以少量資料集從頭訓練一個卷積神經網路

Data : https://www.kaggle.com/c/dogs-vs-cats/data

In [1]:
```
import os, shutil
```

In [2]:
```
# 解壓縮資料夾所在的目錄路徑
original_dataset_dir = r'D:\Anaconda3\Scripts\5 上課資料\電腦視覺與人機互動\20191031HW\dogs-vs-cats\train\train'
# 用來儲存少量資料集的目錄位置
base_dir = r'D:\Anaconda3\Scripts\5 上課資料\電腦視覺與人機互動\20191031HW\cats_and_dogs_small'
# 如果目錄不存在，才建立目錄
if not os.path.isdir(base_dir): os.mkdir(base_dir)
```

In [3]:
```
# 分拆成訓練、驗證與測試目錄位置
train_dir = os.path.join(base_dir, 'train')
if not os.path.isdir(train_dir): os.mkdir(train_dir)

validation_dir = os.path.join(base_dir, 'validation')
if not os.path.isdir(validation_dir): os.mkdir(validation_dir)

test_dir = os.path.join(base_dir, 'test')
if not os.path.isdir(test_dir): os.mkdir(test_dir)
```

In [4]:
```
# 用來訓練貓圖片的目錄位置
train_cats_dir = os.path.join(train_dir, 'cats')
if not os.path.isdir(train_cats_dir):
    os.mkdir(train_cats_dir)

# 用來訓練狗圖片的目錄位置
train_dogs_dir = os.path.join(train_dir, 'dogs')
if not os.path.isdir(train_dogs_dir):
    os.mkdir(train_dogs_dir)

# 用來驗證貓圖片的目錄位置
validation_cats_dir = os.path.join(validation_dir, 'cats')
if not os.path.isdir(validation_cats_dir):
    os.mkdir(validation_cats_dir)

# 用來驗證狗圖片的目錄位置
validation_dogs_dir = os.path.join(validation_dir, 'dogs')
if not os.path.isdir(validation_dogs_dir):
    os.mkdir(validation_dogs_dir)

# 用來測試貓圖片的目錄位置
test_cats_dir = os.path.join(test_dir, 'cats')
if not os.path.isdir(test_cats_dir):
    os.mkdir(test_cats_dir)

# 用來測試狗圖片的目錄位置
test_dogs_dir = os.path.join(test_dir, 'dogs')
if not os.path.isdir(test_dogs_dir):
    os.mkdir(test_dogs_dir)
```

In [5]:
```
# 複製前面 1000 張貓圖片到 train_cats_dir 訓練目錄
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)

# 複製下 500 張貓圖片到 validation_cats_dir 驗證目錄
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)

# 複製下 500 張貓圖片到 test_cats_dir 測試目錄
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)
```

In [6]:
```
# 複製前面 1000 張狗圖片到 train_dogs_dir 訓練目錄
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)

# 複製下 500 張狗圖片到 validation_dogs_dir 驗證目錄
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)

# 複製下 500 張狗圖片到 test_dogs_dir 測試目錄
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)

print('複製完成')
```

```
複製完成
```

In [8]:
```
# 計算每個訓練/驗證/測試分組中的圖片數量
print('訓練用的貓照片張數:', len(os.listdir(train_cats_dir)))
print('訓練用的狗照片張數:', len(os.listdir(train_dogs_dir)))
print('驗證用的貓照片張數:', len(os.listdir(validation_cats_dir)))
print('驗證用的狗照片張數:', len(os.listdir(validation_dogs_dir)))
print('測試用的貓照片張數:', len(os.listdir(test_cats_dir)))
```

```
print('測試用的狗照片張數:', len(os.listdir(test_dogs_dir)))
```

```
訓練用的貓照片張數: 1000
訓練用的狗照片張數: 1000
驗證用的貓照片張數: 500
驗證用的狗照片張數: 500
測試用的貓照片張數: 500
測試用的狗照片張數: 500
```

In [9]:
```
# CNN
from keras import layers
from keras import models
from keras import optimizers
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

```
D:\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float`
to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [10]:
```
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.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_4 (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 [11]:
```
model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSprop(lr=1e-4), metrics=['acc'])
```

In [12]:
```
#設定訓練、測試資料的 Python 產生器，並將圖片像素值依 1/255 比例重新壓縮到 [0, 1]
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,               # 目標目錄
    target_size=(150, 150),  # 調整所有影像大小成 150x150
    batch_size=20,
    class_mode='binary')     # 因為使用二元交叉熵 binary_crossentropy 作為損失值，所以需要二位元標籤


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 [13]:
```
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,)
```

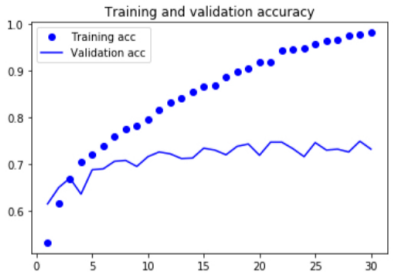
In [14]:
```
history = model.fit_generator(
    train_generator,
    steps_per_epoch=100,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=50)
```

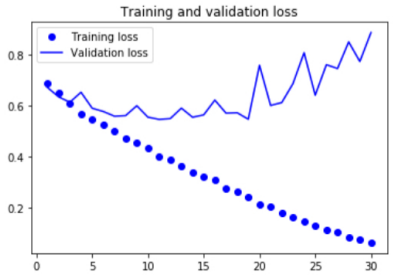
```
Epoch 1/30
100/100 [==============================] - 40s 396ms/step - loss: 0.6881 - acc: 0.5325 - val_loss: 0.6696 - val_acc: 0.6150
Epoch 2/30
100/100 [==============================] - 12s 117ms/step - loss: 0.6498 - acc: 0.6165 - val_loss: 0.6329 - val_acc: 0.6500
Epoch 3/30
100/100 [==============================] - 12s 122ms/step - loss: 0.6061 - acc: 0.6690 - val_loss: 0.6119 - val_acc: 0.6700
Epoch 4/30
100/100 [==============================] - 12s 120ms/step - loss: 0.5673 - acc: 0.7050 - val_loss: 0.6512 - val_acc: 0.6360
Epoch 5/30
100/100 [==============================] - 13s 127ms/step - loss: 0.5456 - acc: 0.7195 - val_loss: 0.5886 - val_acc: 0.6880
Epoch 6/30
100/100 [==============================] - 13s 125ms/step - loss: 0.5258 - acc: 0.7390 - val_loss: 0.5756 - val_acc: 0.6900
Epoch 7/30
100/100 [==============================] - 13s 128ms/step - loss: 0.4983 - acc: 0.7595 - val_loss: 0.5572 - val_acc: 0.7060
Epoch 8/30
100/100 [==============================] - 12s 124ms/step - loss: 0.4716 - acc: 0.7760 - val_loss: 0.5596 - val_acc: 0.7080
```

```
Epoch 9/30
100/100 [==============================] - 13s 129ms/step - loss: 0.4550 - acc: 0.7810 - val_loss: 0.5983 - val_acc: 0.6950
Epoch 10/30
100/100 [==============================] - 13s 129ms/step - loss: 0.4338 - acc: 0.7965 - val_loss: 0.5533 - val_acc: 0.7160
Epoch 11/30
100/100 [==============================] - 13s 131ms/step - loss: 0.4007 - acc: 0.8160 - val_loss: 0.5448 - val_acc: 0.7260
Epoch 12/30
100/100 [==============================] - 13s 127ms/step - loss: 0.3885 - acc: 0.8320 - val_loss: 0.5478 - val_acc: 0.7220
Epoch 13/30
100/100 [==============================] - 13s 128ms/step - loss: 0.3604 - acc: 0.8400 - val_loss: 0.5888 - val_acc: 0.7120
Epoch 14/30
100/100 [==============================] - 13s 129ms/step - loss: 0.3353 - acc: 0.8540 - val_loss: 0.5532 - val_acc: 0.7130
Epoch 15/30
100/100 [==============================] - 13s 131ms/step - loss: 0.3202 - acc: 0.8650 - val_loss: 0.5624 - val_acc: 0.7340
Epoch 16/30
100/100 [==============================] - 14s 140ms/step - loss: 0.3075 - acc: 0.8685 - val_loss: 0.6204 - val_acc: 0.7300
Epoch 17/30
100/100 [==============================] - 13s 133ms/step - loss: 0.2749 - acc: 0.8870 - val_loss: 0.5692 - val_acc: 0.7200
Epoch 18/30
100/100 [==============================] - 13s 132ms/step - loss: 0.2629 - acc: 0.8965 - val_loss: 0.5708 - val_acc: 0.7380
Epoch 19/30
100/100 [==============================] - 14s 136ms/step - loss: 0.2390 - acc: 0.9035 - val_loss: 0.5453 - val_acc: 0.7430
Epoch 20/30
100/100 [==============================] - 13s 128ms/step - loss: 0.2133 - acc: 0.9190 - val_loss: 0.7569 - val_acc: 0.7190
Epoch 21/30
100/100 [==============================] - 13s 132ms/step - loss: 0.2031 - acc: 0.9180 - val_loss: 0.5993 - val_acc: 0.7470
Epoch 22/30
100/100 [==============================] - 13s 130ms/step - loss: 0.1774 - acc: 0.9430 - val_loss: 0.6107 - val_acc: 0.7470
Epoch 23/30
100/100 [==============================] - 13s 130ms/step - loss: 0.1627 - acc: 0.9455 - val_loss: 0.6846 - val_acc: 0.7330
Epoch 24/30
100/100 [==============================] - 13s 129ms/step - loss: 0.1437 - acc: 0.9485 - val_loss: 0.8063 - val_acc: 0.7160
Epoch 25/30
100/100 [==============================] - 13s 130ms/step - loss: 0.1301 - acc: 0.9570 - val_loss: 0.6396 - val_acc: 0.7460
Epoch 26/30
100/100 [==============================] - 13s 135ms/step - loss: 0.1094 - acc: 0.9630 - val_loss: 0.7594 - val_acc: 0.7300
Epoch 27/30
100/100 [==============================] - 13s 129ms/step - loss: 0.1025 - acc: 0.9665 - val_loss: 0.7442 - val_acc: 0.7320
Epoch 28/30
100/100 [==============================] - 13s 129ms/step - loss: 0.0843 - acc: 0.9745 - val_loss: 0.8488 - val_acc: 0.7260
Epoch 29/30
100/100 [==============================] - 14s 139ms/step - loss: 0.0757 - acc: 0.9775 - val_loss: 0.7718 - val_acc: 0.7490
Epoch 30/30
100/100 [==============================] - 14s 136ms/step - loss: 0.0618 - acc: 0.9810 - val_loss: 0.8855 - val_acc: 0.7320
```

In [15]:
```
model.save('cats_and_dogs_small_1.h5')
```

In [16]:
```
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()
```

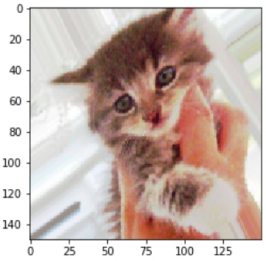


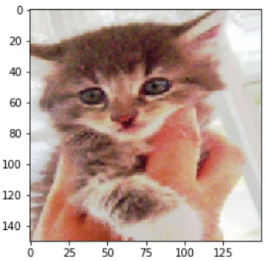


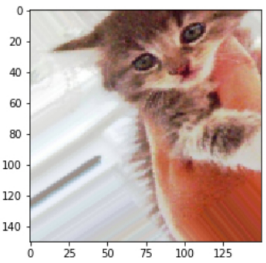
In [17]:
```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```

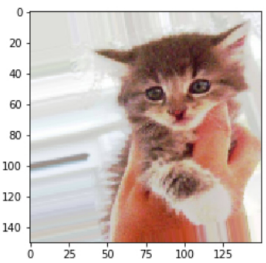
In [18]:
```
fnames = [os.path.join(train_cats_dir, fname) for
    fname in os.listdir(train_cats_dir)]

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 [19]:

```
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))  #  加入 Dropout 層 (丟棄 50 %)
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'])
```

In [20]:

```
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) # 請注意！驗證資料不應該擴充!!!

train_generator = train_datagen.flow_from_directory(
    train_dir,    # 目標目錄
    target_size=(150, 150), # 所有圖像大小調整成 150x150
    batch_size=32,
    class_mode='binary') # 因為使用二元交叉熵 binary_crossentropy 作為損失，所以需要二元標籤


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

# 訓練
history = model.fit_generator(
    train_generator,
```

```
    steps_per_epoch=100,
    epochs=100,
    validation_data=validation_generator,
    validation_steps=50)
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/100
100/100 [==============================] - 25s 249ms/step - loss: 0.6920 - acc: 0.5181 - val_loss: 0.6940 - val_acc: 0.5178
Epoch 2/100
100/100 [==============================] - 25s 250ms/step - loss: 0.6806 - acc: 0.5634 - val_loss: 0.6593 - val_acc: 0.6057
Epoch 3/100
100/100 [==============================] - 28s 282ms/step - loss: 0.6648 - acc: 0.5991 - val_loss: 0.6444 - val_acc: 0.6098
Epoch 4/100
100/100 [==============================] - 33s 334ms/step - loss: 0.6446 - acc: 0.6266 - val_loss: 0.6125 - val_acc: 0.6514
Epoch 5/100
100/100 [==============================] - 32s 318ms/step - loss: 0.6279 - acc: 0.6406 - val_loss: 0.6117 - val_acc: 0.6447
Epoch 6/100
100/100 [==============================] - 35s 353ms/step - loss: 0.6164 - acc: 0.6566 - val_loss: 0.6141 - val_acc: 0.6598
Epoch 7/100
100/100 [==============================] - 33s 329ms/step - loss: 0.6040 - acc: 0.6681 - val_loss: 0.5915 - val_acc: 0.6758
Epoch 8/100
100/100 [==============================] - 34s 338ms/step - loss: 0.5979 - acc: 0.6784 - val_loss: 0.5733 - val_acc: 0.6798
Epoch 9/100
100/100 [==============================] - 33s 327ms/step - loss: 0.5877 - acc: 0.6769 - val_loss: 0.5847 - val_acc: 0.6856
Epoch 10/100
100/100 [==============================] - 33s 328ms/step - loss: 0.5767 - acc: 0.6966 - val_loss: 0.5566 - val_acc: 0.7037
Epoch 11/100
100/100 [==============================] - 35s 348ms/step - loss: 0.5704 - acc: 0.6981 - val_loss: 0.5555 - val_acc: 0.6991
Epoch 12/100
100/100 [==============================] - 33s 332ms/step - loss: 0.5576 - acc: 0.7259 - val_loss: 0.5541 - val_acc: 0.7030
Epoch 13/100
100/100 [==============================] - 34s 344ms/step - loss: 0.5593 - acc: 0.7097 - val_loss: 0.5428 - val_acc: 0.7133
Epoch 14/100
100/100 [==============================] - 33s 331ms/step - loss: 0.5435 - acc: 0.7200 - val_loss: 0.5455 - val_acc: 0.7157
Epoch 15/100
100/100 [==============================] - 34s 341ms/step - loss: 0.5382 - acc: 0.7225 - val_loss: 0.5090 - val_acc: 0.7416
Epoch 16/100
100/100 [==============================] - 34s 337ms/step - loss: 0.5442 - acc: 0.7225 - val_loss: 0.5080 - val_acc: 0.7358
Epoch 17/100
100/100 [==============================] - 34s 344ms/step - loss: 0.5352 - acc: 0.7237 - val_loss: 0.5171 - val_acc: 0.7398
Epoch 18/100
100/100 [==============================] - 36s 362ms/step - loss: 0.5345 - acc: 0.7244 - val_loss: 0.6186 - val_acc: 0.6894
Epoch 19/100
100/100 [==============================] - 34s 339ms/step - loss: 0.5268 - acc: 0.7334 - val_loss: 0.5099 - val_acc: 0.7398
Epoch 20/100
100/100 [==============================] - 35s 349ms/step - loss: 0.5220 - acc: 0.7303 - val_loss: 0.5441 - val_acc: 0.7287
Epoch 21/100
100/100 [==============================] - 33s 329ms/step - loss: 0.5285 - acc: 0.7347 - val_loss: 0.4949 - val_acc: 0.7684
Epoch 22/100
100/100 [==============================] - 34s 340ms/step - loss: 0.5094 - acc: 0.7488 - val_loss: 0.5207 - val_acc: 0.7371
Epoch 23/100
100/100 [==============================] - 35s 355ms/step - loss: 0.5170 - acc: 0.7363 - val_loss: 0.4751 - val_acc: 0.7703
Epoch 24/100
100/100 [==============================] - 35s 347ms/step - loss: 0.5128 - acc: 0.7409 - val_loss: 0.5491 - val_acc: 0.7371
Epoch 25/100
100/100 [==============================] - 32s 316ms/step - loss: 0.5058 - acc: 0.7453 - val_loss: 0.5043 - val_acc: 0.7513
Epoch 26/100
100/100 [==============================] - 33s 332ms/step - loss: 0.4969 - acc: 0.7581 - val_loss: 0.4959 - val_acc: 0.7608
Epoch 27/100
100/100 [==============================] - 34s 339ms/step - loss: 0.4942 - acc: 0.7559 - val_loss: 0.4664 - val_acc: 0.7796
Epoch 28/100
100/100 [==============================] - 34s 344ms/step - loss: 0.4936 - acc: 0.7531 - val_loss: 0.4709 - val_acc: 0.7671
Epoch 29/100
100/100 [==============================] - 35s 346ms/step - loss: 0.4940 - acc: 0.7619 - val_loss: 0.5140 - val_acc: 0.7423
Epoch 30/100
100/100 [==============================] - 37s 367ms/step - loss: 0.4846 - acc: 0.7613 - val_loss: 0.4532 - val_acc: 0.7881
Epoch 31/100
100/100 [==============================] - 36s 361ms/step - loss: 0.4766 - acc: 0.7672 - val_loss: 0.5180 - val_acc: 0.7371
Epoch 32/100
100/100 [==============================] - 38s 380ms/step - loss: 0.4811 - acc: 0.7656 - val_loss: 0.4853 - val_acc: 0.7577
Epoch 33/100
100/100 [==============================] - 36s 361ms/step - loss: 0.4714 - acc: 0.7734 - val_loss: 0.4816 - val_acc: 0.7633
Epoch 34/100
100/100 [==============================] - 38s 378ms/step - loss: 0.4803 - acc: 0.7647 - val_loss: 0.4775 - val_acc: 0.7648
Epoch 35/100
100/100 [==============================] - 40s 402ms/step - loss: 0.4789 - acc: 0.7672 - val_loss: 0.4768 - val_acc: 0.7532
Epoch 36/100
100/100 [==============================] - 48s 479ms/step - loss: 0.4705 - acc: 0.7775 - val_loss: 0.5091 - val_acc: 0.7655
Epoch 37/100
100/100 [==============================] - 37s 372ms/step - loss: 0.4623 - acc: 0.7797 - val_loss: 0.5012 - val_acc: 0.7640
Epoch 38/100
100/100 [==============================] - 40s 397ms/step - loss: 0.4564 - acc: 0.7822 - val_loss: 0.5569 - val_acc: 0.7352
Epoch 39/100
100/100 [==============================] - 37s 373ms/step - loss: 0.4580 - acc: 0.7781 - val_loss: 0.4889 - val_acc: 0.7697
Epoch 40/100
100/100 [==============================] - 38s 381ms/step - loss: 0.4483 - acc: 0.7866 - val_loss: 0.5484 - val_acc: 0.7442
Epoch 41/100
100/100 [==============================] - 35s 350ms/step - loss: 0.4712 - acc: 0.7731 - val_loss: 0.4830 - val_acc: 0.7693
Epoch 42/100
100/100 [==============================] - 33s 334ms/step - loss: 0.4443 - acc: 0.7950 - val_loss: 0.5344 - val_acc: 0.7525
Epoch 43/100
100/100 [==============================] - 34s 343ms/step - loss: 0.4470 - acc: 0.7909 - val_loss: 0.4584 - val_acc: 0.8009
Epoch 44/100
100/100 [==============================] - 33s 330ms/step - loss: 0.4375 - acc: 0.7997 - val_loss: 0.4624 - val_acc: 0.7900
Epoch 45/100
100/100 [==============================] - 36s 356ms/step - loss: 0.4507 - acc: 0.7803 - val_loss: 0.5248 - val_acc: 0.7397
Epoch 46/100
100/100 [==============================] - 33s 333ms/step - loss: 0.4438 - acc: 0.7988 - val_loss: 0.4787 - val_acc: 0.7754
Epoch 47/100
100/100 [==============================] - 34s 345ms/step - loss: 0.4379 - acc: 0.7922 - val_loss: 0.4491 - val_acc: 0.7938
Epoch 48/100
100/100 [==============================] - 33s 335ms/step - loss: 0.4470 - acc: 0.7875 - val_loss: 0.4307 - val_acc: 0.8080
Epoch 49/100
100/100 [==============================] - 33s 328ms/step - loss: 0.4226 - acc: 0.8041 - val_loss: 0.4550 - val_acc: 0.7976
Epoch 50/100
100/100 [==============================] - 34s 343ms/step - loss: 0.4365 - acc: 0.7938 - val_loss: 0.4428 - val_acc: 0.7996
Epoch 51/100
100/100 [==============================] - 33s 328ms/step - loss: 0.4251 - acc: 0.8000 - val_loss: 0.5243 - val_acc: 0.7405
Epoch 52/100
100/100 [==============================] - 36s 364ms/step - loss: 0.4340 - acc: 0.7956 - val_loss: 0.5288 - val_acc: 0.7468
Epoch 53/100
100/100 [==============================] - 33s 333ms/step - loss: 0.4217 - acc: 0.8053 - val_loss: 0.4535 - val_acc: 0.7868
Epoch 54/100
100/100 [==============================] - 34s 344ms/step - loss: 0.4191 - acc: 0.8000 - val_loss: 0.4612 - val_acc: 0.7977
Epoch 55/100
100/100 [==============================] - 33s 332ms/step - loss: 0.4272 - acc: 0.8019 - val_loss: 0.4419 - val_acc: 0.8020
Epoch 56/100
100/100 [==============================] - 33s 330ms/step - loss: 0.4247 - acc: 0.7975 - val_loss: 0.4434 - val_acc: 0.7925
Epoch 57/100
100/100 [==============================] - 35s 354ms/step - loss: 0.4049 - acc: 0.8069 - val_loss: 0.4338 - val_acc: 0.8099
```

```
Epoch 58/100
100/100 [==============================] - 34s 335ms/step - loss: 0.4221 - acc: 0.8088 - val_loss: 0.4195 - val_acc: 0.8109
Epoch 59/100
100/100 [==============================] - 33s 333ms/step - loss: 0.4133 - acc: 0.8100 - val_loss: 0.4690 - val_acc: 0.7996
Epoch 60/100

100/100 [==============================] - 33s 331ms/step - loss: 0.4093 - acc: 0.8053 - val_loss: 0.4477 - val_acc: 0.7893
Epoch 61/100
100/100 [==============================] - 34s 337ms/step - loss: 0.3897 - acc: 0.8228 - val_loss: 0.4386 - val_acc: 0.7964
Epoch 62/100
100/100 [==============================] - 35s 351ms/step - loss: 0.4043 - acc: 0.8091 - val_loss: 0.4471 - val_acc: 0.7919
Epoch 63/100
100/100 [==============================] - 34s 338ms/step - loss: 0.3938 - acc: 0.8203 - val_loss: 0.4373 - val_acc: 0.7925
Epoch 64/100
100/100 [==============================] - 33s 327ms/step - loss: 0.3979 - acc: 0.8134 - val_loss: 0.4371 - val_acc: 0.8164
Epoch 65/100
100/100 [==============================] - 33s 333ms/step - loss: 0.3985 - acc: 0.8197 - val_loss: 0.4513 - val_acc: 0.8084
Epoch 66/100
100/100 [==============================] - 33s 332ms/step - loss: 0.3928 - acc: 0.8156 - val_loss: 0.4091 - val_acc: 0.8299
Epoch 67/100
100/100 [==============================] - 33s 326ms/step - loss: 0.3957 - acc: 0.8281 - val_loss: 0.4548 - val_acc: 0.8027
Epoch 68/100
100/100 [==============================] - 33s 333ms/step - loss: 0.4025 - acc: 0.8175 - val_loss: 0.4434 - val_acc: 0.7899
Epoch 69/100
100/100 [==============================] - 35s 351ms/step - loss: 0.3704 - acc: 0.8297 - val_loss: 0.4671 - val_acc: 0.7982
Epoch 70/100
100/100 [==============================] - 34s 336ms/step - loss: 0.3822 - acc: 0.8284 - val_loss: 0.4233 - val_acc: 0.8189
Epoch 71/100
100/100 [==============================] - 34s 337ms/step - loss: 0.3805 - acc: 0.8238 - val_loss: 0.4013 - val_acc: 0.8236
Epoch 72/100
100/100 [==============================] - 34s 338ms/step - loss: 0.3716 - acc: 0.8306 - val_loss: 0.4476 - val_acc: 0.7803
Epoch 73/100
100/100 [==============================] - 33s 330ms/step - loss: 0.3733 - acc: 0.8325 - val_loss: 0.4086 - val_acc: 0.8254
Epoch 74/100
100/100 [==============================] - 36s 358ms/step - loss: 0.3787 - acc: 0.8297 - val_loss: 0.4302 - val_acc: 0.8039
Epoch 75/100
100/100 [==============================] - 33s 334ms/step - loss: 0.3798 - acc: 0.8269 - val_loss: 0.4454 - val_acc: 0.7945
Epoch 76/100
100/100 [==============================] - 33s 332ms/step - loss: 0.3864 - acc: 0.8287 - val_loss: 0.4593 - val_acc: 0.7887
Epoch 77/100
100/100 [==============================] - 33s 332ms/step - loss: 0.3601 - acc: 0.8341 - val_loss: 0.4157 - val_acc: 0.8138
Epoch 78/100
100/100 [==============================] - 33s 326ms/step - loss: 0.3628 - acc: 0.8425 - val_loss: 0.4602 - val_acc: 0.7912
Epoch 79/100
100/100 [==============================] - 34s 341ms/step - loss: 0.3657 - acc: 0.8369 - val_loss: 0.4358 - val_acc: 0.8170
Epoch 80/100
100/100 [==============================] - 32s 322ms/step - loss: 0.3741 - acc: 0.8263 - val_loss: 0.4258 - val_acc: 0.8093
Epoch 81/100
100/100 [==============================] - 35s 349ms/step - loss: 0.3759 - acc: 0.8331 - val_loss: 0.3883 - val_acc: 0.8204
Epoch 82/100
100/100 [==============================] - 33s 327ms/step - loss: 0.3568 - acc: 0.8384 - val_loss: 0.5135 - val_acc: 0.7880
Epoch 83/100
100/100 [==============================] - 33s 334ms/step - loss: 0.3663 - acc: 0.8391 - val_loss: 0.3840 - val_acc: 0.8312
Epoch 84/100
100/100 [==============================] - 33s 329ms/step - loss: 0.3573 - acc: 0.8406 - val_loss: 0.4564 - val_acc: 0.7964
Epoch 85/100
100/100 [==============================] - 32s 323ms/step - loss: 0.3654 - acc: 0.8366 - val_loss: 0.4225 - val_acc: 0.8122
Epoch 86/100
100/100 [==============================] - 35s 355ms/step - loss: 0.3463 - acc: 0.8487 - val_loss: 0.4515 - val_acc: 0.7925
Epoch 87/100
100/100 [==============================] - 33s 330ms/step - loss: 0.3577 - acc: 0.8447 - val_loss: 0.4462 - val_acc: 0.8071
Epoch 88/100
100/100 [==============================] - 34s 343ms/step - loss: 0.3515 - acc: 0.8437 - val_loss: 0.4197 - val_acc: 0.8189
Epoch 89/100
100/100 [==============================] - 33s 333ms/step - loss: 0.3402 - acc: 0.8478 - val_loss: 0.4406 - val_acc: 0.8164
Epoch 90/100
100/100 [==============================] - 33s 332ms/step - loss: 0.3470 - acc: 0.8409 - val_loss: 0.4135 - val_acc: 0.8357
Epoch 91/100
100/100 [==============================] - 34s 337ms/step - loss: 0.3289 - acc: 0.8584 - val_loss: 0.4187 - val_acc: 0.8376
Epoch 92/100
100/100 [==============================] - 33s 330ms/step - loss: 0.3566 - acc: 0.8416 - val_loss: 0.4417 - val_acc: 0.8071
Epoch 93/100
100/100 [==============================] - 36s 360ms/step - loss: 0.3401 - acc: 0.8513 - val_loss: 0.4712 - val_acc: 0.8247
Epoch 94/100
100/100 [==============================] - 35s 350ms/step - loss: 0.3456 - acc: 0.8456 - val_loss: 0.4433 - val_acc: 0.8128
Epoch 95/100
100/100 [==============================] - 40s 396ms/step - loss: 0.3487 - acc: 0.8419 - val_loss: 0.4181 - val_acc: 0.8331
Epoch 96/100
100/100 [==============================] - 36s 364ms/step - loss: 0.3514 - acc: 0.8444 - val_loss: 0.4014 - val_acc: 0.8235
Epoch 97/100
100/100 [==============================] - 36s 365ms/step - loss: 0.3166 - acc: 0.8559 - val_loss: 0.4481 - val_acc: 0.8293
Epoch 98/100
100/100 [==============================] - 39s 392ms/step - loss: 0.3346 - acc: 0.8509 - val_loss: 0.4649 - val_acc: 0.7919
Epoch 99/100
100/100 [==============================] - 35s 354ms/step - loss: 0.3358 - acc: 0.8494 - val_loss: 0.4357 - val_acc: 0.8027
Epoch 100/100
100/100 [==============================] - 36s 362ms/step - loss: 0.3235 - acc: 0.8547 - val_loss: 0.4401 - val_acc: 0.8138
```

In [21]:
```
model.save('cats_and_dogs_small_2.h5')
```

In [22]:
```
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()
```