An Introduction to Deep Learning With Python

[5.2] Training a convnet from scratch on a small dataset

Prof. Yuzo Iano

pgs: 130 - 142

Dowloading the data

You can dowload the original dataset from www.kaggle.com/c/dogs-vs-cats/data (you'll need to create a Kaggle account if you don´t already have one)

Copying images to training, validation and test directories

Creating folders

In [1]:

```
import os, shutil

original_dataset_dir = '../CAP_5/kaggle_original_data/'
#print(os.listdir('../CAP_5/kaggle_original_data/'))

base_dir = '../CAP_5/cats_and_dogs_small'
os.mkdir(base_dir)

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)
```

Copying cat's images in each one folder

In [2]:

```
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)

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)
```

Copying dog's images in each one folder

In [3]:

```
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)
```

Count how many pictures are in each folder

In [4]:

```
print('Total training cat images: ', len(os.listdir(train_cats_dir)))
print('Total training dog images: ', len(os.listdir(train_dogs_dir)))
print('Total validation cat images: ', len(os.listdir(validation_cats_dir)))
print('Total validation dog images: ', len(os.listdir(validation_dogs_dir)))
print('Total test cat images: ', len(os.listdir(test_cats_dir)))
print('Total test dog images: ', len(os.listdir(test_dogs_dir)))
```

```
Total training cat images:  1000
Total training dog images:  1000
Total validation cat images:  500
Total validation dog images:  500
Total test cat images:  500
Total test dog images:  500
```

Building your network

In [5]:

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten

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

```
Using TensorFlow backend.

WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\py
thon\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is depr
ecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
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
_________________________________________________________________
```

Configuring the model for training

In [6]:

```
from keras.optimizers import RMSprop

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

Data preprocessing

Using ImageDataGenerator to read images from directories

In [7]:

```
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 [8]:

```
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,)
```

Fitting the model using a batch generator

In [9]:

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

```
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\py
thon\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will b
e removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/30
100/100 [==============================] - 75s 752ms/step - loss: 0.6898 - acc: 0.5280 - val_loss:
0.6710 - val_acc: 0.5730
Epoch 2/30
100/100 [==============================] - 78s 777ms/step - loss: 0.6661 - acc: 0.5880 - val_loss:
0.6655 - val_acc: 0.5750
Epoch 3/30
100/100 [==============================] - 66s 662ms/step - loss: 0.6297 - acc: 0.6405 - val_loss:
0.6204 - val_acc: 0.6570
Epoch 4/30
100/100 [==============================] - 67s 675ms/step - loss: 0.5901 - acc: 0.6850 - val_loss:
0.5932 - val_acc: 0.6850
Epoch 5/30
100/100 [==============================] - 82s 824ms/step - loss: 0.5546 - acc: 0.7055 - val_loss:
0.5887 - val_acc: 0.6900
Epoch 6/30
100/100 [==============================] - 86s 858ms/step - loss: 0.5235 - acc: 0.7405 - val_loss:
0.5654 - val_acc: 0.7110
Epoch 7/30
100/100 [==============================] - 93s 925ms/step - loss: 0.5015 - acc: 0.7535 - val_loss:
0.6365 - val_acc: 0.6440
Epoch 8/30
100/100 [==============================] - 69s 693ms/step - loss: 0.4665 - acc: 0.7650 - val_loss:
0.5328 - val_acc: 0.7350
Epoch 9/30
100/100 [==============================] - 74s 743ms/step - loss: 0.4337 - acc: 0.8010 - val_loss:
0.5766 - val_acc: 0.7140
Epoch 10/30
100/100 [==============================] - 91s 907ms/step - loss: 0.4186 - acc: 0.8035 - val_loss:
0.6908 - val_acc: 0.6570
Epoch 11/30
100/100 [==============================] - 86s 857ms/step - loss: 0.3888 - acc: 0.8250 - val_loss:
0.5294 - val_acc: 0.7460
Epoch 12/30
100/100 [==============================] - 83s 832ms/step - loss: 0.3690 - acc: 0.8425 - val_loss:
0.5259 - val_acc: 0.7500
Epoch 13/30
100/100 [==============================] - 73s 727ms/step - loss: 0.3481 - acc: 0.8470 - val_loss:
0.6515 - val_acc: 0.7010
Epoch 14/30
100/100 [==============================] - 75s 754ms/step - loss: 0.3324 - acc: 0.8500 - val_loss:
0.5537 - val_acc: 0.7400
Epoch 15/30
100/100 [==============================] - 69s 695ms/step - loss: 0.3121 - acc: 0.8645 - val_loss:
0.5582 - val_acc: 0.7390
Epoch 16/30
100/100 [==============================] - 83s 826ms/step - loss: 0.2842 - acc: 0.8830 - val_loss:
0.6626 - val_acc: 0.7090
Epoch 17/30
100/100 [==============================] - 88s 879ms/step - loss: 0.2688 - acc: 0.8850 - val_loss:
0.6211 - val_acc: 0.7420
Epoch 18/30
100/100 [==============================] - 89s 888ms/step - loss: 0.2570 - acc: 0.9020 - val_loss:
0.6099 - val_acc: 0.7300
Epoch 19/30
100/100 [==============================] - 70s 702ms/step - loss: 0.2197 - acc: 0.9095 - val_loss:
0.5923 - val_acc: 0.7530
Epoch 20/30
100/100 [==============================] - 71s 715ms/step - loss: 0.2125 - acc: 0.9180 - val_loss:
0.6393 - val_acc: 0.7480
Epoch 21/30
100/100 [==============================] - 79s 795ms/step - loss: 0.2011 - acc: 0.9180 - val_loss:
0.5836 - val_acc: 0.7610
Epoch 22/30
100/100 [==============================] - 80s 797ms/step - loss: 0.1779 - acc: 0.9295 - val_loss:
0.6387 - val_acc: 0.7520
Epoch 23/30
100/100 [==============================] - 79s 795ms/step - loss: 0.1584 - acc: 0.9475 - val_loss:
0.6343 - val_acc: 0.7490
Epoch 24/30
100/100 [==============================] - 79s 787ms/step - loss: 0.1411 - acc: 0.9565 - val_loss:
0.6616 - val_acc: 0.7470
Epoch 25/30
100/100 [==============================] - 81s 811ms/step - loss: 0.1214 - acc: 0.9585 - val_loss:
```

```
0.7097 - val_acc: 0.7420
Epoch 26/30
100/100 [==============================] - 79s 786ms/step - loss: 0.1136 - acc: 0.9555 - val_loss:
0.7103 - val_acc: 0.7440
Epoch 27/30
100/100 [==============================] - 82s 815ms/step - loss: 0.0980 - acc: 0.9715 - val_loss:
0.8269 - val_acc: 0.7280
Epoch 28/30
100/100 [==============================] - 83s 834ms/step - loss: 0.0877 - acc: 0.9740 - val_loss:
0.9028 - val_acc: 0.7250
Epoch 29/30
100/100 [==============================] - 80s 801ms/step - loss: 0.0777 - acc: 0.9770 - val_loss:
0.7807 - val_acc: 0.7440
Epoch 30/30
100/100 [==============================] - 81s 815ms/step - loss: 0.0654 - acc: 0.9825 - val_loss:
0.8211 - val_acc: 0.7590
```

Saving the model

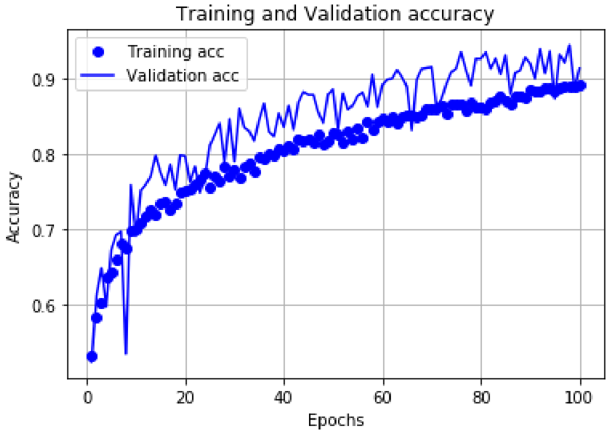
In [10]:

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

Displaying curves of loss and accuracy during training

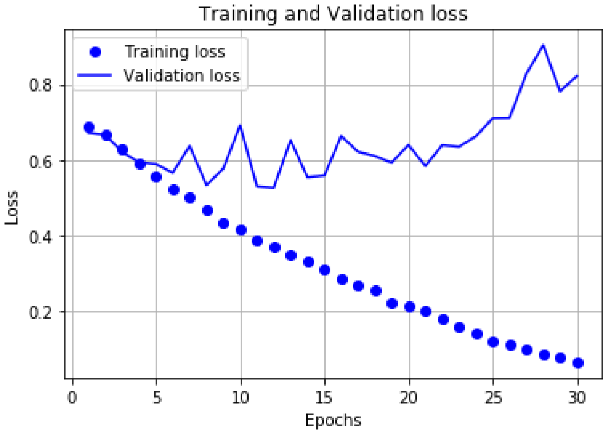
In [22]:

```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.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show()
```



In [12]:

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



Setting up a data augmentation configuration via ImageDataGenerator

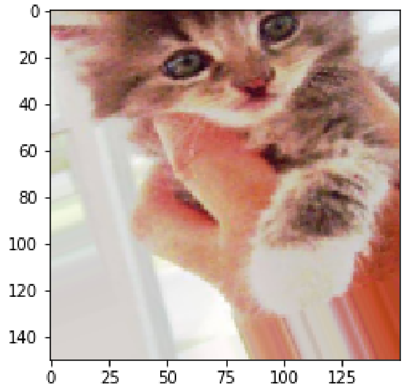
In [13]:

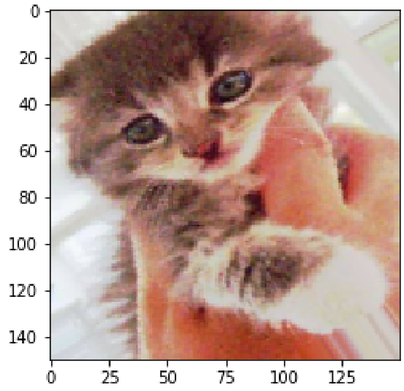
```
datagen = ImageDataGenerator(rotation_range=40,
                             width_shift_range=0.2,
                             height_shift_range=0.2,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True,
 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

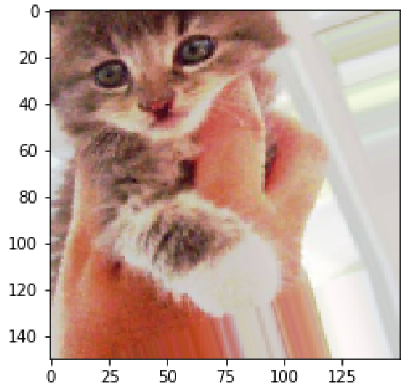
Displaying some randomly augmented training images

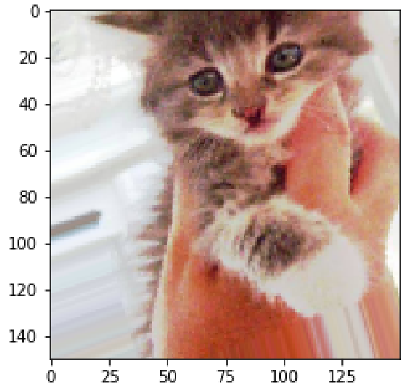
In [14]:

```
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()
```









Defining a new convnet that includes dropout

In [15]:
```
from keras.layers import Dropout

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

```
WARNING:tensorflow:From C:\Users\pablo\Python\envs\DAVID\lib\site-packages\keras\backend\tensorflo
w_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecate
d and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_5 (Conv2D)            (None, 148, 148, 32)      896       
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 74, 74, 32)        0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 72, 72, 64)        18496     
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 36, 36, 64)        0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 34, 34, 128)       73856     
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 17, 17, 128)       0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 15, 15, 128)       147584    
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (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 [16]:
```
model.compile(loss = 'binary_crossentropy',
              optimizer=RMSprop(lr=1e-4),
              metrics = ['acc'])
```

Training the convnet usign data-augmentation generators

In [17]:

```
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)
train_generator = train_datagen.flow_from_directory(validation_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 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

In [18]:

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

```
Epoch 1/100
100/100 [==============================] - 128s 1s/step - loss: 0.6880 - acc: 0.5368 - val_loss:
0.6810 - val_acc: 0.5241
Epoch 2/100
100/100 [==============================] - 110s 1s/step - loss: 0.6672 - acc: 0.5859 - val_loss:
0.6444 - val_acc: 0.6121
Epoch 3/100
100/100 [==============================] - 100s 996ms/step - loss: 0.6549 - acc: 0.6053 - val_los
s: 0.6227 - val_acc: 0.6485
Epoch 4/100
100/100 [==============================] - 102s 1s/step - loss: 0.6330 - acc: 0.6381 - val_loss:
0.6870 - val_acc: 0.5973
Epoch 5/100
100/100 [==============================] - 106s 1s/step - loss: 0.6221 - acc: 0.6413 - val_loss:
0.5897 - val_acc: 0.6732
Epoch 6/100
100/100 [==============================] - 111s 1s/step - loss: 0.6039 - acc: 0.6619 - val_loss:
0.5578 - val_acc: 0.6927
Epoch 7/100
100/100 [==============================] - 115s 1s/step - loss: 0.5860 - acc: 0.6772 - val_loss:
0.5460 - val_acc: 0.6973
Epoch 8/100
100/100 [==============================] - 115s 1s/step - loss: 0.5895 - acc: 0.6753 - val_loss:
0.8154 - val_acc: 0.5348
Epoch 9/100
100/100 [==============================] - 115s 1s/step - loss: 0.5722 - acc: 0.6969 - val_loss:
0.5078 - val_acc: 0.7590
Epoch 10/100
100/100 [==============================] - 119s 1s/step - loss: 0.5585 - acc: 0.7043 - val_loss:
0.5641 - val_acc: 0.6910
Epoch 11/100
100/100 [==============================] - 121s 1s/step - loss: 0.5532 - acc: 0.7106 - val_loss:
0.5038 - val_acc: 0.7519
Epoch 12/100
100/100 [==============================] - 121s 1s/step - loss: 0.5418 - acc: 0.7172 - val_loss:
0.4827 - val_acc: 0.7595
Epoch 13/100
100/100 [==============================] - 118s 1s/step - loss: 0.5384 - acc: 0.7259 - val_loss:
0.4696 - val_acc: 0.7706
Epoch 14/100
100/100 [==============================] - 119s 1s/step - loss: 0.5359 - acc: 0.7231 - val_loss:
0.4432 - val_acc: 0.7982
Epoch 15/100
100/100 [==============================] - 118s 1s/step - loss: 0.5255 - acc: 0.7331 - val_loss:
0.4660 - val_acc: 0.7758
Epoch 16/100
100/100 [==============================] - 121s 1s/step - loss: 0.5189 - acc: 0.7344 - val_loss:
0.4807 - val_acc: 0.7590
Epoch 17/100
100/100 [==============================] - 117s 1s/step - loss: 0.5153 - acc: 0.7262 - val_loss:
0.4407 - val_acc: 0.7862
Epoch 18/100
100/100 [==============================] - 110s 1s/step - loss: 0.5231 - acc: 0.7322 - val_loss:
0.4939 - val_acc: 0.7526
Epoch 19/100
100/100 [==============================] - 100s 997ms/step - loss: 0.5019 - acc: 0.7494 - val_los
s: 0.4271 - val_acc: 0.7989
Epoch 20/100
100/100 [==============================] - 96s 955ms/step - loss: 0.4953 - acc: 0.7509 - val_loss:
0.4498 - val_acc: 0.7977
Epoch 21/100
100/100 [==============================] - 99s 995ms/step - loss: 0.4878 - acc: 0.7587 - val_loss:
0.4953 - val_acc: 0.7627
Epoch 22/100
100/100 [==============================] - 96s 958ms/step - loss: 0.4904 - acc: 0.7612 - val_loss:
0.4410 - val_acc: 0.7835
Epoch 23/100
100/100 [==============================] - 99s 994ms/step - loss: 0.4867 - acc: 0.7659 - val_loss:
0.5121 - val_acc: 0.7487
Epoch 24/100
100/100 [==============================] - 96s 960ms/step - loss: 0.4683 - acc: 0.7765 - val_loss:
0.4834 - val_acc: 0.7700
Epoch 25/100
100/100 [==============================] - 98s 983ms/step - loss: 0.4938 - acc: 0.7556 - val_loss:
0.4235 - val_acc: 0.8112
Epoch 26/100
100/100 [==============================] - 98s 976ms/step - loss: 0.4737 - acc: 0.7685 - val_loss:
0.4016 - val_acc: 0.8255
Epoch 27/100
```

```
100/100 [==============================] - 98s 975ms/step - loss: 0.4808 - acc: 0.7644 - val_loss:
0.3854 - val_acc: 0.8409
Epoch 28/100
100/100 [==============================] - 98s 983ms/step - loss: 0.4586 - acc: 0.7865 - val_loss:
0.4484 - val_acc: 0.7893
Epoch 29/100
100/100 [==============================] - 97s 966ms/step - loss: 0.4538 - acc: 0.7712 - val_loss:
0.3652 - val_acc: 0.8466
Epoch 30/100
100/100 [==============================] - 99s 992ms/step - loss: 0.4561 - acc: 0.7791 - val_loss:
0.4553 - val_acc: 0.7906
Epoch 31/100
100/100 [==============================] - 96s 961ms/step - loss: 0.4676 - acc: 0.7653 - val_loss:
0.3449 - val_acc: 0.8608
Epoch 32/100
100/100 [==============================] - 99s 993ms/step - loss: 0.4501 - acc: 0.7838 - val_loss:
0.3846 - val_acc: 0.8351
Epoch 33/100
100/100 [==============================] - 96s 957ms/step - loss: 0.4473 - acc: 0.7878 - val_loss:
0.3805 - val_acc: 0.8306
Epoch 34/100
100/100 [==============================] - 100s 997ms/step - loss: 0.4442 - acc: 0.7790 - val_los
s: 0.4109 - val_acc: 0.8177
Epoch 35/100
100/100 [==============================] - 97s 965ms/step - loss: 0.4366 - acc: 0.7953 - val_loss:
0.3343 - val_acc: 0.8452
Epoch 36/100
100/100 [==============================] - 99s 987ms/step - loss: 0.4342 - acc: 0.7944 - val_loss:
0.3266 - val_acc: 0.8673
Epoch 37/100
100/100 [==============================] - 97s 973ms/step - loss: 0.4315 - acc: 0.7978 - val_loss:
0.3627 - val_acc: 0.8299
Epoch 38/100
100/100 [==============================] - 98s 978ms/step - loss: 0.4322 - acc: 0.7953 - val_loss:
0.4055 - val_acc: 0.8235
Epoch 39/100
100/100 [==============================] - 98s 982ms/step - loss: 0.4064 - acc: 0.8075 - val_loss:
0.3268 - val_acc: 0.8541
Epoch 40/100
100/100 [==============================] - 96s 964ms/step - loss: 0.4186 - acc: 0.8044 - val_loss:
0.3759 - val_acc: 0.8357
Epoch 41/100
100/100 [==============================] - 99s 989ms/step - loss: 0.4072 - acc: 0.8085 - val_loss:
0.3313 - val_acc: 0.8640
Epoch 42/100
100/100 [==============================] - 97s 966ms/step - loss: 0.4158 - acc: 0.8044 - val_loss:
0.3522 - val_acc: 0.8325
Epoch 43/100
100/100 [==============================] - 99s 994ms/step - loss: 0.3978 - acc: 0.8181 - val_loss:
0.3122 - val_acc: 0.8692
Epoch 44/100
100/100 [==============================] - 96s 957ms/step - loss: 0.3897 - acc: 0.8178 - val_loss:
0.3069 - val_acc: 0.8826
Epoch 45/100
100/100 [==============================] - 100s 998ms/step - loss: 0.3973 - acc: 0.8197 - val_los
s: 0.2868 - val_acc: 0.8789
Epoch 46/100
100/100 [==============================] - 97s 973ms/step - loss: 0.3959 - acc: 0.8128 - val_loss:
0.2796 - val_acc: 0.8788
Epoch 47/100
100/100 [==============================] - 105s 1s/step - loss: 0.3804 - acc: 0.8253 - val_loss:
0.3212 - val_acc: 0.8557
Epoch 48/100
100/100 [==============================] - 104s 1s/step - loss: 0.4022 - acc: 0.8122 - val_loss:
0.3889 - val_acc: 0.8415
Epoch 49/100
100/100 [==============================] - 113s 1s/step - loss: 0.3982 - acc: 0.8178 - val_loss:
0.2784 - val_acc: 0.8794
Epoch 50/100
100/100 [==============================] - 126s 1s/step - loss: 0.3782 - acc: 0.8266 - val_loss:
0.2700 - val_acc: 0.8860
Epoch 51/100
100/100 [==============================] - 130s 1s/step - loss: 0.3783 - acc: 0.8235 - val_loss:
0.3395 - val_acc: 0.8319
Epoch 52/100
100/100 [==============================] - 133s 1s/step - loss: 0.3891 - acc: 0.8184 - val_loss:
0.2925 - val_acc: 0.8808
Epoch 53/100
100/100 [==============================] - 132s 1s/step - loss: 0.3684 - acc: 0.8306 - val_loss:
0.3296 - val_acc: 0.8591
```

```
Epoch 54/100
100/100 [==============================] - 131s 1s/step - loss: 0.3728 - acc: 0.8209 - val_loss:
0.3081 - val_acc: 0.8647
Epoch 55/100
100/100 [==============================] - 126s 1s/step - loss: 0.3692 - acc: 0.8347 - val_loss:
0.2837 - val_acc: 0.8769
Epoch 56/100
100/100 [==============================] - 100s 996ms/step - loss: 0.3670 - acc: 0.8216 - val_los
s: 0.2828 - val_acc: 0.8821
Epoch 57/100
100/100 [==============================] - 100s 1s/step - loss: 0.3462 - acc: 0.8440 - val_loss:
0.3192 - val_acc: 0.8634
Epoch 58/100
100/100 [==============================] - 100s 997ms/step - loss: 0.3701 - acc: 0.8334 - val_los
s: 0.2427 - val_acc: 0.9055
Epoch 59/100
100/100 [==============================] - 102s 1s/step - loss: 0.3503 - acc: 0.8407 - val_loss:
0.3063 - val_acc: 0.8628
Epoch 60/100
100/100 [==============================] - 105s 1s/step - loss: 0.3538 - acc: 0.8478 - val_loss:
0.2686 - val_acc: 0.8915
Epoch 61/100
100/100 [==============================] - 102s 1s/step - loss: 0.3537 - acc: 0.8419 - val_loss:
0.2355 - val_acc: 0.8988
Epoch 62/100
100/100 [==============================] - 98s 984ms/step - loss: 0.3487 - acc: 0.8503 - val_loss:
0.2487 - val_acc: 0.9004
Epoch 63/100
100/100 [==============================] - 100s 1s/step - loss: 0.3520 - acc: 0.8391 - val_loss:
0.2222 - val_acc: 0.9117
Epoch 64/100
100/100 [==============================] - 99s 991ms/step - loss: 0.3488 - acc: 0.8484 - val_loss:
0.2449 - val_acc: 0.9021
Epoch 65/100
100/100 [==============================] - 101s 1s/step - loss: 0.3420 - acc: 0.8506 - val_loss:
0.2512 - val_acc: 0.8896
Epoch 66/100
100/100 [==============================] - 100s 1s/step - loss: 0.3352 - acc: 0.8503 - val_loss:
0.3864 - val_acc: 0.8318
Epoch 67/100
100/100 [==============================] - 106s 1s/step - loss: 0.3343 - acc: 0.8497 - val_loss:
0.2474 - val_acc: 0.8991
Epoch 68/100
100/100 [==============================] - 102s 1s/step - loss: 0.3347 - acc: 0.8544 - val_loss:
0.2073 - val_acc: 0.9130
Epoch 69/100
100/100 [==============================] - 107s 1s/step - loss: 0.3232 - acc: 0.8597 - val_loss:
0.2202 - val_acc: 0.9143
Epoch 70/100
100/100 [==============================] - 118s 1s/step - loss: 0.3162 - acc: 0.8594 - val_loss:
0.2123 - val_acc: 0.9156
Epoch 71/100
100/100 [==============================] - 128s 1s/step - loss: 0.3139 - acc: 0.8566 - val_loss:
0.2996 - val_acc: 0.8629
Epoch 72/100
100/100 [==============================] - 116s 1s/step - loss: 0.3153 - acc: 0.8665 - val_loss:
0.2728 - val_acc: 0.8731
Epoch 73/100
100/100 [==============================] - 119s 1s/step - loss: 0.3268 - acc: 0.8525 - val_loss:
0.2593 - val_acc: 0.8930
Epoch 74/100
100/100 [==============================] - 106s 1s/step - loss: 0.3112 - acc: 0.8662 - val_loss:
0.2142 - val_acc: 0.9086
Epoch 75/100
100/100 [==============================] - 99s 988ms/step - loss: 0.3070 - acc: 0.8666 - val_loss:
0.2189 - val_acc: 0.9137
Epoch 76/100
100/100 [==============================] - 101s 1s/step - loss: 0.3015 - acc: 0.8678 - val_loss:
0.1775 - val_acc: 0.9353
Epoch 77/100
100/100 [==============================] - 98s 977ms/step - loss: 0.3113 - acc: 0.8606 - val_loss:
0.2166 - val_acc: 0.9117
Epoch 78/100
100/100 [==============================] - 105s 1s/step - loss: 0.3091 - acc: 0.8687 - val_loss:
0.2623 - val_acc: 0.8909
Epoch 79/100
100/100 [==============================] - 100s 1000ms/step - loss: 0.3121 - acc: 0.8619 - val_los
s: 0.1917 - val_acc: 0.9265
Epoch 80/100
100/100 [==============================] - 100s 998ms/step - loss: 0.3042 - acc: 0.8637 - val_los
```

```
s: 0.2066 - val_acc: 0.9265
Epoch 81/100
100/100 [==============================] - 100s 998ms/step - loss: 0.3088 - acc: 0.8606 - val_los
s: 0.1704 - val_acc: 0.9353
Epoch 82/100
100/100 [==============================] - 99s 992ms/step - loss: 0.2978 - acc: 0.8672 - val_loss:
0.2023 - val_acc: 0.9137
Epoch 83/100
100/100 [==============================] - 100s 1s/step - loss: 0.2881 - acc: 0.8728 - val_loss:
0.1795 - val_acc: 0.9270
Epoch 84/100
100/100 [==============================] - 103s 1s/step - loss: 0.2844 - acc: 0.8778 - val_loss:
0.2208 - val_acc: 0.9059
Epoch 85/100
100/100 [==============================] - 100s 1s/step - loss: 0.2983 - acc: 0.8694 - val_loss:
0.1799 - val_acc: 0.9315
Epoch 86/100
100/100 [==============================] - 97s 968ms/step - loss: 0.2974 - acc: 0.8669 - val_loss:
0.2796 - val_acc: 0.8795
Epoch 87/100
100/100 [==============================] - 101s 1s/step - loss: 0.2810 - acc: 0.8781 - val_loss:
0.2052 - val_acc: 0.9080
Epoch 88/100
100/100 [==============================] - 97s 971ms/step - loss: 0.2807 - acc: 0.8760 - val_loss:
0.2312 - val_acc: 0.9111
Epoch 89/100
100/100 [==============================] - 100s 999ms/step - loss: 0.2838 - acc: 0.8784 - val_los
s: 0.1730 - val_acc: 0.9285
Epoch 90/100
100/100 [==============================] - 99s 990ms/step - loss: 0.2690 - acc: 0.8859 - val_loss:
0.1854 - val_acc: 0.9207
Epoch 91/100
100/100 [==============================] - 99s 990ms/step - loss: 0.2677 - acc: 0.8853 - val_loss:
0.2404 - val_acc: 0.9001
Epoch 92/100
100/100 [==============================] - 106s 1s/step - loss: 0.2692 - acc: 0.8844 - val_loss:
0.1533 - val_acc: 0.9397
Epoch 93/100
100/100 [==============================] - 101s 1s/step - loss: 0.2717 - acc: 0.8875 - val_loss:
0.2180 - val_acc: 0.9072
Epoch 94/100
100/100 [==============================] - 122s 1s/step - loss: 0.2634 - acc: 0.8863 - val_loss:
0.1695 - val_acc: 0.9365
Epoch 95/100
100/100 [==============================] - 131s 1s/step - loss: 0.2715 - acc: 0.8837 - val_loss:
0.2946 - val_acc: 0.8737
Epoch 96/100
100/100 [==============================] - 123s 1s/step - loss: 0.2596 - acc: 0.8878 - val_loss:
0.1686 - val_acc: 0.9323
Epoch 97/100
100/100 [==============================] - 122s 1s/step - loss: 0.2626 - acc: 0.8881 - val_loss:
0.1958 - val_acc: 0.9213
Epoch 98/100
100/100 [==============================] - 109s 1s/step - loss: 0.2521 - acc: 0.8884 - val_loss:
0.1676 - val_acc: 0.9446
Epoch 99/100
100/100 [==============================] - 104s 1s/step - loss: 0.2476 - acc: 0.8894 - val_loss:
0.2974 - val_acc: 0.8864
Epoch 100/100
100/100 [==============================] - 103s 1s/step - loss: 0.2654 - acc: 0.8879 - val_loss:
0.1961 - val_acc: 0.9137
```

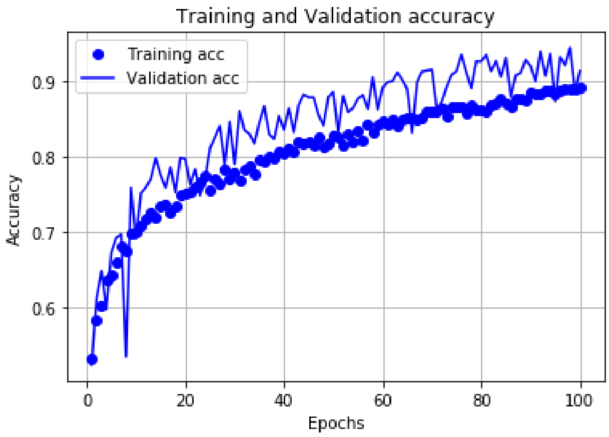
In [19]:

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

Displaying curves of loss and accuracy during training

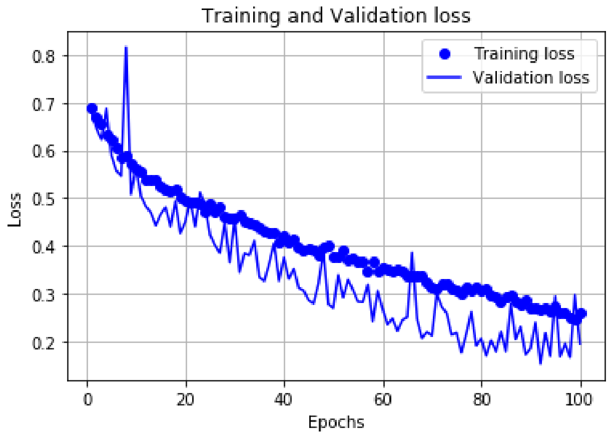
In [20]:

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

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



In [21]:

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



Pablo Minango

- pablodavid218@gmail.com