00036: loss did not improve from 0.35286
Epoch 37/100
4/4 [==============================] - 119s 30s/step - loss: 0.306
4 - acc: 0.9072 - val_loss: 0.3516 - val_acc: 0.8873

Epoch 00037: loss improved from 0.35286 to 0.31620, saving model t
o ../result/mobilenet.h5
Epoch 38/100
4/4 [==============================] - 108s 27s/step - loss: 0.472
3 - acc: 0.8065 - val_loss: 0.4097 - val_acc: 0.8592

Epoch 00038: loss did not improve from 0.31620
Epoch 39/100
4/4 [==============================] - 108s 27s/step - loss: 0.429
0 - acc: 0.8427 - val_loss: 0.3314 - val_acc: 0.8732

Epoch 00039: loss did not improve from 0.31620
Epoch 40/100
4/4 [==============================] - 109s 27s/step - loss: 0.320
1 - acc: 0.8911 - val_loss: 0.3421 - val_acc: 0.8592

Epoch 00040: loss improved from 0.31620 to 0.30347, saving model t
o ../result/mobilenet.h5
Epoch 41/100
4/4 [==============================] - 110s 28s/step - loss: 0.326
9 - acc: 0.8750 - val_loss: 0.2918 - val_acc: 0.9014

Epoch 00041: loss improved from 0.30347 to 0.30150, saving model t
o ../result/mobilenet.h5
Epoch 42/100
4/4 [==============================] - 110s 27s/step - loss: 0.300
8 - acc: 0.9072 - val_loss: 0.3434 - val_acc: 0.8732

Epoch 00042: loss did not improve from 0.30150
Epoch 43/100
4/4 [==============================] - 116s 29s/step - loss: 0.388
6 - acc: 0.8629 - val_loss: 0.3300 - val_acc: 0.8873

Epoch 00043: loss did not improve from 0.30150
Epoch 44/100
4/4 [==============================] - 113s 28s/step - loss: 0.348
1 - acc: 0.8710 - val_loss: 0.4753 - val_acc: 0.8592

Epoch 00044: loss did not improve from 0.30150
Epoch 45/100
4/4 [==============================] - 106s 26s/step - loss: 0.315
```

```
9 - acc: 0.9032 - val_loss: 0.2067 - val_acc: 0.9296

Epoch 00045: loss did not improve from 0.30150
Epoch 46/100
4/4 [==============================] - 107s 27s/step - loss: 0.325
1 - acc: 0.8710 - val_loss: 0.3003 - val_acc: 0.9155

Epoch 00046: loss did not improve from 0.30150
Epoch 47/100
4/4 [==============================] - 107s 27s/step - loss: 0.377
8 - acc: 0.8669 - val_loss: 0.2915 - val_acc: 0.9014

Epoch 00047: loss did not improve from 0.30150
Epoch 48/100
4/4 [==============================] - 106s 27s/step - loss: 0.309
6 - acc: 0.8790 - val_loss: 0.2529 - val_acc: 0.9296

Epoch 00048: loss did not improve from 0.30150
Epoch 49/100
4/4 [==============================] - 106s 27s/step - loss: 0.358
2 - acc: 0.8549 - val_loss: 0.2004 - val_acc: 0.9437

Epoch 00049: loss did not improve from 0.30150
Epoch 50/100
4/4 [==============================] - 106s 26s/step - loss: 0.293
5 - acc: 0.9073 - val_loss: 0.2214 - val_acc: 0.9577

Epoch 00050: loss improved from 0.30150 to 0.28932, saving model t
o ../result/mobilenet.h5
Epoch 51/100
4/4 [==============================] - 106s 27s/step - loss: 0.276
8 - acc: 0.8992 - val_loss: 0.2848 - val_acc: 0.9296

Epoch 00051: loss improved from 0.28932 to 0.27403, saving model t
o ../result/mobilenet.h5
Epoch 52/100
4/4 [==============================] - 106s 27s/step - loss: 0.344
6 - acc: 0.8710 - val_loss: 0.3337 - val_acc: 0.8592

Epoch 00052: loss did not improve from 0.27403
Epoch 53/100
4/4 [==============================] - 107s 27s/step - loss: 0.224
7 - acc: 0.9274 - val_loss: 0.2300 - val_acc: 0.9296

Epoch 00053: loss improved from 0.27403 to 0.21935, saving model t
o ../result/mobilenet.h5
Epoch 54/100
4/4 [==============================] - 107s 27s/step - loss: 0.391
7 - acc: 0.8468 - val_loss: 0.1610 - val_acc: 0.9718

Epoch 00054: loss did not improve from 0.21935
Epoch 55/100
4/4 [==============================] - 106s 27s/step - loss: 0.241
8 - acc: 0.9193 - val_loss: 0.2674 - val_acc: 0.9296

Epoch 00055: loss did not improve from 0.21935
Epoch 56/100
```

```
4/4 [==============================] - 107s 27s/step - loss: 0.320
2 - acc: 0.8790 - val_loss: 0.3132 - val_acc: 0.9014

Epoch 00056: loss did not improve from 0.21935
Epoch 57/100
4/4 [==============================] - 110s 28s/step - loss: 0.275
4 - acc: 0.9113 - val_loss: 0.2020 - val_acc: 0.9577

Epoch 00057: loss did not improve from 0.21935
Epoch 58/100
4/4 [==============================] - 109s 27s/step - loss: 0.254
4 - acc: 0.9153 - val_loss: 0.2209 - val_acc: 0.9155

Epoch 00058: loss did not improve from 0.21935
Epoch 59/100
4/4 [==============================] - 109s 27s/step - loss: 0.254
1 - acc: 0.9234 - val_loss: 0.2519 - val_acc: 0.9014

Epoch 00059: loss did not improve from 0.21935
Epoch 60/100
4/4 [==============================] - 109s 27s/step - loss: 0.309
3 - acc: 0.8871 - val_loss: 0.2809 - val_acc: 0.9014

Epoch 00060: loss did not improve from 0.21935
Epoch 61/100
4/4 [==============================] - 108s 27s/step - loss: 0.318
0 - acc: 0.8911 - val_loss: 0.2569 - val_acc: 0.8873

Epoch 00061: loss did not improve from 0.21935
Epoch 62/100
4/4 [==============================] - 109s 27s/step - loss: 0.317
7 - acc: 0.8508 - val_loss: 0.1930 - val_acc: 0.9437

Epoch 00062: loss did not improve from 0.21935
Epoch 63/100
4/4 [==============================] - 109s 27s/step - loss: 0.278
7 - acc: 0.8911 - val_loss: 0.2048 - val_acc: 0.9296

Epoch 00063: loss did not improve from 0.21935
Epoch 64/100
4/4 [==============================] - 108s 27s/step - loss: 0.229
9 - acc: 0.9153 - val_loss: 0.2238 - val_acc: 0.9296

Epoch 00064: loss did not improve from 0.21935
Epoch 65/100
4/4 [==============================] - 106s 26s/step - loss: 0.295
3 - acc: 0.8952 - val_loss: 0.2341 - val_acc: 0.9155

Epoch 00065: loss did not improve from 0.21935
Epoch 66/100
4/4 [==============================] - 105s 26s/step - loss: 0.277
8 - acc: 0.9033 - val_loss: 0.1906 - val_acc: 0.9577

Epoch 00066: loss did not improve from 0.21935
Epoch 67/100
4/4 [==============================] - 105s 26s/step - loss: 0.352
2 - acc: 0.8509 - val_loss: 0.2644 - val_acc: 0.8732
```

```
Epoch 00067: loss did not improve from 0.21935
Epoch 68/100
4/4 [==============================] - 105s 26s/step - loss: 0.280
8 - acc: 0.8992 - val_loss: 0.2606 - val_acc: 0.9014

Epoch 00068: loss did not improve from 0.21935
Epoch 69/100
4/4 [==============================] - 105s 26s/step - loss: 0.252
5 - acc: 0.8992 - val_loss: 0.2698 - val_acc: 0.8873

Epoch 00069: loss did not improve from 0.21935
Epoch 70/100
4/4 [==============================] - 106s 26s/step - loss: 0.207
8 - acc: 0.9435 - val_loss: 0.1952 - val_acc: 0.9437

Epoch 00070: loss did not improve from 0.21935
Epoch 71/100
4/4 [==============================] - 106s 26s/step - loss: 0.260
4 - acc: 0.9032 - val_loss: 0.2206 - val_acc: 0.9155

Epoch 00071: loss did not improve from 0.21935
Epoch 72/100
4/4 [==============================] - 105s 26s/step - loss: 0.230
4 - acc: 0.9153 - val_loss: 0.1384 - val_acc: 0.9577

Epoch 00072: loss improved from 0.21935 to 0.20684, saving model t
o ../result/mobilenet.h5
Epoch 73/100
4/4 [==============================] - 106s 26s/step - loss: 0.250
9 - acc: 0.9194 - val_loss: 0.2035 - val_acc: 0.9577

Epoch 00073: loss did not improve from 0.20684
Epoch 74/100
4/4 [==============================] - 106s 27s/step - loss: 0.174
2 - acc: 0.9435 - val_loss: 0.3043 - val_acc: 0.8732

Epoch 00074: loss improved from 0.20684 to 0.18465, saving model t
o ../result/mobilenet.h5
Epoch 75/100
4/4 [==============================] - 106s 27s/step - loss: 0.262
8 - acc: 0.9234 - val_loss: 0.1689 - val_acc: 0.9577

Epoch 00075: loss did not improve from 0.18465
Epoch 76/100
4/4 [==============================] - 105s 26s/step - loss: 0.292
0 - acc: 0.8670 - val_loss: 0.1834 - val_acc: 0.9437

Epoch 00076: loss did not improve from 0.18465
Epoch 77/100
4/4 [==============================] - 106s 27s/step - loss: 0.202
7 - acc: 0.9315 - val_loss: 0.2010 - val_acc: 0.9155

Epoch 00077: loss did not improve from 0.18465
Epoch 78/100
4/4 [==============================] - 106s 27s/step - loss: 0.195
4 - acc: 0.9314 - val_loss: 0.1374 - val_acc: 0.9437
```

```
Epoch 00078: loss did not improve from 0.18465
Epoch 79/100
4/4 [==============================] - 106s 27s/step - loss: 0.209
6 - acc: 0.9314 - val_loss: 0.1461 - val_acc: 0.9859

Epoch 00079: loss did not improve from 0.18465
Epoch 80/100
4/4 [==============================] - 106s 27s/step - loss: 0.351
7 - acc: 0.8790 - val_loss: 0.0793 - val_acc: 1.0000

Epoch 00080: loss did not improve from 0.18465
Epoch 81/100
4/4 [==============================] - 106s 26s/step - loss: 0.249
6 - acc: 0.9274 - val_loss: 0.1392 - val_acc: 0.9577

Epoch 00081: loss did not improve from 0.18465
Epoch 82/100
4/4 [==============================] - 106s 26s/step - loss: 0.192
3 - acc: 0.9476 - val_loss: 0.1726 - val_acc: 0.9577

Epoch 00082: loss did not improve from 0.18465
Epoch 83/100
4/4 [==============================] - 106s 26s/step - loss: 0.227
6 - acc: 0.9194 - val_loss: 0.2411 - val_acc: 0.9437

Epoch 00083: loss did not improve from 0.18465
Epoch 84/100
4/4 [==============================] - 106s 27s/step - loss: 0.274
5 - acc: 0.8912 - val_loss: 0.1028 - val_acc: 0.9859

Epoch 00084: loss did not improve from 0.18465
Epoch 85/100
4/4 [==============================] - 106s 27s/step - loss: 0.195
1 - acc: 0.9355 - val_loss: 0.1350 - val_acc: 0.9859

Epoch 00085: loss did not improve from 0.18465
Epoch 86/100
4/4 [==============================] - 106s 26s/step - loss: 0.227
9 - acc: 0.9113 - val_loss: 0.1330 - val_acc: 0.9718

Epoch 00086: loss did not improve from 0.18465
Epoch 87/100
4/4 [==============================] - 106s 26s/step - loss: 0.257
9 - acc: 0.8992 - val_loss: 0.1471 - val_acc: 0.9437

Epoch 00087: loss did not improve from 0.18465
Epoch 88/100
4/4 [==============================] - 106s 26s/step - loss: 0.217
8 - acc: 0.9274 - val_loss: 0.1285 - val_acc: 0.9718

Epoch 00088: loss did not improve from 0.18465
Epoch 89/100
4/4 [==============================] - 106s 26s/step - loss: 0.241
8 - acc: 0.9113 - val_loss: 0.1633 - val_acc: 0.9718

Epoch 00089: loss did not improve from 0.18465
```

```
Epoch 90/100
4/4 [==============================] - 107s 27s/step - loss: 0.132
5 - acc: 0.9718 - val_loss: 0.1140 - val_acc: 0.9577

Epoch 00090: loss improved from 0.18465 to 0.13821, saving model t
o ../result/mobilenet.h5
Epoch 91/100
4/4 [==============================] - 110s 28s/step - loss: 0.191
4 - acc: 0.9315 - val_loss: 0.1826 - val_acc: 0.9437

Epoch 00091: loss did not improve from 0.13821
Epoch 92/100
4/4 [==============================] - 109s 27s/step - loss: 0.256
5 - acc: 0.8831 - val_loss: 0.2154 - val_acc: 0.9437

Epoch 00092: loss did not improve from 0.13821
Epoch 93/100
4/4 [==============================] - 109s 27s/step - loss: 0.233
1 - acc: 0.9153 - val_loss: 0.1388 - val_acc: 0.9718

Epoch 00093: loss did not improve from 0.13821
Epoch 94/100
4/4 [==============================] - 107s 27s/step - loss: 0.222
8 - acc: 0.8992 - val_loss: 0.2108 - val_acc: 0.9296

Epoch 00094: loss did not improve from 0.13821
Epoch 95/100
4/4 [==============================] - 106s 27s/step - loss: 0.180
6 - acc: 0.9355 - val_loss: 0.2379 - val_acc: 0.9155

Epoch 00095: loss did not improve from 0.13821
Epoch 96/100
4/4 [==============================] - 105s 26s/step - loss: 0.189
3 - acc: 0.9436 - val_loss: 0.1222 - val_acc: 0.9859

Epoch 00096: loss did not improve from 0.13821
Epoch 97/100
4/4 [==============================] - 106s 26s/step - loss: 0.176
1 - acc: 0.9234 - val_loss: 0.1239 - val_acc: 0.9577

Epoch 00097: loss did not improve from 0.13821
Epoch 98/100
4/4 [==============================] - 106s 26s/step - loss: 0.231
1 - acc: 0.9033 - val_loss: 0.1665 - val_acc: 0.9437

Epoch 00098: loss did not improve from 0.13821
Epoch 99/100
4/4 [==============================] - 106s 26s/step - loss: 0.218
4 - acc: 0.9113 - val_loss: 0.1968 - val_acc: 0.9296

Epoch 00099: loss did not improve from 0.13821
Epoch 100/100
4/4 [==============================] - 106s 26s/step - loss: 0.117
0 - acc: 0.9758 - val_loss: 0.1373 - val_acc: 0.9718

Epoch 00100: loss improved from 0.13821 to 0.12067, saving model t
o ../result/mobilenet.h5
```

```
Learning Time:  184.99 min.
```

Key Point 6: Prepare model not only for Android but for iOS

I can use the original format of model for Android but cannot use it for iOS. So, I convert the Keras model to CoreML model using coremltools. However, you will get "ValueError: Unknown activation function:relu6" message if you just try to convert to CoreML model.

MobileNet has its own custom functions like relu6 that Keras does not know. So, you need to tell Keras those unknown functionns. I teach Keras what exactly relu6 and DepthwiseConv2D indicate by using CustomObjectScope. You can see the error when you commented out the first line of the following code.

In [30]:

```
# You will get "ValueError: Unknown activation function:relu6" if you commente
d out the following first line of code.
with CustomObjectScope({'relu6': mobilenet.relu6, 'DepthwiseConv2D': mobilenet
.DepthwiseConv2D}):
    my_coreml_model = coremltools.converters.keras.convert(os.path.join(result
_dir, 'mobilenet.h5'),
        is_bgr=False,
        image_scale=1.0/255,
        input_names='image',
        image_input_names='image',
        class_labels=class_labels)
    my_coreml_model.save(os.path.join(result_dir, 'mobilenet.mlmodel'))
```

```
0 : input_3, <keras.engine.topology.InputLayer object at 0x13c8017
48>
1 : conv1_pad, <keras.layers.convolutional.ZeroPadding2D object at
0x13c801668>
2 : conv1, <keras.layers.convolutional.Conv2D object at 0x13c8019b
0>
3 : conv1_bn, <keras.layers.normalization.BatchNormalization objec
t at 0x13c801898>
4 : conv1_relu, <keras.layers.core.Activation object at 0x13c801a2
0>
5 : conv_pad_1, <keras.layers.convolutional.ZeroPadding2D object a
t 0x13c801cf8>
6 : conv_dw_1, <keras.layers.convolutional.DepthwiseConv2D object
at 0x13c801d30>
7 : conv_dw_1_bn, <keras.layers.normalization.BatchNormalization o
bject at 0x13c801dd8>
8 : conv_dw_1_relu, <keras.layers.core.Activation object at 0x13c8
01e48>
9 : conv_pw_1, <keras.layers.convolutional.Conv2D object at 0x1377
fb128>
10 : conv_pw_1_bn, <keras.layers.normalization.BatchNormalization
object at 0x1377fb278>
11 : conv_pw_1_relu, <keras.layers.core.Activation object at 0x137
7fb3c8>
12 : conv_pad_2, <keras.layers.convolutional.ZeroPadding2D object
at 0x1377fb400>
```

```
13 : conv_dw_2, <keras.layers.convolutional.DepthwiseConv2D object
at 0x1377fb470>
14 : conv_dw_2_bn, <keras.layers.normalization.BatchNormalization
object at 0x1377fb4e0>
15 : conv_dw_2_relu, <keras.layers.core.Activation object at 0x137
7fb7b8>
16 : conv_pw_2, <keras.layers.convolutional.Conv2D object at 0x137
7fb7f0>
17 : conv_pw_2_bn, <keras.layers.normalization.BatchNormalization
object at 0x1377fb940>
18 : conv_pw_2_relu, <keras.layers.core.Activation object at 0x137
7fba90>
19 : conv_pad_3, <keras.layers.convolutional.ZeroPadding2D object
at 0x1377fbac8>
20 : conv_dw_3, <keras.layers.convolutional.DepthwiseConv2D object
at 0x1377fbb38>
21 : conv_dw_3_bn, <keras.layers.normalization.BatchNormalization
object at 0x1377fbba8>
22 : conv_dw_3_relu, <keras.layers.core.Activation object at 0x137
7fbe80>
23 : conv_pw_3, <keras.layers.convolutional.Conv2D object at 0x137
7fbeb8>
24 : conv_pw_3_bn, <keras.layers.normalization.BatchNormalization
object at 0x13c801fd0>
25 : conv_pw_3_relu, <keras.layers.core.Activation object at 0x137
7ff198>
26 : conv_pad_4, <keras.layers.convolutional.ZeroPadding2D object
at 0x1377ff1d0>
27 : conv_dw_4, <keras.layers.convolutional.DepthwiseConv2D object
at 0x1377ff240>
28 : conv_dw_4_bn, <keras.layers.normalization.BatchNormalization
object at 0x1377ff2b0>
29 : conv_dw_4_relu, <keras.layers.core.Activation object at 0x137
7ff588>
30 : conv_pw_4, <keras.layers.convolutional.Conv2D object at 0x137
7ff5c0>
31 : conv_pw_4_bn, <keras.layers.normalization.BatchNormalization
object at 0x1377ff710>
32 : conv_pw_4_relu, <keras.layers.core.Activation object at 0x137
7ff860>
33 : conv_pad_5, <keras.layers.convolutional.ZeroPadding2D object
at 0x1377ff898>
34 : conv_dw_5, <keras.layers.convolutional.DepthwiseConv2D object
at 0x1377ff908>
35 : conv_dw_5_bn, <keras.layers.normalization.BatchNormalization
object at 0x1377ff978>
36 : conv_dw_5_relu, <keras.layers.core.Activation object at 0x137
7ffc50>
37 : conv_pw_5, <keras.layers.convolutional.Conv2D object at 0x137
7ffc88>
38 : conv_pw_5_bn, <keras.layers.normalization.BatchNormalization
object at 0x1377ffdd8>
39 : conv_pw_5_relu, <keras.layers.core.Activation object at 0x137
7fff28>
40 : conv_pad_6, <keras.layers.convolutional.ZeroPadding2D object
at 0x1377fff60>
41 : conv_dw_6, <keras.layers.convolutional.DepthwiseConv2D object
```

```
at 0x1377fbf60>
42 : conv_dw_6_bn, <keras.layers.normalization.BatchNormalization
object at 0x137808080>
43 : conv_dw_6_relu, <keras.layers.core.Activation object at 0x137
808358>
44 : conv_pw_6, <keras.layers.convolutional.Conv2D object at 0x137
808390>
45 : conv_pw_6_bn, <keras.layers.normalization.BatchNormalization
object at 0x1378084e0>
46 : conv_pw_6_relu, <keras.layers.core.Activation object at 0x137
808630>
47 : conv_pad_7, <keras.layers.convolutional.ZeroPadding2D object
at 0x137808668>
48 : conv_dw_7, <keras.layers.convolutional.DepthwiseConv2D object
at 0x1378086d8>
49 : conv_dw_7_bn, <keras.layers.normalization.BatchNormalization
object at 0x137808748>
50 : conv_dw_7_relu, <keras.layers.core.Activation object at 0x137
808a20>
51 : conv_pw_7, <keras.layers.convolutional.Conv2D object at 0x137
808a58>
52 : conv_pw_7_bn, <keras.layers.normalization.BatchNormalization
object at 0x137808ba8>
53 : conv_pw_7_relu, <keras.layers.core.Activation object at 0x137
808cf8>
54 : conv_pad_8, <keras.layers.convolutional.ZeroPadding2D object
at 0x137808d30>
55 : conv_dw_8, <keras.layers.convolutional.DepthwiseConv2D object
at 0x137808da0>
56 : conv_dw_8_bn, <keras.layers.normalization.BatchNormalization
object at 0x137808e10>
57 : conv_dw_8_relu, <keras.layers.core.Activation object at 0x137
7fffd0>
58 : conv_pw_8, <keras.layers.convolutional.Conv2D object at 0x137
819160>
59 : conv_pw_8_bn, <keras.layers.normalization.BatchNormalization
object at 0x1378192b0>
60 : conv_pw_8_relu, <keras.layers.core.Activation object at 0x137
819400>
61 : conv_pad_9, <keras.layers.convolutional.ZeroPadding2D object
at 0x137819438>
62 : conv_dw_9, <keras.layers.convolutional.DepthwiseConv2D object
at 0x1378194a8>
63 : conv_dw_9_bn, <keras.layers.normalization.BatchNormalization
object at 0x137819518>
64 : conv_dw_9_relu, <keras.layers.core.Activation object at 0x137
8197f0>
65 : conv_pw_9, <keras.layers.convolutional.Conv2D object at 0x137
819828>
66 : conv_pw_9_bn, <keras.layers.normalization.BatchNormalization
object at 0x137819978>
67 : conv_pw_9_relu, <keras.layers.core.Activation object at 0x137
819ac8>
68 : conv_pad_10, <keras.layers.convolutional.ZeroPadding2D object
at 0x137819b00>
69 : conv_dw_10, <keras.layers.convolutional.DepthwiseConv2D objec
t at 0x137819b70>
```

```
70 : conv_dw_10_bn, <keras.layers.normalization.BatchNormalization
object at 0x137819be0>
71 : conv_dw_10_relu, <keras.layers.core.Activation object at 0x13
7819eb8>
72 : conv_pw_10, <keras.layers.convolutional.Conv2D object at 0x13
7819ef0>
73 : conv_pw_10_bn, <keras.layers.normalization.BatchNormalization
object at 0x137808f60>
74 : conv_pw_10_relu, <keras.layers.core.Activation object at 0x13
7820208>
75 : conv_pad_11, <keras.layers.convolutional.ZeroPadding2D object
at 0x137820240>
76 : conv_dw_11, <keras.layers.convolutional.DepthwiseConv2D objec
t at 0x137820198>
77 : conv_dw_11_bn, <keras.layers.normalization.BatchNormalization
object at 0x137820320>
78 : conv_dw_11_relu, <keras.layers.core.Activation object at 0x13
78205f8>
79 : conv_pw_11, <keras.layers.convolutional.Conv2D object at 0x13
78206a0>
80 : conv_pw_11_bn, <keras.layers.normalization.BatchNormalization
object at 0x137820630>
81 : conv_pw_11_relu, <keras.layers.core.Activation object at 0x13
78208d0>
82 : conv_pad_12, <keras.layers.convolutional.ZeroPadding2D object
at 0x137820908>
83 : conv_dw_12, <keras.layers.convolutional.DepthwiseConv2D objec
t at 0x1378209b0>
84 : conv_dw_12_bn, <keras.layers.normalization.BatchNormalization
object at 0x137820a20>
85 : conv_dw_12_relu, <keras.layers.core.Activation object at 0x13
7820cc0>
86 : conv_pw_12, <keras.layers.convolutional.Conv2D object at 0x13
7820ba8>
87 : conv_pw_12_bn, <keras.layers.normalization.BatchNormalization
object at 0x137820e48>
88 : conv_pw_12_relu, <keras.layers.core.Activation object at 0x13
7820f60>
89 : conv_pad_13, <keras.layers.convolutional.ZeroPadding2D object
at 0x137820d68>
90 : conv_dw_13, <keras.layers.convolutional.DepthwiseConv2D objec
t at 0x137826048>
91 : conv_dw_13_bn, <keras.layers.normalization.BatchNormalization
object at 0x1378260b8>
92 : conv_dw_13_relu, <keras.layers.core.Activation object at 0x13
7826390>
93 : conv_pw_13, <keras.layers.convolutional.Conv2D object at 0x13
78263c8>
94 : conv_pw_13_bn, <keras.layers.normalization.BatchNormalization
object at 0x137826518>
95 : conv_pw_13_relu, <keras.layers.core.Activation object at 0x13
7826668>
96 : global_average_pooling2d_3, <keras.layers.pooling.GlobalAvera
gePooling2D object at 0x1378266a0>
97 : fc1, <keras.layers.core.Dense object at 0x137826710>
98 : fc1__activation__, <keras.layers.core.Activation object at 0x
144d33710>
```

```
99 : batch_normalization_3, <keras.layers.normalization.BatchNorma
lization object at 0x137826860>
100 : predictions, <keras.layers.core.Dense object at 0x1378269b0>
101 : predictions__activation__, <keras.layers.core.Activation obj
ect at 0x144d3e3c8>
```

Write History

I write the history data into the file to record the progress of the training. I can see how the training goes after completing the training.

In [31]:

```
result_file = os.path.join(result_dir, 'training_result.txt')
loss = history.history['loss']
acc = history.history['acc']
val_loss = history.history['val_loss']
val_acc = history.history['val_acc']
nb_epoch = len(acc)

with open(result_file, "w") as fp:
    fp.write("epoch\tloss\tacc\tval_loss\tval_acc\n")
    for i in range(nb_epoch):
        fp.write("%d\t%f\t%f\t%f\t%f\n" %
                    (i, loss[i], acc[i], val_loss[i], val_acc[i]))
```

Plot the learning curve

I plot the learning curve and then draw the graph to help better understanding of the training progress. I turned training loss value to negative to converge both validation / training loss toward zero.

In [35]:

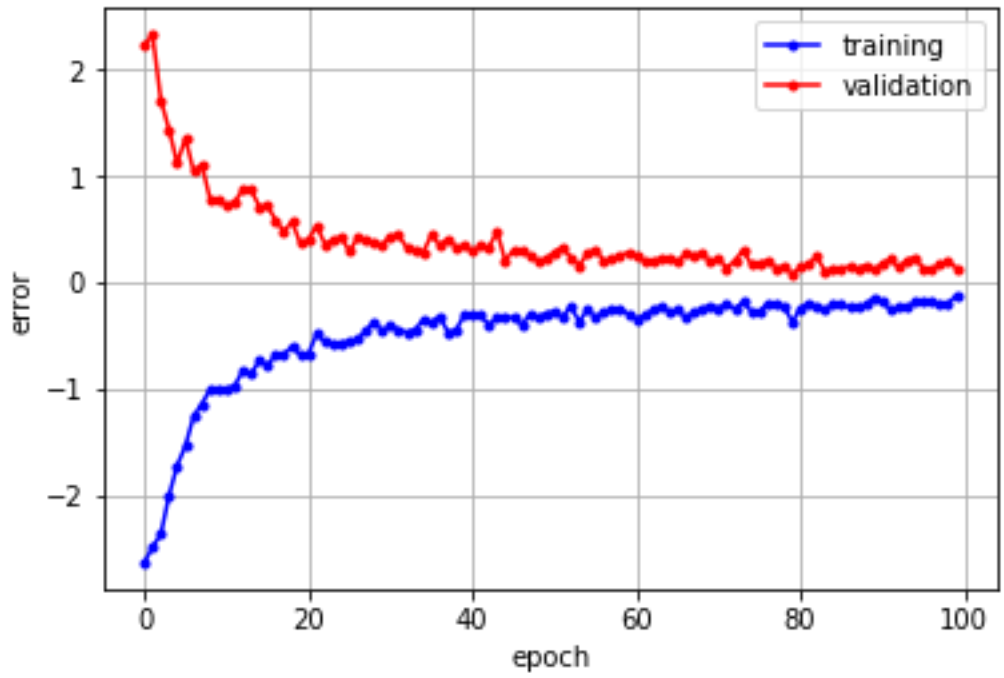
```
epoch_list = []
train_error_list = []
train_acc_list = []
val_error_list = []
val_acc_list = []

with open(result_file) as fp:
    fp.readline()  # skip title
    for line in fp:
        line = line.rstrip()
        cols = line.split('\t')

        epoch = int(cols[0])
        train_error = float(cols[1]) * -1
        train_acc = float(cols[2])
        val_error = float(cols[3])
        val_acc = float(cols[4])

        epoch_list.append(epoch)
        train_error_list.append(train_error)
        train_acc_list.append(train_acc)
        val_error_list.append(val_error)
        val_acc_list.append(val_acc)

plt.figure()
plt.plot(epoch_list, train_error_list, 'b-', marker='.', label='training')
plt.plot(epoch_list, val_error_list, 'r-', marker='.', label='validation')
plt.grid()
plt.legend()
plt.xlabel('epoch')
plt.ylabel('error')
plt.savefig(result_file + ".png")
plt.show()
```



In [ ]: