Step1 データセット準備（Train, Validation, Test）

In [1]:

```
import os, shutil
```

In [2]:

```
os.getcwd()
```

Out[2]:

```
'/Users/youngsend/KerasLearning'
```

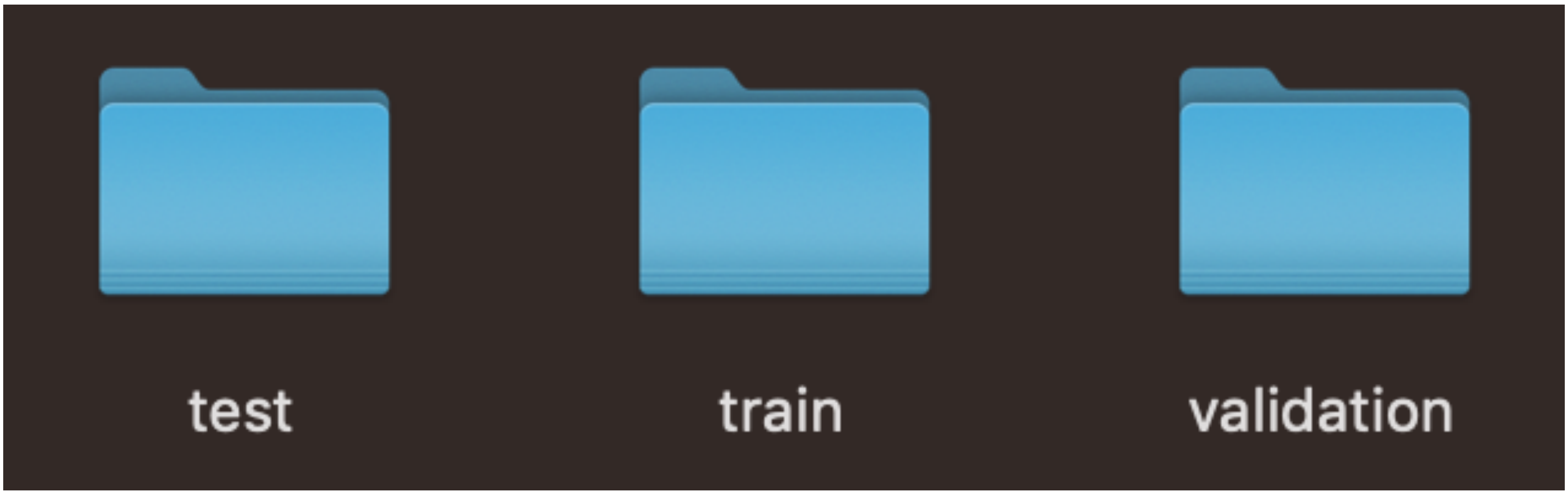
In [3]:

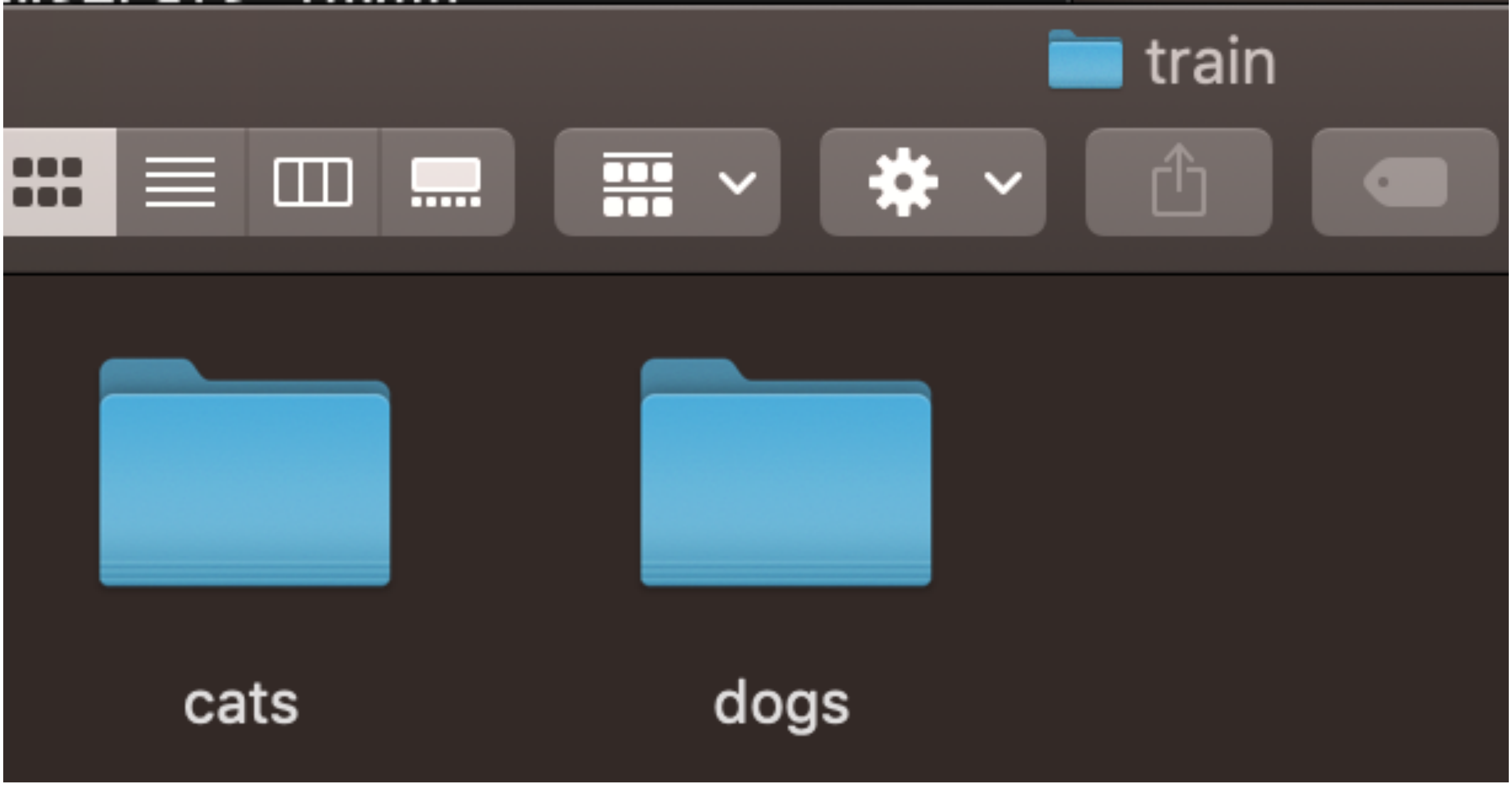
```
original_dataset_dir = '/Users/youngsend/KerasLearning/dogs-vs-cats/train'

base_dir = '/Users/youngsend/KerasLearning/cats_and_dogs_small'
os.mkdir(base_dir)
```

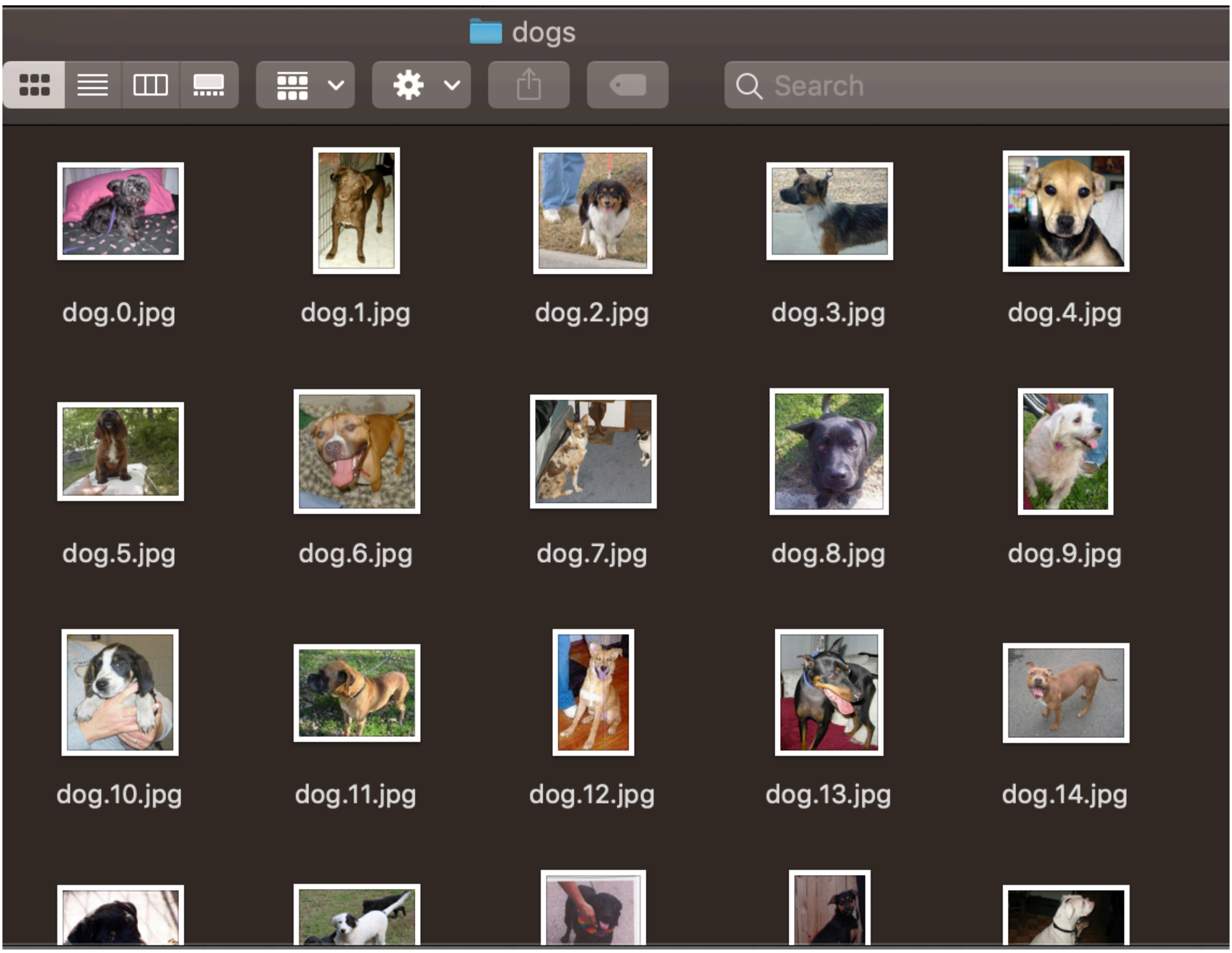
In [4]:

```
train_dir = os.path.join(base_dir, 'train')
os.mkdir(train_dir)
validation_dir = os.path.join(base_dir, 'validation')
os.mkdir(validation_dir)
test_dir = os.path.join(base_dir, 'test')
os.mkdir(test_dir)

train_cats_dir = os.path.join(train_dir, 'cats')
os.mkdir(train_cats_dir)

train_dogs_dir = os.path.join(train_dir, 'dogs')
os.mkdir(train_dogs_dir)

validation_cats_dir = os.path.join(validation_dir, 'cats')
os.mkdir(validation_cats_dir)

validation_dogs_dir = os.path.join(validation_dir, 'dogs')
os.mkdir(validation_dogs_dir)

test_cats_dir = os.path.join(test_dir, 'cats')
os.mkdir(test_cats_dir)

test_dogs_dir = os.path.join(test_dir, 'dogs')
os.mkdir(test_dogs_dir)

fnames = ['cat.{}.jpg'.format(i) for i in range(1000)]
for fname in fnames:
    src = os.path.join(original_dataset_dir, fname)
    dst = os.path.join(train_cats_dir, fname)
    shutil.copyfile(src, dst)
```

```python
fnames = ['cat.{}.jpg'.format(i) for i in range(1000, 1500)]
for fname in fnames:
    src = os.path.join(original_dataset_dir, fname)
    dst = os.path.join(validation_cats_dir, fname)
    shutil.copyfile(src, dst)

fnames = ['cat.{}.jpg'.format(i) for i in range(1500, 2000)]
for fname in fnames:
    src = os.path.join(original_dataset_dir, fname)
    dst = os.path.join(test_cats_dir, fname)
    shutil.copyfile(src, dst)

fnames = ['dog.{}.jpg'.format(i) for i in range(1000)]
for fname in fnames:
    src = os.path.join(original_dataset_dir, fname)
    dst = os.path.join(train_dogs_dir, fname)
    shutil.copyfile(src, dst)

fnames = ['dog.{}.jpg'.format(i) for i in range(1000, 1500)]
for fname in fnames:
    src = os.path.join(original_dataset_dir, fname)
    dst = os.path.join(validation_dogs_dir, fname)
    shutil.copyfile(src, dst)

fnames = ['dog.{}.jpg'.format(i) for i in range(1500, 2000)]
for fname in fnames:
    src = os.path.join(original_dataset_dir, fname)
    dst = os.path.join(test_dogs_dir, fname)
    shutil.copyfile(src, dst)
```







Step2 Networkの構築やコンパイル

In [9]:

```
from keras import layers
from keras import models
```

In [10]:

```
model = models.Sequential()
```

【共有１】： Conv2D関数にinput_shapeというパラメータがないけど、一層目に使うときは、input_shapeを追加する

When using this layer as the first layer in a model, provide the keyword argument `input_shape` (tuple of integers, does not include the batch axis), e.g. `input_shape=(128, 128, 3)` for 128x128 RGB pictures in `data_format="channels_last"`.

In [11]:

```
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

In [12]:

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_8 (Conv2D)            (None, 148, 148, 32)      896       
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 74, 74, 32)        0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 72, 72, 64)        18496     
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 36, 36, 64)        0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 34, 34, 128)       73856     
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 17, 17, 128)       0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 15, 15, 128)       147584    
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 7, 7, 128)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               3211776   
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513       
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

In [14]:

```
from keras import optimizers

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])
```

Step3 データ前処理

In [16]:

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255)
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 [17]:

```
for data_batch, labels_batch in train_generator:
    print('data batch shape:', data_batch.shape)
    print('labels batch shape:', labels_batch.shape)
    break
```

```
data batch shape: (20, 150, 150, 3)
labels batch shape: (20,)
```

In [18]:

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

```
WARNING:tensorflow:From /Library/Frameworks/Python.framework/Versi
ons/3.7/lib/python3.7/site-packages/tensorflow/python/ops/math_ops
.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprec
ated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/30
100/100 [==============================] - 103s 1s/step - loss: 0.
6896 - acc: 0.5290 - val_loss: 0.6738 - val_acc: 0.6240
Epoch 2/30
100/100 [==============================] - 98s 984ms/step - loss:
0.6585 - acc: 0.5975 - val_loss: 0.6442 - val_acc: 0.6320
Epoch 3/30
100/100 [==============================] - 101s 1s/step - loss: 0.
6228 - acc: 0.6545 - val_loss: 0.6126 - val_acc: 0.6580
Epoch 4/30
100/100 [==============================] - 100s 995ms/step - loss:
0.5680 - acc: 0.7075 - val_loss: 0.6033 - val_acc: 0.6610
```

```
Epoch 5/30
100/100 [==============================] - 97s 969ms/step - loss:
0.5416 - acc: 0.7200 - val_loss: 0.5909 - val_acc: 0.6860
Epoch 6/30
100/100 [==============================] - 100s 996ms/step - loss:
0.5110 - acc: 0.7410 - val_loss: 0.5771 - val_acc: 0.6940
Epoch 7/30
100/100 [==============================] - 95s 950ms/step - loss:
0.4818 - acc: 0.7685 - val_loss: 0.5608 - val_acc: 0.7080
Epoch 8/30
100/100 [==============================] - 98s 976ms/step - loss:
0.4543 - acc: 0.7965 - val_loss: 0.5391 - val_acc: 0.7220
Epoch 9/30
100/100 [==============================] - 99s 990ms/step - loss:
0.4262 - acc: 0.8080 - val_loss: 0.6098 - val_acc: 0.6960
Epoch 10/30
100/100 [==============================] - 98s 983ms/step - loss:
0.3997 - acc: 0.8170 - val_loss: 0.5560 - val_acc: 0.7160
Epoch 11/30
100/100 [==============================] - 96s 960ms/step - loss:
0.3793 - acc: 0.8405 - val_loss: 0.6119 - val_acc: 0.7210
Epoch 12/30
100/100 [==============================] - 97s 972ms/step - loss:
0.3537 - acc: 0.8450 - val_loss: 0.6475 - val_acc: 0.7000
Epoch 13/30
100/100 [==============================] - 98s 985ms/step - loss:
0.3215 - acc: 0.8555 - val_loss: 0.5905 - val_acc: 0.7200
Epoch 14/30
100/100 [==============================] - 96s 956ms/step - loss:
0.3065 - acc: 0.8680 - val_loss: 0.5904 - val_acc: 0.7270
Epoch 15/30
100/100 [==============================] - 99s 995ms/step - loss:
0.2687 - acc: 0.8965 - val_loss: 0.6111 - val_acc: 0.7240
Epoch 16/30
100/100 [==============================] - 95s 947ms/step - loss:
0.2512 - acc: 0.8950 - val_loss: 0.6109 - val_acc: 0.7300
Epoch 17/30
100/100 [==============================] - 95s 946ms/step - loss:
0.2303 - acc: 0.9095 - val_loss: 0.6491 - val_acc: 0.7300
Epoch 18/30
100/100 [==============================] - 94s 939ms/step - loss:
0.2106 - acc: 0.9170 - val_loss: 0.6147 - val_acc: 0.7260
Epoch 19/30
100/100 [==============================] - 94s 939ms/step - loss:
0.1901 - acc: 0.9325 - val_loss: 0.6788 - val_acc: 0.7240
Epoch 20/30
100/100 [==============================] - 94s 937ms/step - loss:
0.1674 - acc: 0.9385 - val_loss: 0.7107 - val_acc: 0.7240
Epoch 21/30
100/100 [==============================] - 114s 1s/step - loss: 0.
1469 - acc: 0.9445 - val_loss: 0.7486 - val_acc: 0.7160
Epoch 22/30
100/100 [==============================] - 194s 2s/step - loss: 0.
1336 - acc: 0.9485 - val_loss: 0.7458 - val_acc: 0.7260
Epoch 23/30
100/100 [==============================] - 94s 945ms/step - loss:
0.1121 - acc: 0.9610 - val_loss: 0.7576 - val_acc: 0.7290
```

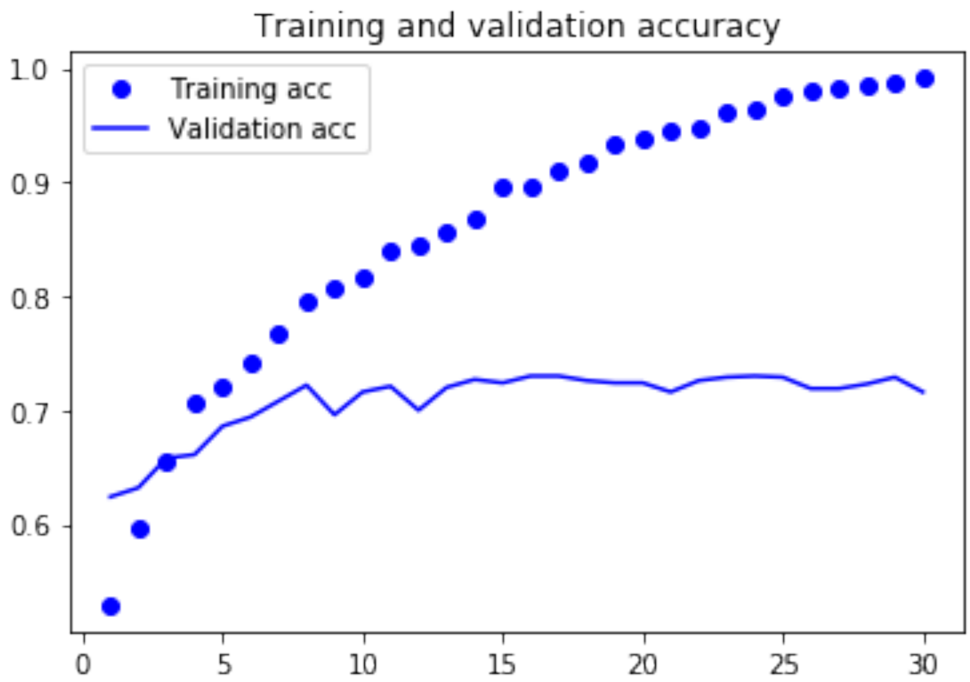
```
Epoch 24/30
100/100 [==============================] - 95s 946ms/step - loss:
0.1016 - acc: 0.9645 - val_loss: 0.7878 - val_acc: 0.7300
Epoch 25/30
100/100 [==============================] - 93s 930ms/step - loss:
0.0808 - acc: 0.9750 - val_loss: 0.8648 - val_acc: 0.7290
Epoch 26/30
100/100 [==============================] - 93s 926ms/step - loss:
0.0789 - acc: 0.9795 - val_loss: 1.0088 - val_acc: 0.7190
Epoch 27/30
100/100 [==============================] - 93s 926ms/step - loss:
0.0603 - acc: 0.9835 - val_loss: 0.9817 - val_acc: 0.7190
Epoch 28/30
100/100 [==============================] - 95s 948ms/step - loss:
0.0561 - acc: 0.9855 - val_loss: 1.0047 - val_acc: 0.7230
Epoch 29/30
100/100 [==============================] - 105s 1s/step - loss: 0.
0495 - acc: 0.9880 - val_loss: 1.0058 - val_acc: 0.7290
Epoch 30/30
100/100 [==============================] - 101s 1s/step - loss: 0.
0405 - acc: 0.9915 - val_loss: 1.0321 - val_acc: 0.7160
```

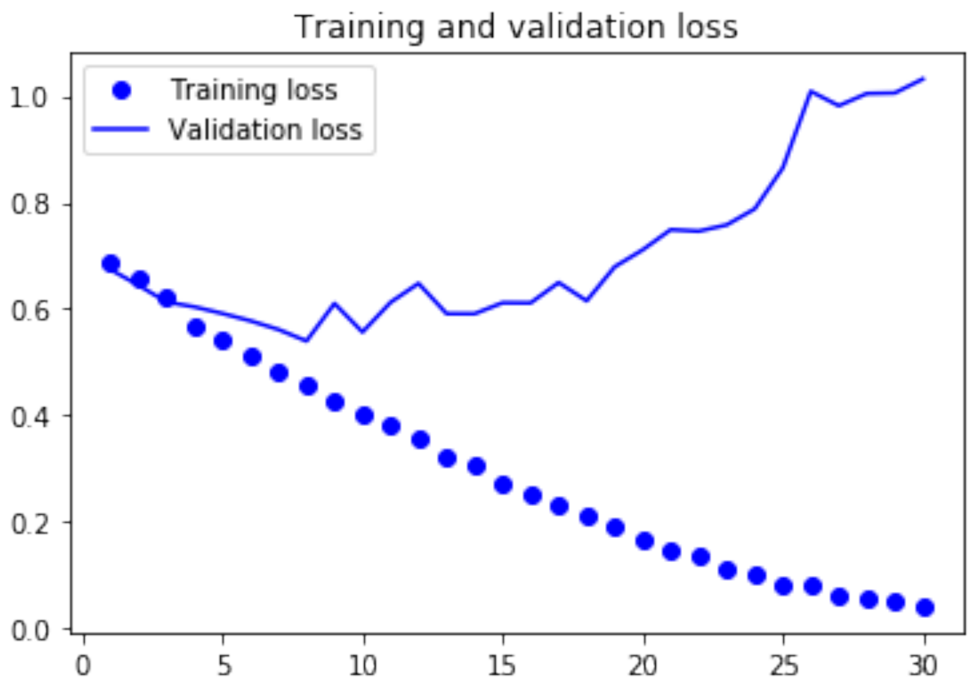
In [19]:

```
model.save('cats_and_dogs_small_1.h5')
```

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.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.legend()

plt.show()
```





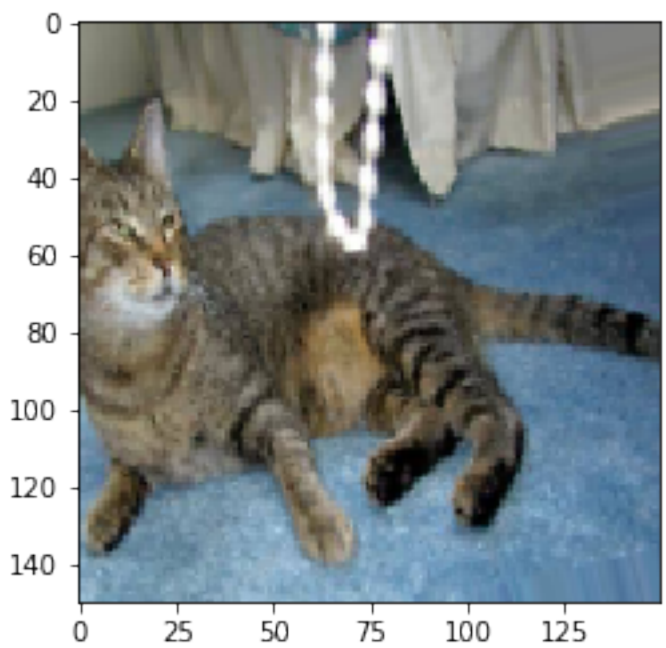
Step4 Data Augmentation（データ増強）を適用する

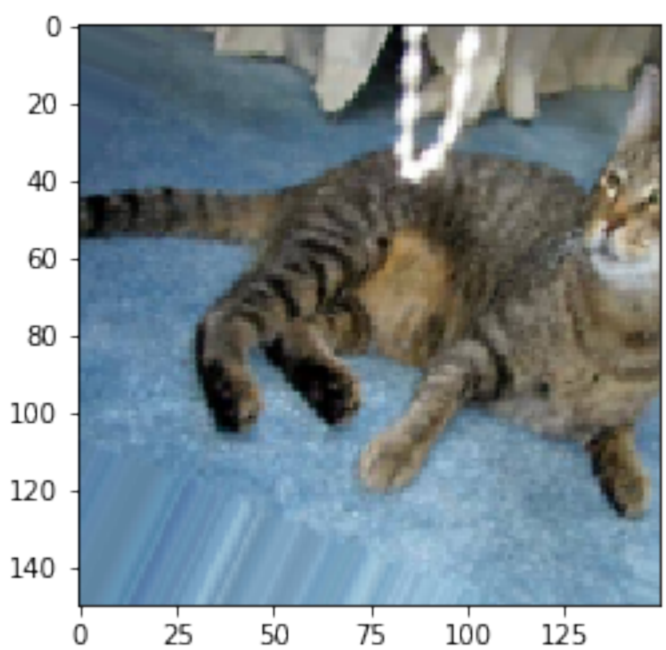
In [22]:

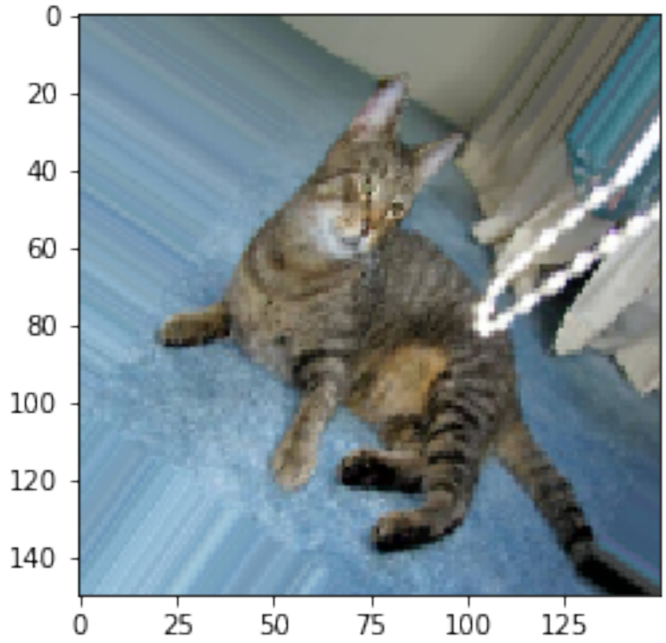
```
datagen = ImageDataGenerator(rotation_range=40,
                             width_shift_range=0.2,
                             height_shift_range=0.2,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True,
                             fill_mode='nearest')
```

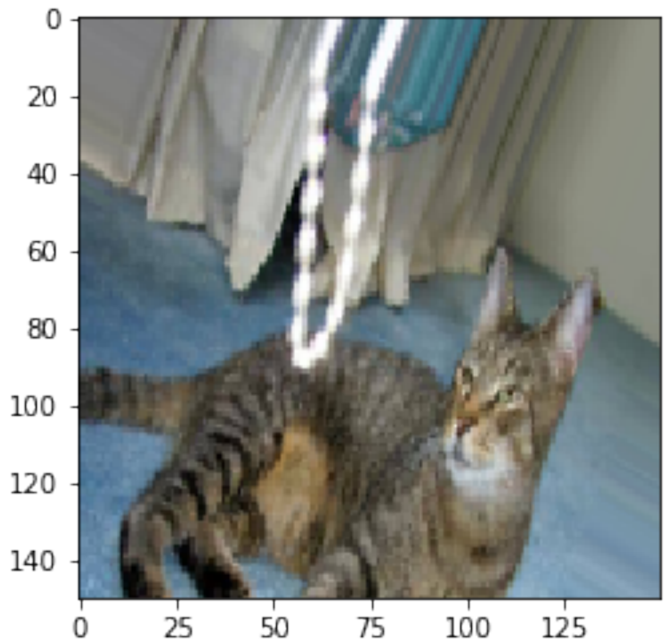
In [26]:

```
from keras.preprocessing import image

fnames = [os.path.join(train_cats_dir, fname) for
          fname in os.listdir(train_cats_dir)]

img_path = fnames[3]

img = image.load_img(img_path, target_size=(150, 150))
x = image.img_to_array(img)

x = x.reshape((1,) + x.shape)

i = 0
for batch in datagen.flow(x, batch_size=1):
    plt.figure(i)
    imgplot = plt.imshow(image.array_to_img(batch[0]))
    i += 1
    if i % 4 == 0:
        break

plt.show()
```









In [27]:

```
img_path
```

Out[27]:

```
'/Users/youngsend/KerasLearning/cats_and_dogs_small/train/cats/cat
.749.jpg'
```


cat.749.jpg
Search

In [32]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
             optimizer=optimizers.RMSprop(lr=1e-4),
             metrics=['acc'])
```

```
WARNING:tensorflow:From /Library/Frameworks/Python.framework/Versi
ons/3.7/lib/python3.7/site-packages/keras/backend/tensorflow_backe
nd.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) wi
th keep_prob is deprecated and will be removed in a future version
.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `r
ate = 1 - keep_prob`.
```

In [33]:

```
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 6272)              0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

In [36]:

```
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)

test_datagen = ImageDataGenerator(rescale=1./255)
```

なぜData Augmentationがoverfittingを解消できるか?

The network will never see the same input twice. But the inputs it sees are still heavily intercorrelated. しかし、Data Augmentationをしない場合、epoch毎に同じinputで訓練する。

In [37]:

```
train_generator = train_datagen.flow_from_directory(train_dir,
                                                     target_size=(150, 150),
                                                     batch_size=32,
                                                     class_mode='binary')

validation_generator = test_datagen.flow_from_directory(validation_dir,
                                                         target_size=(150, 150),
                                                         batch_size=32,
                                                         class_mode='binary')
```

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

In [38]:

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

```
Epoch 1/100
100/100 [==============================] - 159s 2s/step - loss: 0.
6934 - acc: 0.5159 - val_loss: 0.6844 - val_acc: 0.4975
Epoch 2/100
100/100 [==============================] - 156s 2s/step - loss: 0.
6759 - acc: 0.5647 - val_loss: 0.6603 - val_acc: 0.5902
Epoch 3/100
100/100 [==============================] - 162s 2s/step - loss: 0.
6656 - acc: 0.5916 - val_loss: 0.6422 - val_acc: 0.5964
Epoch 4/100
100/100 [==============================] - 152s 2s/step - loss: 0.
6468 - acc: 0.6050 - val_loss: 0.6242 - val_acc: 0.6430
Epoch 5/100
100/100 [==============================] - 157s 2s/step - loss: 0.
6293 - acc: 0.6344 - val_loss: 0.7152 - val_acc: 0.5749
Epoch 6/100
100/100 [==============================] - 151s 2s/step - loss: 0.
6174 - acc: 0.6447 - val_loss: 0.6256 - val_acc: 0.6334
Epoch 7/100
100/100 [==============================] - 151s 2s/step - loss: 0.
6100 - acc: 0.6581 - val_loss: 0.5946 - val_acc: 0.6770
Epoch 8/100
100/100 [==============================] - 150s 2s/step - loss: 0.
5975 - acc: 0.6778 - val_loss: 0.5669 - val_acc: 0.6985
Epoch 9/100
100/100 [==============================] - 149s 1s/step - loss: 0.
5915 - acc: 0.6922 - val_loss: 0.5684 - val_acc: 0.7036
Epoch 10/100
100/100 [==============================] - 150s 2s/step - loss: 0.
5819 - acc: 0.6997 - val_loss: 0.5844 - val_acc: 0.6885
Epoch 11/100
100/100 [==============================] - 148s 1s/step - loss: 0.
5916 - acc: 0.6822 - val_loss: 0.5510 - val_acc: 0.6991
```

```
Epoch 12/100
100/100 [==============================] - 149s 1s/step - loss: 0.
5772 - acc: 0.6944 - val_loss: 0.5298 - val_acc: 0.7246
Epoch 13/100
100/100 [==============================] - 148s 1s/step - loss: 0.
5613 - acc: 0.6978 - val_loss: 0.5543 - val_acc: 0.7043
Epoch 14/100
100/100 [==============================] - 148s 1s/step - loss: 0.
5731 - acc: 0.6972 - val_loss: 0.5737 - val_acc: 0.6808
Epoch 15/100
100/100 [==============================] - 149s 1s/step - loss: 0.
5649 - acc: 0.7025 - val_loss: 0.5382 - val_acc: 0.7120
Epoch 16/100
100/100 [==============================] - 148s 1s/step - loss: 0.
5570 - acc: 0.7166 - val_loss: 0.5516 - val_acc: 0.7223
Epoch 17/100
100/100 [==============================] - 149s 1s/step - loss: 0.
5573 - acc: 0.7156 - val_loss: 0.5357 - val_acc: 0.7373
Epoch 18/100
100/100 [==============================] - 148s 1s/step - loss: 0.
5462 - acc: 0.7228 - val_loss: 0.5611 - val_acc: 0.7159
Epoch 19/100
100/100 [==============================] - 152s 2s/step - loss: 0.
5331 - acc: 0.7334 - val_loss: 0.5776 - val_acc: 0.6904
Epoch 20/100
100/100 [==============================] - 162s 2s/step - loss: 0.
5474 - acc: 0.7159 - val_loss: 0.5309 - val_acc: 0.7300
Epoch 21/100
100/100 [==============================] - 164s 2s/step - loss: 0.
5341 - acc: 0.7288 - val_loss: 0.5254 - val_acc: 0.7303
Epoch 22/100
100/100 [==============================] - 167s 2s/step - loss: 0.
5323 - acc: 0.7344 - val_loss: 0.5017 - val_acc: 0.7494
Epoch 23/100
100/100 [==============================] - 169s 2s/step - loss: 0.
5350 - acc: 0.7216 - val_loss: 0.5487 - val_acc: 0.7094
Epoch 24/100
100/100 [==============================] - 170s 2s/step - loss: 0.
5183 - acc: 0.7384 - val_loss: 0.5201 - val_acc: 0.7268
Epoch 25/100
100/100 [==============================] - 168s 2s/step - loss: 0.
5273 - acc: 0.7287 - val_loss: 0.5241 - val_acc: 0.7378
Epoch 26/100
100/100 [==============================] - 169s 2s/step - loss: 0.
5259 - acc: 0.7359 - val_loss: 0.5072 - val_acc: 0.7437
Epoch 27/100
100/100 [==============================] - 168s 2s/step - loss: 0.
5189 - acc: 0.7331 - val_loss: 0.5068 - val_acc: 0.7481
Epoch 28/100
100/100 [==============================] - 168s 2s/step - loss: 0.
5195 - acc: 0.7366 - val_loss: 0.5242 - val_acc: 0.7259
Epoch 29/100
100/100 [==============================] - 168s 2s/step - loss: 0.
5132 - acc: 0.7400 - val_loss: 0.4971 - val_acc: 0.7500
Epoch 30/100
100/100 [==============================] - 173s 2s/step - loss: 0.
5008 - acc: 0.7509 - val_loss: 0.4943 - val_acc: 0.7716
```

```
Epoch 31/100
100/100 [==============================] - 167s 2s/step - loss: 0.
5029 - acc: 0.7550 - val_loss: 0.5001 - val_acc: 0.7577
Epoch 32/100
100/100 [==============================] - 168s 2s/step - loss: 0.
4972 - acc: 0.7553 - val_loss: 0.5018 - val_acc: 0.7622
Epoch 33/100
100/100 [==============================] - 168s 2s/step - loss: 0.
4835 - acc: 0.7653 - val_loss: 0.5997 - val_acc: 0.7100
Epoch 34/100
100/100 [==============================] - 168s 2s/step - loss: 0.
5061 - acc: 0.7559 - val_loss: 0.4932 - val_acc: 0.7532
Epoch 35/100
100/100 [==============================] - 167s 2s/step - loss: 0.
4914 - acc: 0.7534 - val_loss: 0.5067 - val_acc: 0.7487
Epoch 36/100
100/100 [==============================] - 166s 2s/step - loss: 0.
4931 - acc: 0.7584 - val_loss: 0.5051 - val_acc: 0.7532
Epoch 37/100
100/100 [==============================] - 167s 2s/step - loss: 0.
4891 - acc: 0.7597 - val_loss: 0.5294 - val_acc: 0.7494
Epoch 38/100
100/100 [==============================] - 167s 2s/step - loss: 0.
4783 - acc: 0.7800 - val_loss: 0.5036 - val_acc: 0.7610
Epoch 39/100
100/100 [==============================] - 168s 2s/step - loss: 0.
4925 - acc: 0.7628 - val_loss: 0.4688 - val_acc: 0.7779
Epoch 40/100
100/100 [==============================] - 167s 2s/step - loss: 0.
4786 - acc: 0.7703 - val_loss: 0.5188 - val_acc: 0.7455
Epoch 41/100
100/100 [==============================] - 167s 2s/step - loss: 0.
4859 - acc: 0.7703 - val_loss: 0.4956 - val_acc: 0.7655
Epoch 42/100
100/100 [==============================] - 172s 2s/step - loss: 0.
4806 - acc: 0.7703 - val_loss: 0.5597 - val_acc: 0.7449
Epoch 43/100
100/100 [==============================] - 166s 2s/step - loss: 0.
4818 - acc: 0.7710 - val_loss: 0.6214 - val_acc: 0.6856
Epoch 44/100
100/100 [==============================] - 167s 2s/step - loss: 0.
4598 - acc: 0.7797 - val_loss: 0.4790 - val_acc: 0.7747
Epoch 45/100
100/100 [==============================] - 166s 2s/step - loss: 0.
4715 - acc: 0.7756 - val_loss: 0.5058 - val_acc: 0.7345
Epoch 46/100
100/100 [==============================] - 166s 2s/step - loss: 0.
4662 - acc: 0.7750 - val_loss: 0.4684 - val_acc: 0.7792
Epoch 47/100
100/100 [==============================] - 167s 2s/step - loss: 0.
4710 - acc: 0.7819 - val_loss: 0.4858 - val_acc: 0.7861
Epoch 48/100
100/100 [==============================] - 170s 2s/step - loss: 0.
4484 - acc: 0.7941 - val_loss: 0.5712 - val_acc: 0.7487
Epoch 49/100
100/100 [==============================] - 170s 2s/step - loss: 0.
4628 - acc: 0.7759 - val_loss: 0.7476 - val_acc: 0.6726
```

```
Epoch 50/100
100/100 [==============================] - 150s 1s/step - loss: 0.
4694 - acc: 0.7737 - val_loss: 0.4580 - val_acc: 0.7713
Epoch 51/100
100/100 [==============================] - 151s 2s/step - loss: 0.
4556 - acc: 0.7847 - val_loss: 0.5068 - val_acc: 0.7506
Epoch 52/100
100/100 [==============================] - 150s 1s/step - loss: 0.
4628 - acc: 0.7784 - val_loss: 0.4983 - val_acc: 0.7668
Epoch 53/100
100/100 [==============================] - 150s 2s/step - loss: 0.
4542 - acc: 0.7922 - val_loss: 0.4668 - val_acc: 0.8008
Epoch 54/100
100/100 [==============================] - 150s 2s/step - loss: 0.
4370 - acc: 0.7975 - val_loss: 0.5384 - val_acc: 0.7577
Epoch 55/100
100/100 [==============================] - 149s 1s/step - loss: 0.
4580 - acc: 0.7803 - val_loss: 0.4427 - val_acc: 0.7931
Epoch 56/100
100/100 [==============================] - 149s 1s/step - loss: 0.
4427 - acc: 0.7944 - val_loss: 0.6238 - val_acc: 0.7191
Epoch 57/100
100/100 [==============================] - 147s 1s/step - loss: 0.
4522 - acc: 0.7822 - val_loss: 0.4684 - val_acc: 0.7693
Epoch 58/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4474 - acc: 0.7906 - val_loss: 0.4705 - val_acc: 0.7773
Epoch 59/100
100/100 [==============================] - 147s 1s/step - loss: 0.
4432 - acc: 0.7975 - val_loss: 0.5166 - val_acc: 0.7758
Epoch 60/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4497 - acc: 0.7903 - val_loss: 0.4468 - val_acc: 0.7995
Epoch 61/100
100/100 [==============================] - 147s 1s/step - loss: 0.
4487 - acc: 0.7947 - val_loss: 0.4796 - val_acc: 0.8003
Epoch 62/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4291 - acc: 0.7963 - val_loss: 0.5282 - val_acc: 0.7684
Epoch 63/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4443 - acc: 0.7912 - val_loss: 0.4811 - val_acc: 0.7790
Epoch 64/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4392 - acc: 0.7966 - val_loss: 0.4897 - val_acc: 0.7758
Epoch 65/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4523 - acc: 0.7928 - val_loss: 0.5685 - val_acc: 0.7367
Epoch 66/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4288 - acc: 0.8066 - val_loss: 0.4908 - val_acc: 0.7577
Epoch 67/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4351 - acc: 0.7987 - val_loss: 0.5071 - val_acc: 0.7862
Epoch 68/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4437 - acc: 0.7922 - val_loss: 0.4577 - val_acc: 0.7764
```

```
Epoch 69/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4159 - acc: 0.8141 - val_loss: 0.4567 - val_acc: 0.8020
Epoch 70/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4273 - acc: 0.8013 - val_loss: 0.5008 - val_acc: 0.7751
Epoch 71/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4159 - acc: 0.7987 - val_loss: 0.4826 - val_acc: 0.7817
Epoch 72/100
100/100 [==============================] - 145s 1s/step - loss: 0.
4249 - acc: 0.8034 - val_loss: 0.4719 - val_acc: 0.7841
Epoch 73/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4228 - acc: 0.8010 - val_loss: 0.4670 - val_acc: 0.7745
Epoch 74/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4061 - acc: 0.8156 - val_loss: 0.4946 - val_acc: 0.7519
Epoch 75/100
100/100 [==============================] - 145s 1s/step - loss: 0.
4140 - acc: 0.8122 - val_loss: 0.5202 - val_acc: 0.7745
Epoch 76/100
100/100 [==============================] - 147s 1s/step - loss: 0.
4269 - acc: 0.7928 - val_loss: 0.5147 - val_acc: 0.7811
Epoch 77/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4202 - acc: 0.8041 - val_loss: 0.4525 - val_acc: 0.7957
Epoch 78/100
100/100 [==============================] - 147s 1s/step - loss: 0.
4237 - acc: 0.8078 - val_loss: 0.4749 - val_acc: 0.7900
Epoch 79/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4255 - acc: 0.8084 - val_loss: 0.4322 - val_acc: 0.8009
Epoch 80/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4161 - acc: 0.8087 - val_loss: 0.4560 - val_acc: 0.7932
Epoch 81/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4191 - acc: 0.8109 - val_loss: 0.4984 - val_acc: 0.7779
Epoch 82/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4066 - acc: 0.8166 - val_loss: 0.4451 - val_acc: 0.7925
Epoch 83/100
100/100 [==============================] - 147s 1s/step - loss: 0.
4121 - acc: 0.8069 - val_loss: 0.4672 - val_acc: 0.7766
Epoch 84/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4021 - acc: 0.8131 - val_loss: 0.4766 - val_acc: 0.7880
Epoch 85/100
100/100 [==============================] - 147s 1s/step - loss: 0.
3940 - acc: 0.8219 - val_loss: 0.4116 - val_acc: 0.8147
Epoch 86/100
100/100 [==============================] - 146s 1s/step - loss: 0.
3911 - acc: 0.8153 - val_loss: 0.5076 - val_acc: 0.7423
Epoch 87/100
100/100 [==============================] - 147s 1s/step - loss: 0.
4056 - acc: 0.8144 - val_loss: 0.4589 - val_acc: 0.7792
```

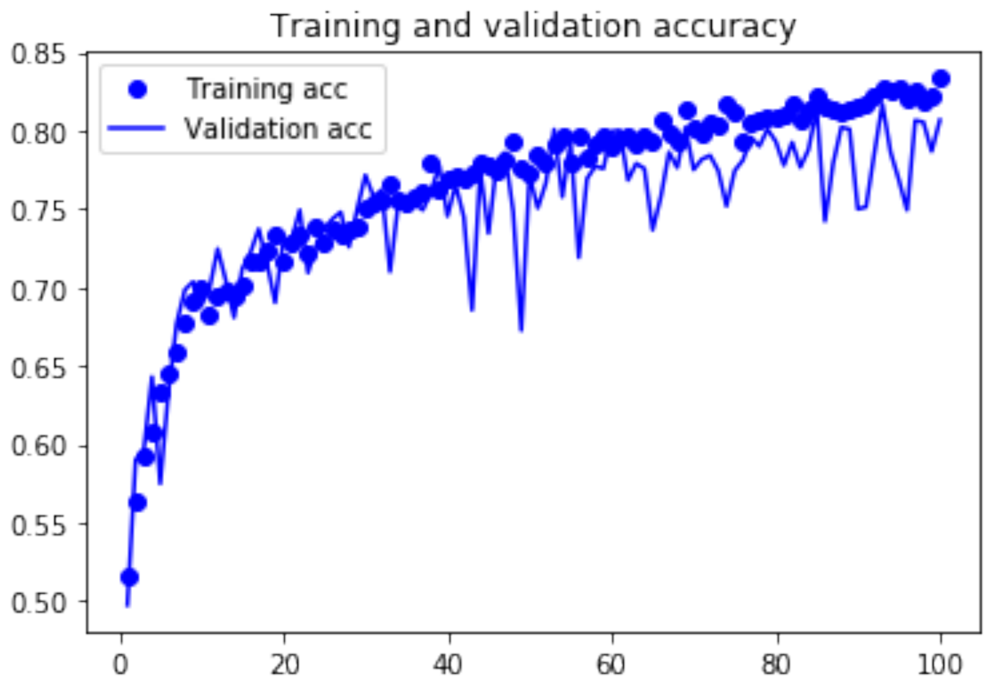
```
Epoch 88/100
100/100 [==============================] - 147s 1s/step - loss: 0.
3981 - acc: 0.8116 - val_loss: 0.4441 - val_acc: 0.8022
Epoch 89/100
100/100 [==============================] - 146s 1s/step - loss: 0.
3992 - acc: 0.8144 - val_loss: 0.4779 - val_acc: 0.8009
Epoch 90/100
100/100 [==============================] - 147s 1s/step - loss: 0.
3970 - acc: 0.8153 - val_loss: 0.5245 - val_acc: 0.7500
Epoch 91/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4025 - acc: 0.8166 - val_loss: 0.5735 - val_acc: 0.7513
Epoch 92/100
100/100 [==============================] - 147s 1s/step - loss: 0.
3982 - acc: 0.8203 - val_loss: 0.4874 - val_acc: 0.7849
Epoch 93/100
100/100 [==============================] - 146s 1s/step - loss: 0.
3879 - acc: 0.8281 - val_loss: 0.4185 - val_acc: 0.8170
Epoch 94/100
100/100 [==============================] - 146s 1s/step - loss: 0.
3863 - acc: 0.8256 - val_loss: 0.4770 - val_acc: 0.7862
Epoch 95/100
100/100 [==============================] - 147s 1s/step - loss: 0.
3875 - acc: 0.8266 - val_loss: 0.5528 - val_acc: 0.7687
Epoch 96/100
100/100 [==============================] - 147s 1s/step - loss: 0.
3850 - acc: 0.8216 - val_loss: 0.5511 - val_acc: 0.7494
Epoch 97/100
100/100 [==============================] - 147s 1s/step - loss: 0.
3919 - acc: 0.8253 - val_loss: 0.4230 - val_acc: 0.8065
Epoch 98/100
100/100 [==============================] - 146s 1s/step - loss: 0.
4045 - acc: 0.8181 - val_loss: 0.4312 - val_acc: 0.8054
Epoch 99/100
100/100 [==============================] - 146s 1s/step - loss: 0.
3966 - acc: 0.8222 - val_loss: 0.4582 - val_acc: 0.7868
Epoch 100/100
100/100 [==============================] - 146s 1s/step - loss: 0.
3949 - acc: 0.8331 - val_loss: 0.4384 - val_acc: 0.8067
```

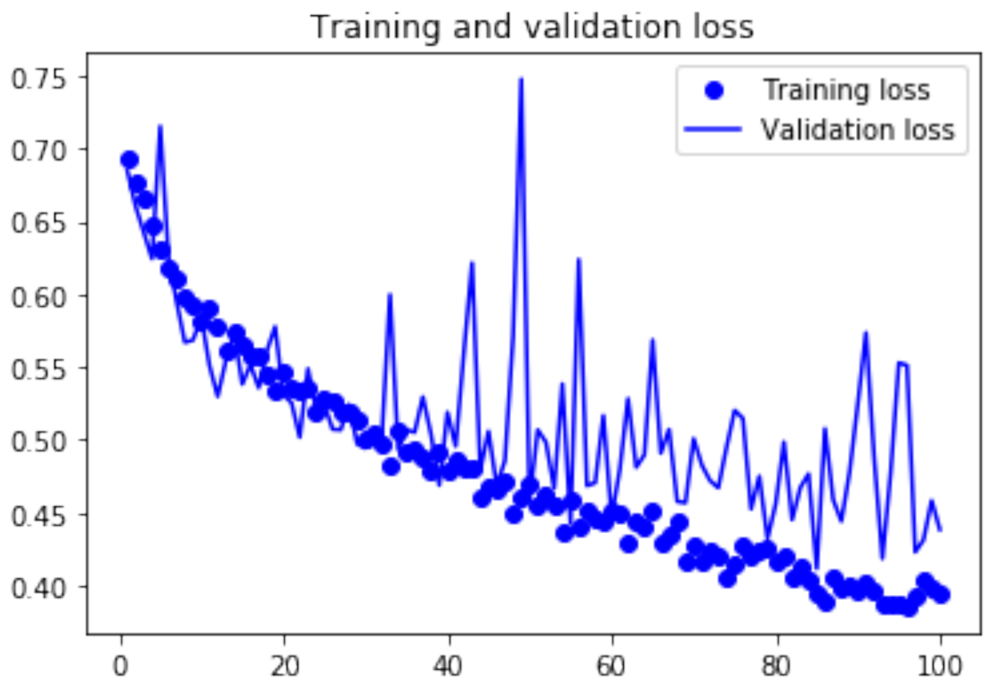
In [39]:

```
model.save('cats_and_dogs_small_2.h5')
```

In [40]:

```
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.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.legend()

plt.show()
```





Step5 Pretrainedモデルを使う

Pretrainedモデルを使う2つ方法：feature extraction and fine-tuning

In [43]:

```
from keras.applications import VGG16

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

```
Downloading data from https://github.com/fchollet/deep-learning-mo
dels/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kerne
ls_notop.h5
58892288/58889256 [==============================] - 40s 1us/step
```

In [52]:

```
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: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

In [46]:

```
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = '/Users/youngsend/KerasLearning/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

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 [49]:

```
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))
```

In [50]:

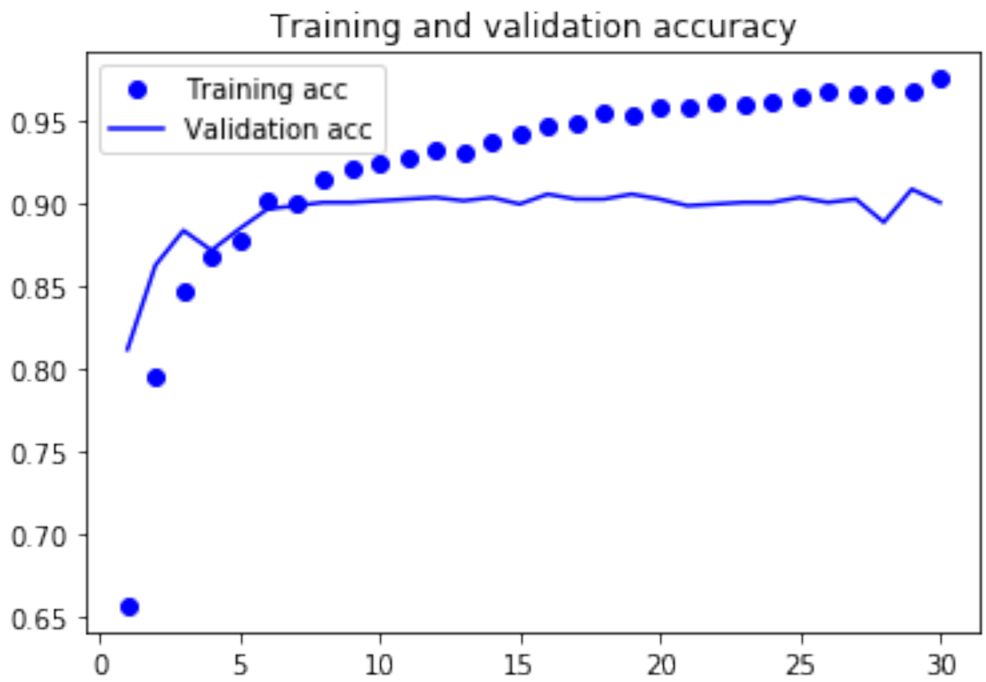
```
from keras import models
from keras import layers
from keras import optimizers

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

model.compile(optimizer=optimizers.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))
```

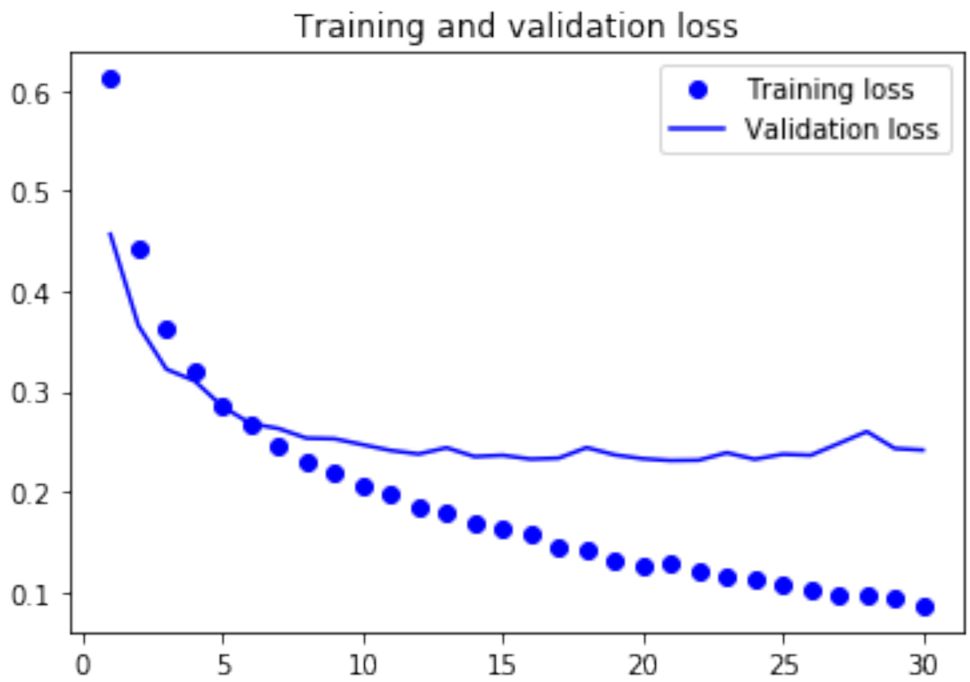
```
Train on 2000 samples, validate on 1000 samples
Epoch 1/30
2000/2000 [==============================] - 3s 2ms/step - loss: 0
.6125 - acc: 0.6570 - val_loss: 0.4569 - val_acc: 0.8120
Epoch 2/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.4415 - acc: 0.7960 - val_loss: 0.3657 - val_acc: 0.8630
Epoch 3/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.3623 - acc: 0.8470 - val_loss: 0.3222 - val_acc: 0.8840
Epoch 4/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.3208 - acc: 0.8675 - val_loss: 0.3107 - val_acc: 0.8720
Epoch 5/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.2855 - acc: 0.8780 - val_loss: 0.2849 - val_acc: 0.8850
Epoch 6/30
2000/2000 [==============================] - 3s 2ms/step - loss: 0
.2661 - acc: 0.9015 - val_loss: 0.2680 - val_acc: 0.8970
Epoch 7/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.2456 - acc: 0.9010 - val_loss: 0.2630 - val_acc: 0.8990
Epoch 8/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.2307 - acc: 0.9145 - val_loss: 0.2534 - val_acc: 0.9010
Epoch 9/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.2183 - acc: 0.9215 - val_loss: 0.2527 - val_acc: 0.9010
Epoch 10/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.2052 - acc: 0.9255 - val_loss: 0.2469 - val_acc: 0.9020
Epoch 11/30
2000/2000 [==============================] - 3s 2ms/step - loss: 0
.1978 - acc: 0.9275 - val_loss: 0.2411 - val_acc: 0.9030
Epoch 12/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
```

```
.1843 - acc: 0.9330 - val_loss: 0.2375 - val_acc: 0.9040
Epoch 13/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1779 - acc: 0.9320 - val_loss: 0.2438 - val_acc: 0.9020
Epoch 14/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1680 - acc: 0.9375 - val_loss: 0.2350 - val_acc: 0.9040
Epoch 15/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1630 - acc: 0.9430 - val_loss: 0.2363 - val_acc: 0.9000
Epoch 16/30
2000/2000 [==============================] - 3s 2ms/step - loss: 0
.1585 - acc: 0.9470 - val_loss: 0.2325 - val_acc: 0.9060
Epoch 17/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1446 - acc: 0.9490 - val_loss: 0.2334 - val_acc: 0.9030
Epoch 18/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1421 - acc: 0.9550 - val_loss: 0.2439 - val_acc: 0.9030
Epoch 19/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1299 - acc: 0.9540 - val_loss: 0.2367 - val_acc: 0.9060
Epoch 20/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1266 - acc: 0.9580 - val_loss: 0.2329 - val_acc: 0.9030
Epoch 21/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1277 - acc: 0.9590 - val_loss: 0.2310 - val_acc: 0.8990
Epoch 22/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1199 - acc: 0.9615 - val_loss: 0.2314 - val_acc: 0.9000
Epoch 23/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1162 - acc: 0.9600 - val_loss: 0.2388 - val_acc: 0.9010
Epoch 24/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1138 - acc: 0.9615 - val_loss: 0.2323 - val_acc: 0.9010
Epoch 25/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1074 - acc: 0.9645 - val_loss: 0.2373 - val_acc: 0.9040
Epoch 26/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.1022 - acc: 0.9685 - val_loss: 0.2363 - val_acc: 0.9010
Epoch 27/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.0963 - acc: 0.9670 - val_loss: 0.2481 - val_acc: 0.9030
Epoch 28/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.0954 - acc: 0.9675 - val_loss: 0.2600 - val_acc: 0.8890
Epoch 29/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.0929 - acc: 0.9680 - val_loss: 0.2430 - val_acc: 0.9090
Epoch 30/30
2000/2000 [==============================] - 3s 1ms/step - loss: 0
.0862 - acc: 0.9760 - val_loss: 0.2415 - val_acc: 0.9010
```

In [51]:

```
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.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.legend()

plt.show()
```





In [53]:

```
from keras import models
from keras import layers

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

In [54]:

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688  
_________________________________________________________________
flatten_3 (Flatten)          (None, 8192)              0         
_________________________________________________________________
dense_9 (Dense)              (None, 256)               2097408   
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 257       
=================================================================
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0
_________________________________________________________________
```

Freezeの理由は、ただ訓練コストだけじゃなく、Because the Dense layers on top are randomly initialized, very large weight updates would be propagated through network, effectively destroying the representations previously learned.

In [55]:

```
print('This is the number of trainable weights '
     'before freezing the conv base:', len(model.trainable_weights))
```

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

In [56]:

```
conv_base.trainable = False
```

In [57]:

```
print('This is the number of trainable weights '
     'after freezing the conv base:', len(model.trainable_weights))
```

```
This is the number of trainable weights after freezing the conv ba
se: 4
```

4の意味：four weight tensors: two per layer (the main weight matrix and the bias vector)

Note: This technique is so expensive that you should only attempt it if you have access to a GPU - it's absolutely intractable on CPU. If you can't run your code on GPU, then the previous technique is the way to go.

In [58]:

```
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers

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')

model.compile(loss='binary_crossentropy',
             optimizer=optimizers.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)
```

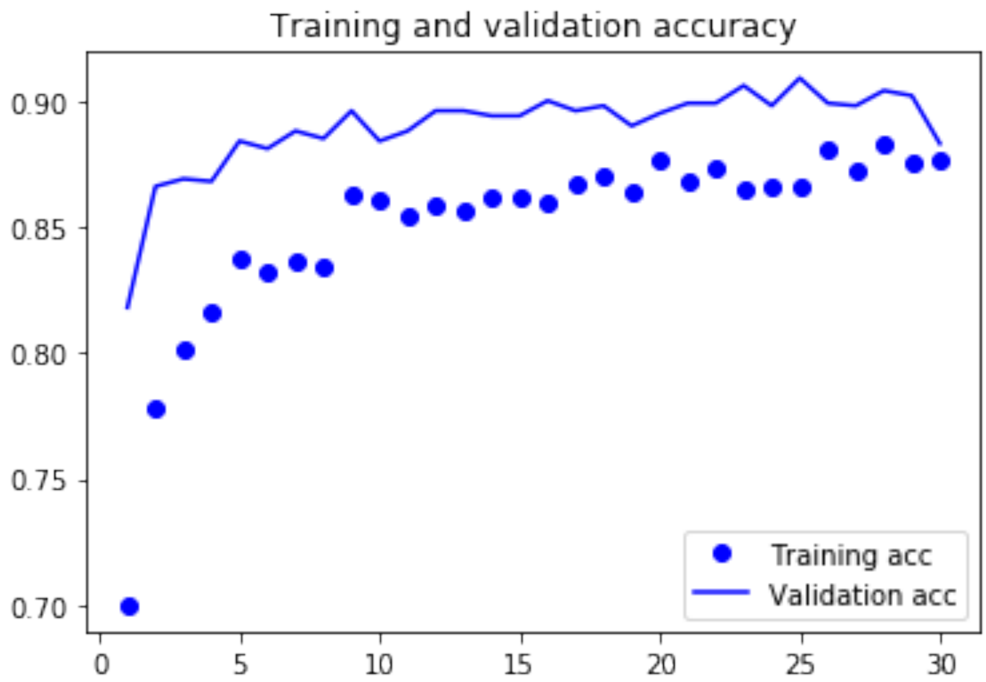
```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/30
100/100 [==============================] - 561s 6s/step - loss: 0.
5837 - acc: 0.7000 - val_loss: 0.4416 - val_acc: 0.8180
Epoch 2/30
100/100 [==============================] - 545s 5s/step - loss: 0.
4811 - acc: 0.7780 - val_loss: 0.3598 - val_acc: 0.8660
Epoch 3/30
100/100 [==============================] - 548s 5s/step - loss: 0.
4363 - acc: 0.8010 - val_loss: 0.3266 - val_acc: 0.8690
```

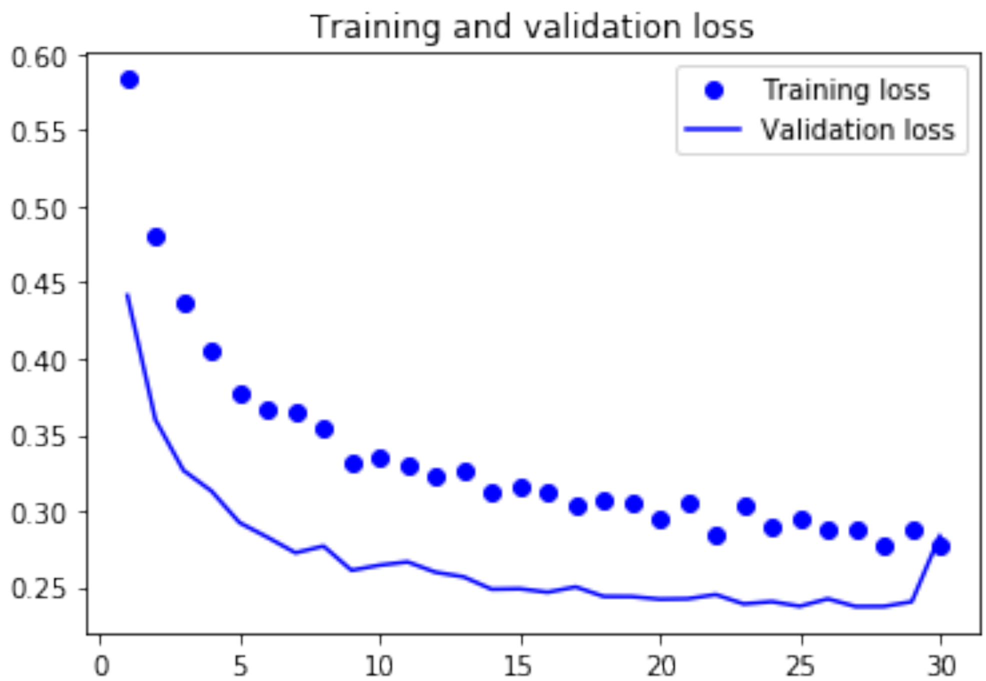
```
Epoch 4/30
100/100 [==============================] - 551s 6s/step - loss: 0.
4054 - acc: 0.8165 - val_loss: 0.3130 - val_acc: 0.8680
Epoch 5/30
100/100 [==============================] - 562s 6s/step - loss: 0.
3780 - acc: 0.8375 - val_loss: 0.2923 - val_acc: 0.8840
Epoch 6/30
100/100 [==============================] - 568s 6s/step - loss: 0.
3674 - acc: 0.8320 - val_loss: 0.2827 - val_acc: 0.8810
Epoch 7/30
100/100 [==============================] - 546s 5s/step - loss: 0.
3655 - acc: 0.8365 - val_loss: 0.2724 - val_acc: 0.8880
Epoch 8/30
100/100 [==============================] - 542s 5s/step - loss: 0.
3545 - acc: 0.8340 - val_loss: 0.2768 - val_acc: 0.8850
Epoch 9/30
100/100 [==============================] - 542s 5s/step - loss: 0.
3311 - acc: 0.8625 - val_loss: 0.2609 - val_acc: 0.8960
Epoch 10/30
100/100 [==============================] - 659s 7s/step - loss: 0.
3345 - acc: 0.8610 - val_loss: 0.2643 - val_acc: 0.8840
Epoch 11/30
100/100 [==============================] - 672s 7s/step - loss: 0.
3302 - acc: 0.8545 - val_loss: 0.2665 - val_acc: 0.8880
Epoch 12/30
100/100 [==============================] - 617s 6s/step - loss: 0.
3234 - acc: 0.8590 - val_loss: 0.2597 - val_acc: 0.8960
Epoch 13/30
100/100 [==============================] - 543s 5s/step - loss: 0.
3260 - acc: 0.8560 - val_loss: 0.2567 - val_acc: 0.8960
Epoch 14/30
100/100 [==============================] - 543s 5s/step - loss: 0.
3129 - acc: 0.8615 - val_loss: 0.2485 - val_acc: 0.8940
Epoch 15/30
100/100 [==============================] - 547s 5s/step - loss: 0.
3154 - acc: 0.8620 - val_loss: 0.2488 - val_acc: 0.8940
Epoch 16/30
100/100 [==============================] - 562s 6s/step - loss: 0.
3119 - acc: 0.8600 - val_loss: 0.2465 - val_acc: 0.9000
Epoch 17/30
100/100 [==============================] - 542s 5s/step - loss: 0.
3038 - acc: 0.8665 - val_loss: 0.2500 - val_acc: 0.8960
Epoch 18/30
100/100 [==============================] - 541s 5s/step - loss: 0.
3065 - acc: 0.8700 - val_loss: 0.2438 - val_acc: 0.8980
Epoch 19/30
100/100 [==============================] - 543s 5s/step - loss: 0.
3048 - acc: 0.8640 - val_loss: 0.2437 - val_acc: 0.8900
Epoch 20/30
100/100 [==============================] - 542s 5s/step - loss: 0.
2948 - acc: 0.8770 - val_loss: 0.2420 - val_acc: 0.8950
Epoch 21/30
100/100 [==============================] - 542s 5s/step - loss: 0.
3057 - acc: 0.8675 - val_loss: 0.2422 - val_acc: 0.8990
Epoch 22/30
100/100 [==============================] - 542s 5s/step - loss: 0.
2836 - acc: 0.8735 - val_loss: 0.2450 - val_acc: 0.8990
```

```
Epoch 23/30
100/100 [==============================] - 541s 5s/step - loss: 0.
3033 - acc: 0.8645 - val_loss: 0.2388 - val_acc: 0.9060
Epoch 24/30
100/100 [==============================] - 544s 5s/step - loss: 0.
2888 - acc: 0.8655 - val_loss: 0.2404 - val_acc: 0.8980
Epoch 25/30
100/100 [==============================] - 542s 5s/step - loss: 0.
2948 - acc: 0.8655 - val_loss: 0.2373 - val_acc: 0.9090
Epoch 26/30
100/100 [==============================] - 541s 5s/step - loss: 0.
2884 - acc: 0.8805 - val_loss: 0.2422 - val_acc: 0.8990
Epoch 27/30
100/100 [==============================] - 542s 5s/step - loss: 0.
2871 - acc: 0.8725 - val_loss: 0.2371 - val_acc: 0.8980
Epoch 28/30
100/100 [==============================] - 542s 5s/step - loss: 0.
2780 - acc: 0.8825 - val_loss: 0.2372 - val_acc: 0.9040
Epoch 29/30
100/100 [==============================] - 542s 5s/step - loss: 0.
2872 - acc: 0.8755 - val_loss: 0.2404 - val_acc: 0.9020
Epoch 30/30
100/100 [==============================] - 542s 5s/step - loss: 0.
2777 - acc: 0.8770 - val_loss: 0.2836 - val_acc: 0.8830
```

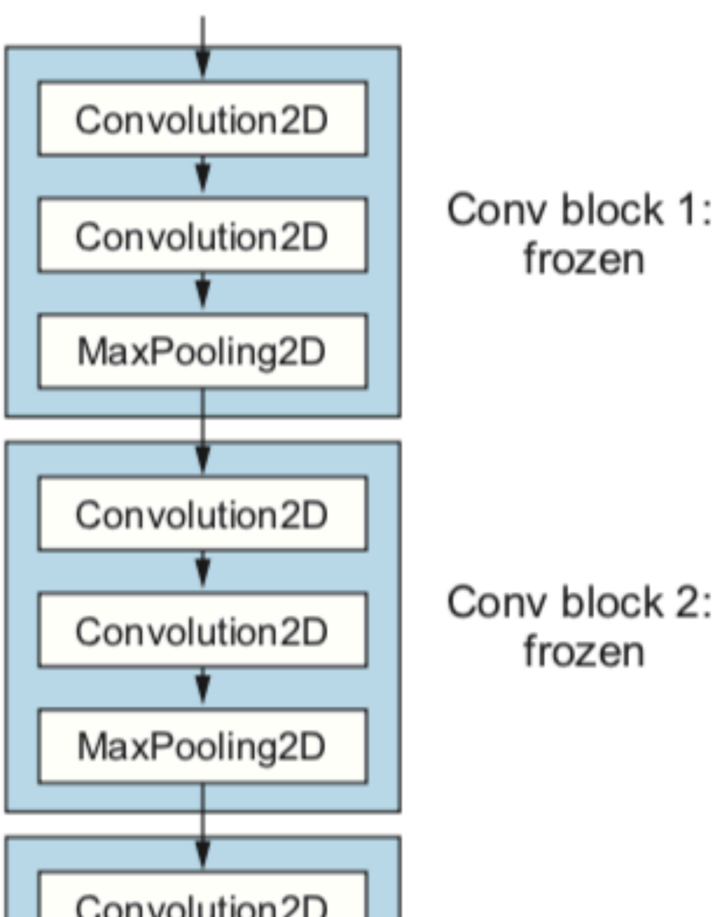
In [60]:

```
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.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.legend()

plt.show()
```





この結果は本に書いてあるのと違う。。。Validation accやlossは変わっていない。ただTraining accやlossが減った。でも、この訓練したモデルのdensely connected classifierはfine tuningで使う。



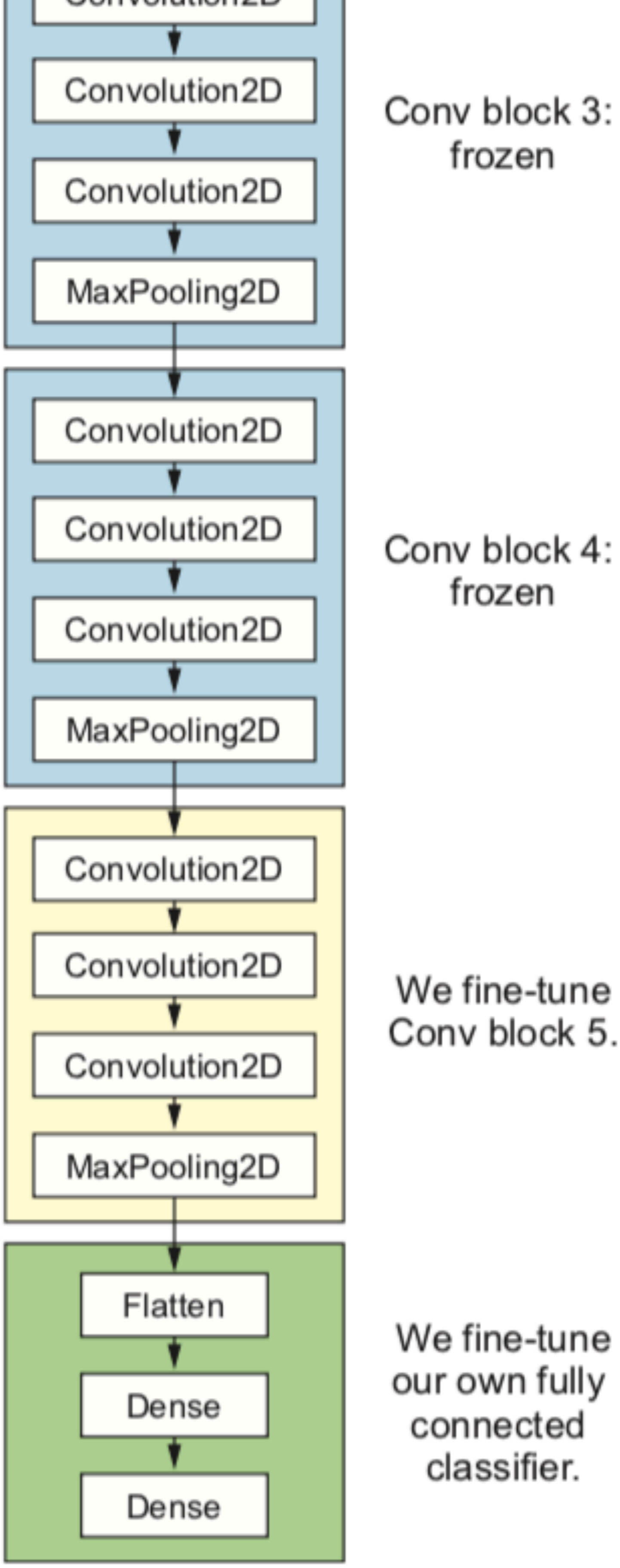


Figure 5.19 Fine-tuning the last convolutional block of the VGG16 network

Step6: Fine-tuning

1. Add your custom network on top of an already-trained base network
2. Freeze the base network
3. Train the part you added
4. Unfreeze some layers in the base network
5. Jointly train both these layers and the part you added

なぜもっと多い層をfine-tuneしない？ なぜ全ての層をfine-tuneしない？

1. Earlier layers in the convolutional base encode more-generic, reusable features, whereas layers higher up encode more-specialized features. It's more useful to fine-tune the more specialized features, because these are the ones that need to be repurposed on your new problem. There would be fast-decreasing returns in fine-tuning lower layers.
2. The more parameters you're training, the more you're at risk of overfitting. The convolutional base has 15 million parameters, so it would be risky to attempt to train it on you small dataset.

In [61]:

```
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
```

In [62]:

```
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.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)
```

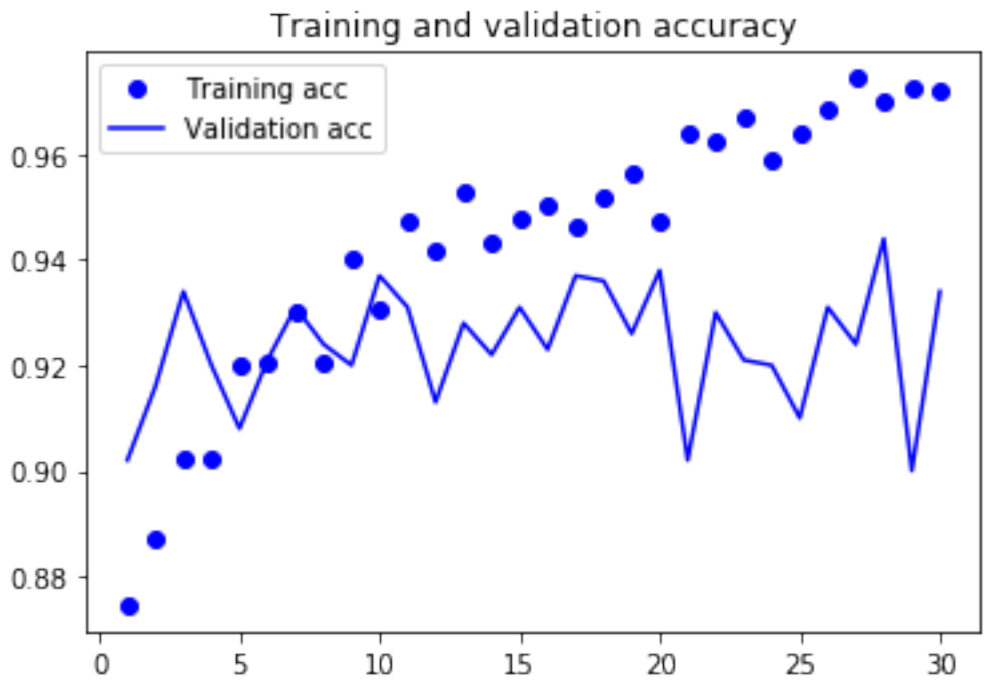
```
Epoch 1/30
100/100 [==============================] - 628s 6s/step - loss: 0.
2812 - acc: 0.8745 - val_loss: 0.2527 - val_acc: 0.9020
Epoch 2/30
100/100 [==============================] - 613s 6s/step - loss: 0.
2561 - acc: 0.8870 - val_loss: 0.2104 - val_acc: 0.9160
Epoch 3/30
100/100 [==============================] - 624s 6s/step - loss: 0.
2377 - acc: 0.9025 - val_loss: 0.1978 - val_acc: 0.9340
Epoch 4/30
100/100 [==============================] - 610s 6s/step - loss: 0.
2226 - acc: 0.9025 - val_loss: 0.2046 - val_acc: 0.9200
Epoch 5/30
100/100 [==============================] - 608s 6s/step - loss: 0.
1972 - acc: 0.9200 - val_loss: 0.2378 - val_acc: 0.9080
Epoch 6/30
100/100 [==============================] - 684s 7s/step - loss: 0.
1957 - acc: 0.9205 - val_loss: 0.2271 - val_acc: 0.9210
```

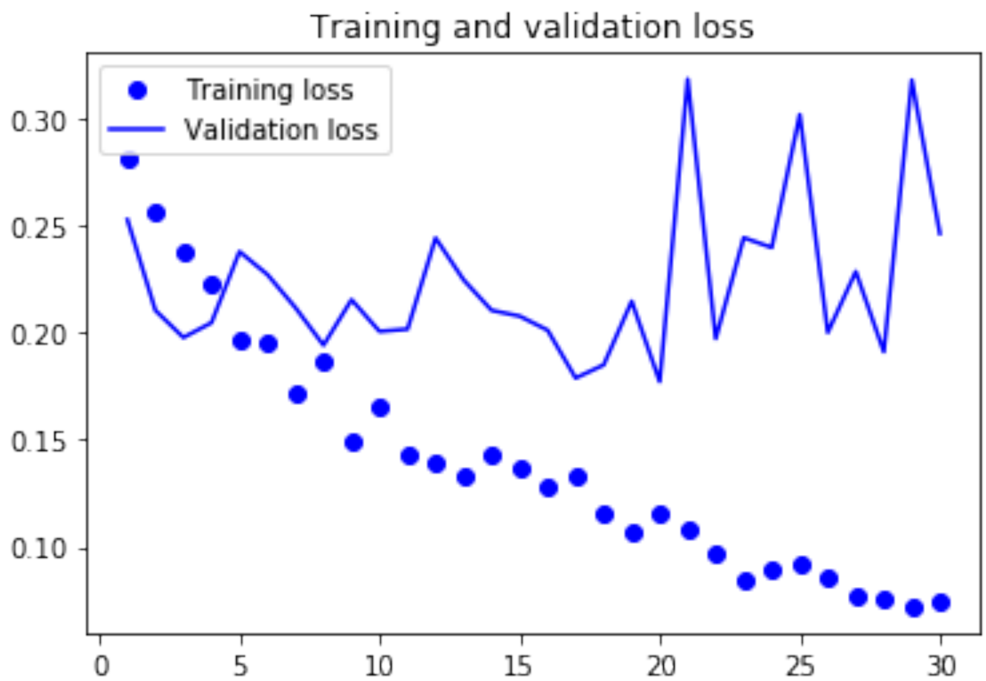
```
Epoch 7/30
100/100 [==============================] - 682s 7s/step - loss: 0.
1722 - acc: 0.9300 - val_loss: 0.2117 - val_acc: 0.9310
Epoch 8/30
100/100 [==============================] - 622s 6s/step - loss: 0.
1862 - acc: 0.9205 - val_loss: 0.1942 - val_acc: 0.9240
Epoch 9/30
100/100 [==============================] - 665s 7s/step - loss: 0.
1488 - acc: 0.9400 - val_loss: 0.2154 - val_acc: 0.9200
Epoch 10/30
100/100 [==============================] - 656s 7s/step - loss: 0.
1652 - acc: 0.9305 - val_loss: 0.2006 - val_acc: 0.9370
Epoch 11/30
100/100 [==============================] - 634s 6s/step - loss: 0.
1430 - acc: 0.9475 - val_loss: 0.2017 - val_acc: 0.9310
Epoch 12/30
100/100 [==============================] - 640s 6s/step - loss: 0.
1399 - acc: 0.9415 - val_loss: 0.2442 - val_acc: 0.9130
Epoch 13/30
100/100 [==============================] - 633s 6s/step - loss: 0.
1337 - acc: 0.9530 - val_loss: 0.2247 - val_acc: 0.9280
Epoch 14/30
100/100 [==============================] - 614s 6s/step - loss: 0.
1434 - acc: 0.9435 - val_loss: 0.2104 - val_acc: 0.9220
Epoch 15/30
100/100 [==============================] - 609s 6s/step - loss: 0.
1367 - acc: 0.9480 - val_loss: 0.2077 - val_acc: 0.9310
Epoch 16/30
100/100 [==============================] - 611s 6s/step - loss: 0.
1276 - acc: 0.9505 - val_loss: 0.2012 - val_acc: 0.9230
Epoch 17/30
100/100 [==============================] - 611s 6s/step - loss: 0.
1330 - acc: 0.9465 - val_loss: 0.1789 - val_acc: 0.9370
Epoch 18/30
100/100 [==============================] - 609s 6s/step - loss: 0.
1163 - acc: 0.9520 - val_loss: 0.1850 - val_acc: 0.9360
Epoch 19/30
100/100 [==============================] - 637s 6s/step - loss: 0.
1065 - acc: 0.9565 - val_loss: 0.2147 - val_acc: 0.9260
Epoch 20/30
100/100 [==============================] - 612s 6s/step - loss: 0.
1162 - acc: 0.9475 - val_loss: 0.1772 - val_acc: 0.9380
Epoch 21/30
100/100 [==============================] - 643s 6s/step - loss: 0.
1080 - acc: 0.9640 - val_loss: 0.3183 - val_acc: 0.9020
Epoch 22/30
100/100 [==============================] - 648s 6s/step - loss: 0.
0969 - acc: 0.9625 - val_loss: 0.1973 - val_acc: 0.9300
Epoch 23/30
100/100 [==============================] - 658s 7s/step - loss: 0.
0847 - acc: 0.9670 - val_loss: 0.2443 - val_acc: 0.9210
Epoch 24/30
100/100 [==============================] - 611s 6s/step - loss: 0.
0897 - acc: 0.9590 - val_loss: 0.2397 - val_acc: 0.9200
Epoch 25/30
100/100 [==============================] - 608s 6s/step - loss: 0.
0925 - acc: 0.9640 - val_loss: 0.3017 - val_acc: 0.9100
```

```
Epoch 26/30
100/100 [==============================] - 608s 6s/step - loss: 0.
0863 - acc: 0.9685 - val_loss: 0.2001 - val_acc: 0.9310
Epoch 27/30
100/100 [==============================] - 607s 6s/step - loss: 0.
0769 - acc: 0.9745 - val_loss: 0.2285 - val_acc: 0.9240
Epoch 28/30
100/100 [==============================] - 608s 6s/step - loss: 0.
0761 - acc: 0.9700 - val_loss: 0.1911 - val_acc: 0.9440
Epoch 29/30
100/100 [==============================] - 608s 6s/step - loss: 0.
0723 - acc: 0.9725 - val_loss: 0.3179 - val_acc: 0.9000
Epoch 30/30
100/100 [==============================] - 608s 6s/step - loss: 0.
0747 - acc: 0.9720 - val_loss: 0.2463 - val_acc: 0.9340
```

In [63]:

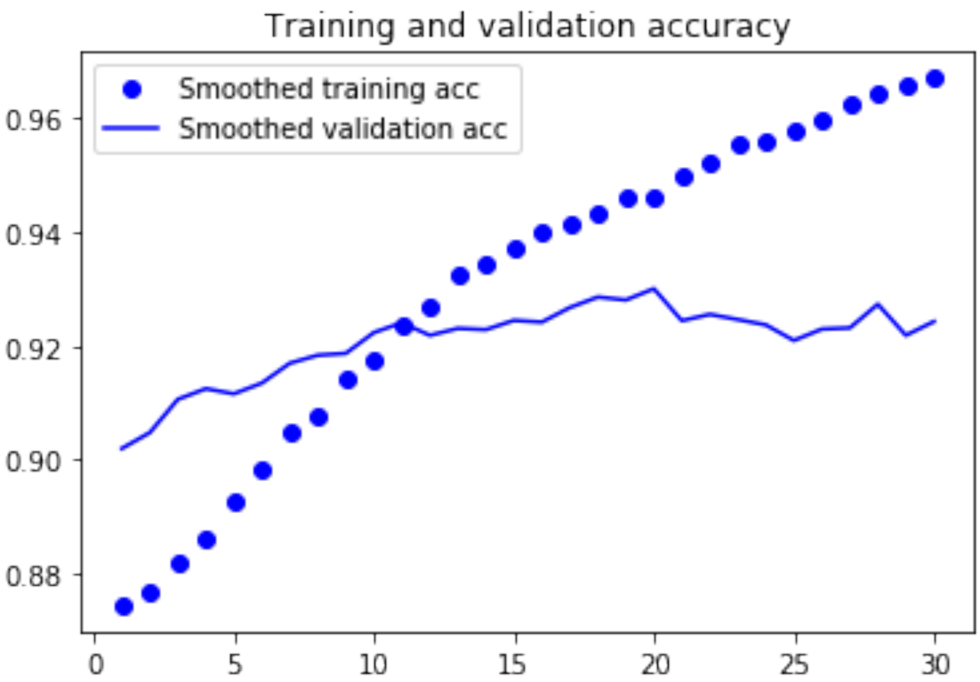
```python
import matplotlib.pyplot as plt

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.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.legend()

plt.show()
```

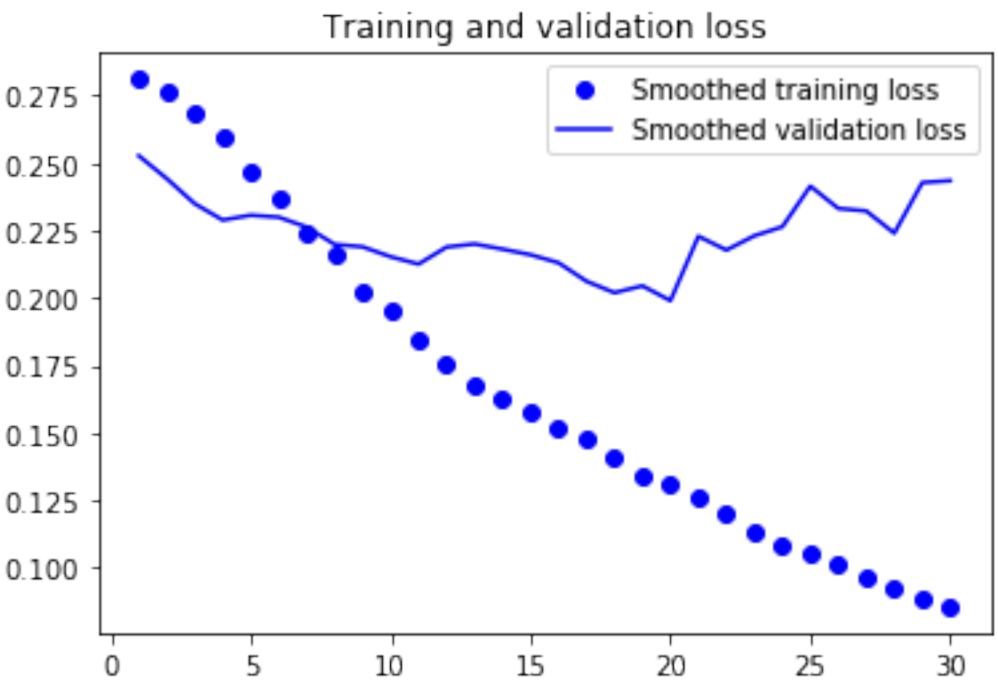

Training and validation accuracy
Training acc
Validation acc
0.96
0.94
0.92
0.90
0.88
0
5
10
15
20
25
30


Training and validation loss
Training loss
Validation loss
0.30
0.25
0.20
0.15
0.10
0
5
10
15
20
25
30

In [65]:

```
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

plt.plot(epochs, smooth_curve(acc), 'bo', label='Smoothed training acc')
plt.plot(epochs, smooth_curve(val_acc), 'b', label='Smoothed validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, smooth_curve(loss), 'bo', label='Smoothed training loss')
plt.plot(epochs, smooth_curve(val_loss), 'b', label='Smoothed validation loss'
)
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





You may wonder, how could accuracy stay stable or improve if the loss isn't decreasing? The answer is simple: what you display is an average of pointwise loss values; but what matters for accuracy is the distribution of the loss values, not their average, because accuracy is the result of a binary thresholding of the class probability predicted by the model.

In [66]:

```
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: 0.928999993801117
```