An Introduction to Deep Learning With Python

[5.3] Using a pretrained convnet

Prof. Yuzo Iano

pgs: 143 - 158

Instantiating the VGG16 convolutional base

In [1]:
```
from keras.applications import VGG16

conv_base = VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3))
conv_base.summary()
```

```
Using TensorFlow backend.

WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\py
thon\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is depr
ecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 150, 150, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Extracting features using the pretrained convolutional base

In [2]:
```
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = '../CAP_5/cats_and_dogs_small'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

datagen = ImageDataGenerator(rescale=1./255)
batch_size = 20

def extract_features(directory, sample_count):
    features = np.zeros(shape=(sample_count, 4, 4, 512))
    labels = np.zeros(shape=(sample_count))
    generator = datagen.flow_from_directory(directory,
                                            target_size=(150, 150),
                                            batch_size=batch_size,
                                            class_mode='binary')
    i = 0
    for inputs_batch, labels_batch in generator:
        features_batch = conv_base.predict(inputs_batch)
        features[i * batch_size : (i + 1) * batch_size] = features_batch
        labels[i * batch_size : (i + 1) * batch_size] = labels_batch
        i += 1
        if i * batch_size >= sample_count:
            break
    return features, labels
```

In [3]:
```
train_features, train_labels = extract_features(train_dir, 2000)
validation_features, validation_labels = extract_features(validation_dir, 1000)
test_features, test_labels = extract_features(test_dir, 1000)
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

In [4]:
```
train_features = np.reshape(train_features, (2000, 4 * 4 * 512))
validation_features = np.reshape(validation_features, (1000, 4 * 4 * 512))
test_features = np.reshape(test_features, (1000, 4 * 4 * 512))
```

Defining and training the densely connected classifier

In [5]:
```python
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop

model = Sequential()
model.add(Dense(256, activation='relu', input_dim=4 * 4 * 512))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From C:\Users\pablo\Python\envs\DAVID\lib\site-packages\keras\backend\tensorflo
w_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecate
d and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 256)               2097408   
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257       
=================================================================
Total params: 2,097,665
Trainable params: 2,097,665
Non-trainable params: 0
_________________________________________________________________
```

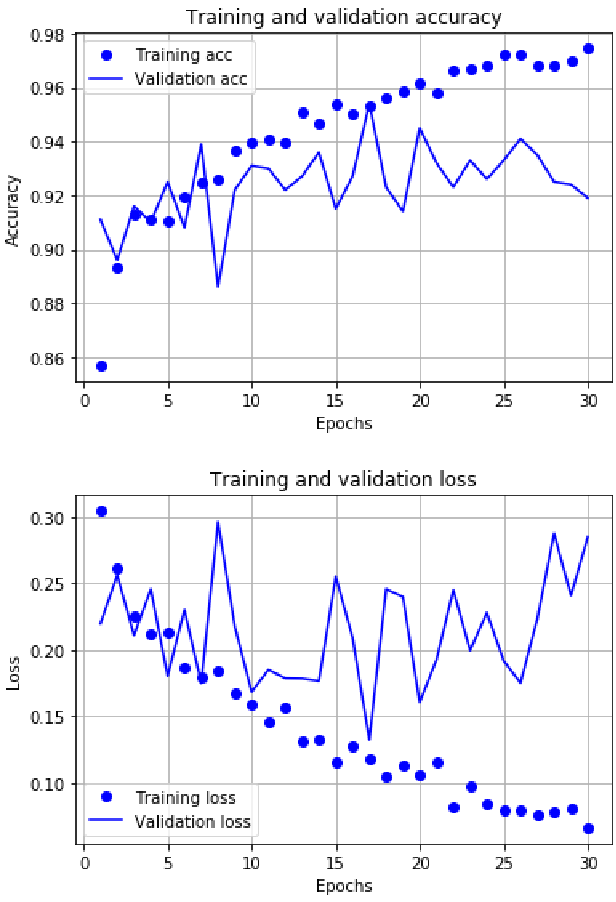
In [6]:
```
model.compile(optimizer=RMSprop(lr=2e-5),
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(train_features, train_labels,
                    epochs=30,
                    batch_size=20,
                    validation_data=(validation_features, validation_labels))
```

```
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\py
thon\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will b
e removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 2000 samples, validate on 1000 samples
Epoch 1/30
2000/2000 [==============================] - 3s 2ms/step - loss: 0.6150 - acc: 0.6705 - val_loss:
0.4784 - val_acc: 0.7820
Epoch 2/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.4401 - acc: 0.8015 - val_loss:
0.3699 - val_acc: 0.8600
Epoch 3/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.3712 - acc: 0.8405 - val_loss:
0.3285 - val_acc: 0.8750
Epoch 4/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.3263 - acc: 0.8660 - val_loss:
0.3045 - val_acc: 0.8850
Epoch 5/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.2912 - acc: 0.8885 - val_loss:
0.2866 - val_acc: 0.8930
Epoch 6/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.2659 - acc: 0.8970 - val_loss:
0.2792 - val_acc: 0.8910
Epoch 7/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.2523 - acc: 0.8975 - val_loss:
0.2681 - val_acc: 0.8920
Epoch 8/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.2379 - acc: 0.9060 - val_loss:
0.2629 - val_acc: 0.8950
Epoch 9/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.2242 - acc: 0.9165 - val_loss:
0.2674 - val_acc: 0.8900
Epoch 10/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.2145 - acc: 0.9125 - val_loss:
0.2505 - val_acc: 0.8980
Epoch 11/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1948 - acc: 0.9295 - val_loss:
0.2485 - val_acc: 0.8990
Epoch 12/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1893 - acc: 0.9240 - val_loss:
0.2450 - val_acc: 0.9000
Epoch 13/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1788 - acc: 0.9315 - val_loss:
0.2477 - val_acc: 0.8940
Epoch 14/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1711 - acc: 0.9415 - val_loss:
0.2428 - val_acc: 0.9010
Epoch 15/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1666 - acc: 0.9375 - val_loss:
0.2422 - val_acc: 0.8980
Epoch 16/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1619 - acc: 0.9420 - val_loss:
0.2433 - val_acc: 0.8990
Epoch 17/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1574 - acc: 0.9445 - val_loss:
0.2373 - val_acc: 0.9040
Epoch 18/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1492 - acc: 0.9470 - val_loss:
0.2368 - val_acc: 0.9030
Epoch 19/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1337 - acc: 0.9540 - val_loss:
0.2350 - val_acc: 0.9060
Epoch 20/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1353 - acc: 0.9525 - val_loss:
0.2352 - val_acc: 0.9010
Epoch 21/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1306 - acc: 0.9585 - val_loss:
0.2356 - val_acc: 0.9020
Epoch 22/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1208 - acc: 0.9605 - val_loss:
0.2378 - val_acc: 0.9020
Epoch 23/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1137 - acc: 0.9610 - val_loss:
0.2355 - val_acc: 0.9050
Epoch 24/30
```

```
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1124 - acc: 0.9650 - val_loss:
0.2411 - val_acc: 0.9030
Epoch 25/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1089 - acc: 0.9630 - val_loss:
0.2399 - val_acc: 0.9000
Epoch 26/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.1042 - acc: 0.9680 - val_loss:
0.2358 - val_acc: 0.9050
Epoch 27/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.0959 - acc: 0.9755 - val_loss:
0.2371 - val_acc: 0.9050
Epoch 28/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.0928 - acc: 0.9720 - val_loss:
0.2408 - val_acc: 0.9050
Epoch 29/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.0935 - acc: 0.9685 - val_loss:
0.2472 - val_acc: 0.9030
Epoch 30/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0.0878 - acc: 0.9740 - val_loss:
0.2393 - val_acc: 0.9030
```

Plotting the results

In [21]:
```
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(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.grid()
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()
plt.show()
```



Adding a densely connected classifier on top of the convolutional base

In [8]:
```
from keras.layers import Flatten

model = Sequential()
model.add(conv_base)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688  
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 256)               2097408   
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 257       
=================================================================
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0
_________________________________________________________________
```

In [9]:
```
print('This is the number of trainable weights before freezing the conv base: ', len(model.trainab
le_weights))
```

```
This is the number of trainable weights before freezing the conv base:  30
```

In [10]:
```
conv_base.trainable = False
print('This is the number of trainable weights before freezing the conv base: ', len(model.trainab
le_weights))
```

```
This is the number of trainable weights before freezing the conv base:  4
```

Training the model end to end with a frozen convolutional base

In [11]:
```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=(150, 150),
                                                    batch_size=20,
                                                    class_mode='binary')
validation_generator = test_datagen.flow_from_directory(validation_dir,
                                                        target_size=(150, 150),
                                                        batch_size=20,
                                                        class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

In [12]:
```
model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=2e-5),
              metrics=['acc'])
history = model.fit_generator(train_generator,
                              steps_per_epoch=100,
                              epochs=30,
                              validation_data=validation_generator,
                              validation_steps=50)
```

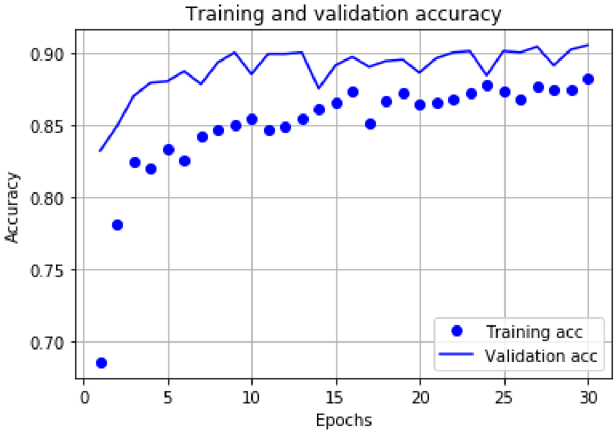
```
Epoch 1/30
100/100 [==============================] - 308s 3s/step - loss: 0.5994 - acc: 0.6860 - val_loss:
0.4581 - val_acc: 0.8320
Epoch 2/30
100/100 [==============================] - 307s 3s/step - loss: 0.5016 - acc: 0.7815 - val_loss:
0.3908 - val_acc: 0.8490
Epoch 3/30
100/100 [==============================] - 306s 3s/step - loss: 0.4370 - acc: 0.8240 - val_loss:
0.3407 - val_acc: 0.8700
Epoch 4/30
100/100 [==============================] - 308s 3s/step - loss: 0.4007 - acc: 0.8195 - val_loss:
0.3150 - val_acc: 0.8790
Epoch 5/30
100/100 [==============================] - 305s 3s/step - loss: 0.3875 - acc: 0.8330 - val_loss:
0.2976 - val_acc: 0.8800
Epoch 6/30
100/100 [==============================] - 305s 3s/step - loss: 0.3818 - acc: 0.8255 - val_loss:
0.2861 - val_acc: 0.8870
Epoch 7/30
100/100 [==============================] - 309s 3s/step - loss: 0.3676 - acc: 0.8425 - val_loss:
0.2909 - val_acc: 0.8780
Epoch 8/30
100/100 [==============================] - 305s 3s/step - loss: 0.3544 - acc: 0.8460 - val_loss:
0.2716 - val_acc: 0.8930
Epoch 9/30
100/100 [==============================] - 309s 3s/step - loss: 0.3454 - acc: 0.8500 - val_loss:
0.2664 - val_acc: 0.9000
Epoch 10/30
100/100 [==============================] - 307s 3s/step - loss: 0.3406 - acc: 0.8540 - val_loss:
0.2643 - val_acc: 0.8850
Epoch 11/30
100/100 [==============================] - 307s 3s/step - loss: 0.3370 - acc: 0.8465 - val_loss:
0.2666 - val_acc: 0.8990
Epoch 12/30
100/100 [==============================] - 308s 3s/step - loss: 0.3347 - acc: 0.8490 - val_loss:
0.2554 - val_acc: 0.8990
Epoch 13/30
100/100 [==============================] - 306s 3s/step - loss: 0.3227 - acc: 0.8540 - val_loss:
0.2506 - val_acc: 0.9000
Epoch 14/30
100/100 [==============================] - 308s 3s/step - loss: 0.3188 - acc: 0.8615 - val_loss:
0.2836 - val_acc: 0.8750
Epoch 15/30
100/100 [==============================] - 305s 3s/step - loss: 0.3194 - acc: 0.8655 - val_loss:
0.2525 - val_acc: 0.8910
Epoch 16/30
100/100 [==============================] - 309s 3s/step - loss: 0.3116 - acc: 0.8730 - val_loss:
0.2601 - val_acc: 0.8970
Epoch 17/30
100/100 [==============================] - 306s 3s/step - loss: 0.3136 - acc: 0.8510 - val_loss:
0.2521 - val_acc: 0.8900
Epoch 18/30
100/100 [==============================] - 309s 3s/step - loss: 0.3155 - acc: 0.8665 - val_loss:
0.2490 - val_acc: 0.8940
Epoch 19/30
100/100 [==============================] - 307s 3s/step - loss: 0.3003 - acc: 0.8725 - val_loss:
0.2461 - val_acc: 0.8950
Epoch 20/30
100/100 [==============================] - 307s 3s/step - loss: 0.3107 - acc: 0.8640 - val_loss:
0.2598 - val_acc: 0.8860
Epoch 21/30
100/100 [==============================] - 308s 3s/step - loss: 0.3096 - acc: 0.8650 - val_loss:
0.2455 - val_acc: 0.8960
Epoch 22/30
100/100 [==============================] - 306s 3s/step - loss: 0.3024 - acc: 0.8675 - val_loss:
0.2419 - val_acc: 0.9000
Epoch 23/30
100/100 [==============================] - 309s 3s/step - loss: 0.3027 - acc: 0.8715 - val_loss:
0.2464 - val_acc: 0.9010
Epoch 24/30
100/100 [==============================] - 307s 3s/step - loss: 0.2884 - acc: 0.8775 - val_loss:
0.2634 - val_acc: 0.8840
Epoch 25/30
100/100 [==============================] - 309s 3s/step - loss: 0.2891 - acc: 0.8730 - val_loss:
0.2399 - val_acc: 0.9010
Epoch 26/30
```

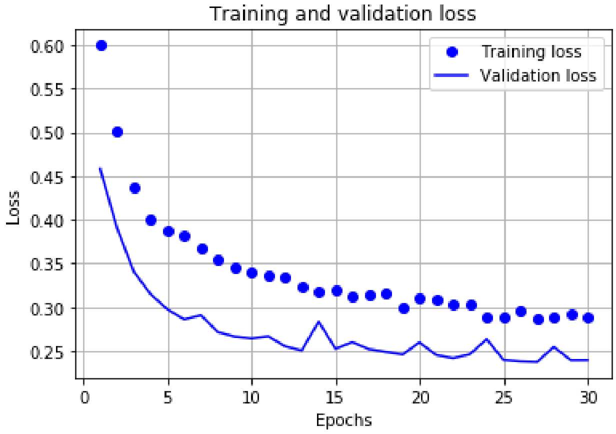
```
100/100 [==============================] - 306s 3s/step - loss: 0.2955 - acc: 0.8680 - val_loss:
0.2382 - val_acc: 0.9000
Epoch 27/30
100/100 [==============================] - 307s 3s/step - loss: 0.2865 - acc: 0.8760 - val_loss:
0.2375 - val_acc: 0.9040
Epoch 28/30
100/100 [==============================] - 307s 3s/step - loss: 0.2886 - acc: 0.8745 - val_loss:
0.2547 - val_acc: 0.8910
Epoch 29/30
100/100 [==============================] - 307s 3s/step - loss: 0.2914 - acc: 0.8745 - val_loss:
0.2395 - val_acc: 0.9020
Epoch 30/30
100/100 [==============================] - 309s 3s/step - loss: 0.2888 - acc: 0.8820 - val_loss:
0.2396 - val_acc: 0.9050
```

Plotting the results

In [13]:

```
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(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.grid()
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()
plt.show()
```





In [14]:
```
conv_base.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 150, 150, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0         
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

Freezing all layers up to a specific one

In [15]:
```
conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'block5_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

Fine-tuning the model

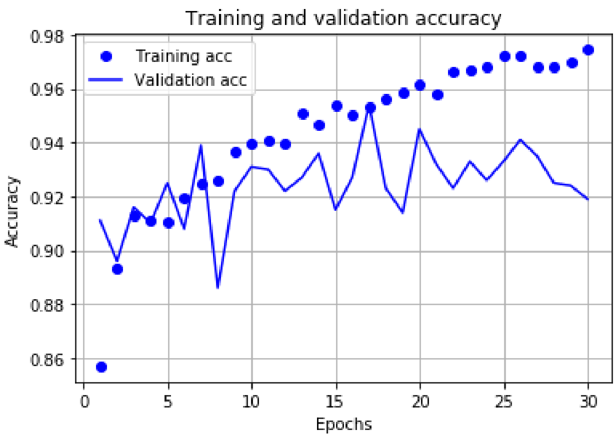
In [16]:
```
model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=1e-5),
              metrics=['acc'])

history = model.fit_generator(train_generator,
                              steps_per_epoch=100,
                              epochs=30,
                              validation_data=validation_generator,
                              validation_steps=50)
```

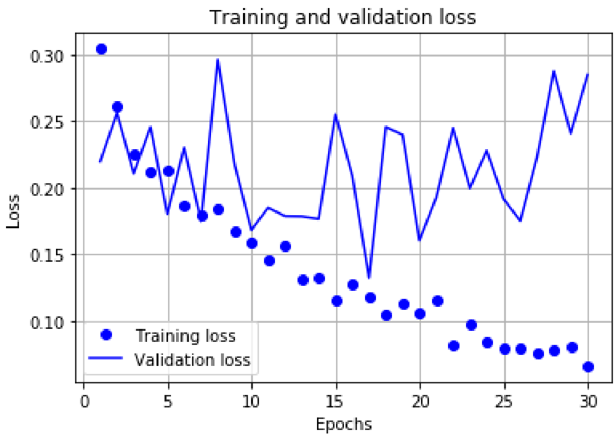
```
Epoch 1/30
100/100 [==============================] - 356s 4s/step - loss: 0.3044 - acc: 0.8570 - val_loss:
0.2197 - val_acc: 0.9110
Epoch 2/30
100/100 [==============================] - 356s 4s/step - loss: 0.2608 - acc: 0.8935 - val_loss:
0.2561 - val_acc: 0.8960
Epoch 3/30
100/100 [==============================] - 355s 4s/step - loss: 0.2253 - acc: 0.9130 - val_loss:
0.2110 - val_acc: 0.9160
Epoch 4/30
100/100 [==============================] - 355s 4s/step - loss: 0.2120 - acc: 0.9110 - val_loss:
0.2456 - val_acc: 0.9100
Epoch 5/30
100/100 [==============================] - 356s 4s/step - loss: 0.2135 - acc: 0.9105 - val_loss:
0.1803 - val_acc: 0.9250
Epoch 6/30
100/100 [==============================] - 355s 4s/step - loss: 0.1866 - acc: 0.9195 - val_loss:
0.2302 - val_acc: 0.9080
Epoch 7/30
100/100 [==============================] - 357s 4s/step - loss: 0.1800 - acc: 0.9250 - val_loss:
0.1749 - val_acc: 0.9390
Epoch 8/30
100/100 [==============================] - 358s 4s/step - loss: 0.1837 - acc: 0.9260 - val_loss:
0.2963 - val_acc: 0.8860
Epoch 9/30
100/100 [==============================] - 357s 4s/step - loss: 0.1669 - acc: 0.9365 - val_loss:
0.2177 - val_acc: 0.9220
Epoch 10/30
100/100 [==============================] - 356s 4s/step - loss: 0.1593 - acc: 0.9395 - val_loss:
0.1684 - val_acc: 0.9310
Epoch 11/30
100/100 [==============================] - 355s 4s/step - loss: 0.1452 - acc: 0.9410 - val_loss:
0.1851 - val_acc: 0.9300
Epoch 12/30
100/100 [==============================] - 355s 4s/step - loss: 0.1572 - acc: 0.9395 - val_loss:
0.1787 - val_acc: 0.9220
Epoch 13/30
100/100 [==============================] - 355s 4s/step - loss: 0.1310 - acc: 0.9510 - val_loss:
0.1785 - val_acc: 0.9270
Epoch 14/30
100/100 [==============================] - 355s 4s/step - loss: 0.1320 - acc: 0.9465 - val_loss:
0.1766 - val_acc: 0.9360
Epoch 15/30
100/100 [==============================] - 355s 4s/step - loss: 0.1155 - acc: 0.9540 - val_loss:
0.2552 - val_acc: 0.9150
Epoch 16/30
100/100 [==============================] - 355s 4s/step - loss: 0.1278 - acc: 0.9500 - val_loss:
0.2094 - val_acc: 0.9270
Epoch 17/30
100/100 [==============================] - 355s 4s/step - loss: 0.1175 - acc: 0.9535 - val_loss:
0.1324 - val_acc: 0.9540
Epoch 18/30
100/100 [==============================] - 355s 4s/step - loss: 0.1053 - acc: 0.9565 - val_loss:
0.2456 - val_acc: 0.9230
Epoch 19/30
100/100 [==============================] - 355s 4s/step - loss: 0.1138 - acc: 0.9585 - val_loss:
0.2398 - val_acc: 0.9140
Epoch 20/30
100/100 [==============================] - 356s 4s/step - loss: 0.1057 - acc: 0.9615 - val_loss:
0.1605 - val_acc: 0.9450
Epoch 21/30
100/100 [==============================] - 355s 4s/step - loss: 0.1159 - acc: 0.9580 - val_loss:
0.1927 - val_acc: 0.9320
Epoch 22/30
100/100 [==============================] - 356s 4s/step - loss: 0.0821 - acc: 0.9665 - val_loss:
0.2450 - val_acc: 0.9230
Epoch 23/30
100/100 [==============================] - 355s 4s/step - loss: 0.0976 - acc: 0.9670 - val_loss:
0.1997 - val_acc: 0.9330
Epoch 24/30
100/100 [==============================] - 356s 4s/step - loss: 0.0843 - acc: 0.9680 - val_loss:
0.2280 - val_acc: 0.9260
Epoch 25/30
100/100 [==============================] - 356s 4s/step - loss: 0.0796 - acc: 0.9720 - val_loss:
0.1918 - val_acc: 0.9330
Epoch 26/30
```

```
100/100 [==============================] - 355s 4s/step - loss: 0.0792 - acc: 0.9720 - val_loss:
0.1751 - val_acc: 0.9410
Epoch 27/30
100/100 [==============================] - 355s 4s/step - loss: 0.0761 - acc: 0.9680 - val_loss:
0.2232 - val_acc: 0.9350
Epoch 28/30
100/100 [==============================] - 355s 4s/step - loss: 0.0779 - acc: 0.9680 - val_loss:
0.2875 - val_acc: 0.9250
Epoch 29/30
100/100 [==============================] - 355s 4s/step - loss: 0.0807 - acc: 0.9700 - val_loss:
0.2408 - val_acc: 0.9240
Epoch 30/30
100/100 [==============================] - 356s 4s/step - loss: 0.0664 - acc: 0.9745 - val_loss:
0.2850 - val_acc: 0.9190
```

Plotting the results

In [17]:
```
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(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.grid()
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()
plt.show()
```

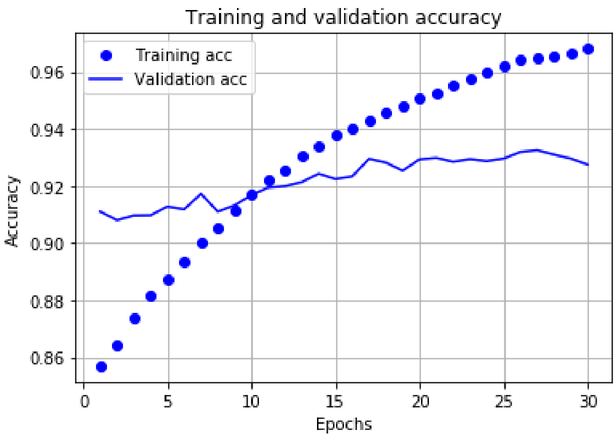


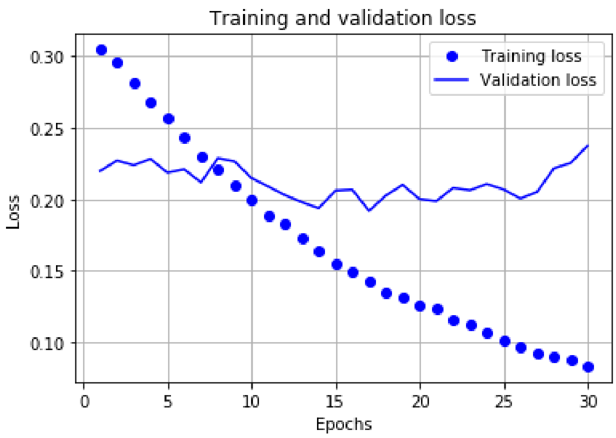


Smoothing the plots

In [18]:
```
def smooth_curve(points, factor=0.8):
    smoothed_points = []
    for point in points:
        if smoothed_points:
            previous = smoothed_points[-1]
            smoothed_points.append(previous * factor + point * (1 - factor))
        else:
            smoothed_points.append(point)
    return smoothed_points
```

In [19]:
```
plt.plot(epochs, smooth_curve(acc), 'bo', label='Training acc')
plt.plot(epochs, smooth_curve(val_acc), 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.grid()
plt.legend()

plt.figure()

plt.plot(epochs, smooth_curve(loss), 'bo', label='Training loss')
plt.plot(epochs, smooth_curve(val_loss), 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid()
plt.legend()
plt.show()
```





In [20]:
```
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  target_size=(150, 150),
                                                  batch_size=20,
                                                  class_mode='binary')

test_loss, test_acc = model.evaluate_generator(test_generator, steps=50)
print('Test acc: ', test_acc)
```

```
Found 1000 images belonging to 2 classes.
Test acc:  0.9249999916553497
```


Pablo Minango

- pablodavid218@gmail.com