```
import keras
keras.__version__
```

```
'2.4.3'
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os, shutil
```

```
base_dir = '/content/drive/MyDrive/cats_and_dogs_small'
!ls '/content/drive/MyDrive/cats_and_dogs_small'
```

```
test  train  validation
```

```
# Directories for our training,
# validation and test splits
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

# Directory with our training cat pictures
train_cats_dir = os.path.join(train_dir, 'cats')

# Directory with our training dog pictures
train_dogs_dir = os.path.join(train_dir, 'dogs')

# Directory with our validation cat pictures
validation_cats_dir = os.path.join(validation_dir, 'cats')

# Directory with our validation dog pictures
validation_dogs_dir = os.path.join(validation_dir, 'dogs')

# Directory with our validation cat pictures
test_cats_dir = os.path.join(test_dir, 'cats')

# Directory with our validation dog pictures
test_dogs_dir = os.path.join(test_dir, 'dogs')
```

```
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: 1068
total training dog images: 1010
total validation cat images: 500
total validation dog images: 500
total test cat images: 589
total test dog images: 580
```

```
from keras import layers
from keras import models

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()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 17, 17, 128)	0
conv2d_3 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 512)	3211776
dense_1 (Dense)	(None, 1)	513

```
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

```
from keras import optimizers

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

```
from keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 150x150
        target_size=(150, 150),
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

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

```
Found 2078 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

```
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,)
```

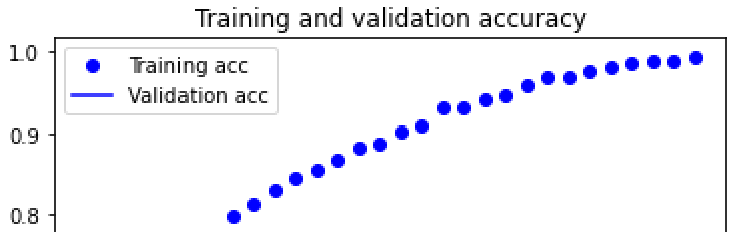
```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.p
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/30
100/100 [==============================] - 868s 9s/step - loss: 0.6968 - acc: 0.
Epoch 2/30
100/100 [==============================] - 99s 990ms/step - loss: 0.6613 - acc:
Epoch 3/30
100/100 [==============================] - 97s 975ms/step - loss: 0.6020 - acc:
Epoch 4/30
100/100 [==============================] - 98s 977ms/step - loss: 0.5593 - acc:
Epoch 5/30
100/100 [==============================] - 98s 978ms/step - loss: 0.5212 - acc:
Epoch 6/30
100/100 [==============================] - 98s 978ms/step - loss: 0.5021 - acc:
Epoch 7/30
100/100 [==============================] - 98s 978ms/step - loss: 0.4584 - acc:
Epoch 8/30
100/100 [==============================] - 98s 977ms/step - loss: 0.4418 - acc:
Epoch 9/30
100/100 [==============================] - 98s 980ms/step - loss: 0.3980 - acc:
Epoch 10/30
100/100 [==============================] - 98s 980ms/step - loss: 0.3733 - acc:
Epoch 11/30
100/100 [==============================] - 98s 979ms/step - loss: 0.3443 - acc:
Epoch 12/30
100/100 [==============================] - 98s 981ms/step - loss: 0.3361 - acc:
Epoch 13/30
100/100 [==============================] - 98s 980ms/step - loss: 0.3143 - acc:
Epoch 14/30
100/100 [==============================] - 98s 978ms/step - loss: 0.2583 - acc:
Epoch 15/30
100/100 [==============================] - 98s 977ms/step - loss: 0.2580 - acc:
Epoch 16/30
100/100 [==============================] - 98s 978ms/step - loss: 0.2428 - acc:
Epoch 17/30
100/100 [==============================] - 98s 976ms/step - loss: 0.2399 - acc:
Epoch 18/30
100/100 [==============================] - 98s 976ms/step - loss: 0.2022 - acc:
Epoch 19/30
100/100 [==============================] - 98s 975ms/step - loss: 0.1739 - acc:
Epoch 20/30
100/100 [==============================] - 97s 973ms/step - loss: 0.1540 - acc:
Epoch 21/30
100/100 [==============================] - 97s 973ms/step - loss: 0.1419 - acc:
Epoch 22/30
100/100 [==============================] - 98s 976ms/step - loss: 0.1133 - acc:
Epoch 23/30
100/100 [==============================] - 98s 977ms/step - loss: 0.1140 - acc:
Epoch 24/30
100/100 [==============================] - 98s 979ms/step - loss: 0.0974 - acc:
Epoch 25/30
100/100 [==============================] - 98s 981ms/step - loss: 0.0776 - acc:
Epoch 26/30
100/100 [==============================] - 98s 977ms/step - loss: 0.0777 - acc:
Epoch 27/30
100/100 [==============================] - 98s 975ms/step - loss: 0.0650 - acc:
```

```
Epoch 28/30
100/100 [==============================] - 98s 977ms/step - loss: 0.0525 - acc: (
Epoch 29/30
```

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

```
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(len(acc))

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()
```



The training accuracy increases linearly over time, until it reaches nearly 100%, while validation accuracy stalls at 70-72%. Our validation loss reaches its minimum after only five epochs then stalls, while the training loss keeps decreasing linearly until it reaches nearly 0.

Because we only have relatively few training samples (2000), overfitting is going to be our number one concern.

to override overfitting let's try: *data augmentation*.

Using data augmentation

```
# Data Augmentation
from keras.preprocessing.image import ImageDataGenerator

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

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 150x150
        target_size=(150, 150),
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

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

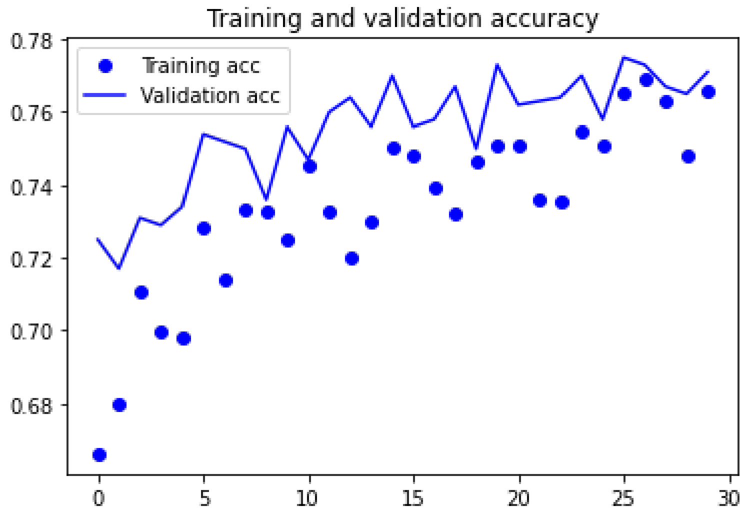
```
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=2e-5),
              metrics=['acc'])

history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50,
      verbose=2)
```

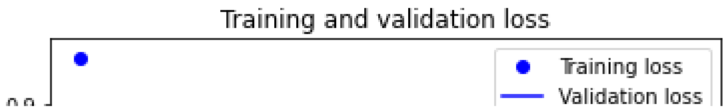
```
Found 2078 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/30
100/100 - 106s - loss: 0.9557 - acc: 0.6662 - val_loss: 0.6983 - val_acc: 0.7250
Epoch 2/30
100/100 - 105s - loss: 0.7020 - acc: 0.6797 - val_loss: 0.5714 - val_acc: 0.7170
Epoch 3/30
100/100 - 105s - loss: 0.5902 - acc: 0.7107 - val_loss: 0.5335 - val_acc: 0.7310
Epoch 4/30
100/100 - 105s - loss: 0.5896 - acc: 0.6997 - val_loss: 0.5223 - val_acc: 0.7290
Epoch 5/30
100/100 - 105s - loss: 0.5829 - acc: 0.6982 - val_loss: 0.5151 - val_acc: 0.7340
Epoch 6/30
100/100 - 105s - loss: 0.5543 - acc: 0.7282 - val_loss: 0.4943 - val_acc: 0.7540
Epoch 7/30
100/100 - 106s - loss: 0.5602 - acc: 0.7142 - val_loss: 0.4999 - val_acc: 0.7520
Epoch 8/30
100/100 - 105s - loss: 0.5445 - acc: 0.7332 - val_loss: 0.5000 - val_acc: 0.7500
Epoch 9/30
100/100 - 105s - loss: 0.5379 - acc: 0.7327 - val_loss: 0.5091 - val_acc: 0.7360
Epoch 10/30
100/100 - 105s - loss: 0.5399 - acc: 0.7252 - val_loss: 0.4964 - val_acc: 0.7560
Epoch 11/30
100/100 - 105s - loss: 0.5237 - acc: 0.7452 - val_loss: 0.4996 - val_acc: 0.7470
Epoch 12/30
100/100 - 105s - loss: 0.5345 - acc: 0.7327 - val_loss: 0.4793 - val_acc: 0.7600
Epoch 13/30
100/100 - 105s - loss: 0.5501 - acc: 0.7200 - val_loss: 0.4783 - val_acc: 0.7640
Epoch 14/30
100/100 - 109s - loss: 0.5355 - acc: 0.7302 - val_loss: 0.4824 - val_acc: 0.7560
Epoch 15/30
100/100 - 106s - loss: 0.5207 - acc: 0.7503 - val_loss: 0.4817 - val_acc: 0.7700
Epoch 16/30
100/100 - 105s - loss: 0.5165 - acc: 0.7482 - val_loss: 0.4832 - val_acc: 0.7560
Epoch 17/30
100/100 - 105s - loss: 0.5291 - acc: 0.7392 - val_loss: 0.4899 - val_acc: 0.7580
Epoch 18/30
100/100 - 105s - loss: 0.5218 - acc: 0.7322 - val_loss: 0.4734 - val_acc: 0.7670
Epoch 19/30
100/100 - 105s - loss: 0.5077 - acc: 0.7462 - val_loss: 0.5012 - val_acc: 0.7500
Epoch 20/30
100/100 - 106s - loss: 0.5128 - acc: 0.7508 - val_loss: 0.4684 - val_acc: 0.7730
```

```
Epoch 21/30
100/100 - 106s - loss: 0.5131 - acc: 0.7508 - val_loss: 0.4764 - val_acc: 0.7620
Epoch 22/30
100/100 - 106s - loss: 0.5244 - acc: 0.7362 - val_loss: 0.4778 - val_acc: 0.7630
Epoch 23/30
100/100 - 106s - loss: 0.5117 - acc: 0.7357 - val_loss: 0.4714 - val_acc: 0.7640
Epoch 24/30
100/100 - 105s - loss: 0.5043 - acc: 0.7548 - val_loss: 0.4670 - val_acc: 0.7700
Epoch 25/30
100/100 - 105s - loss: 0.5070 - acc: 0.7508 - val_loss: 0.4871 - val_acc: 0.7580
Epoch 26/30
100/100 - 105s - loss: 0.4988 - acc: 0.7653 - val_loss: 0.4720 - val_acc: 0.7750
Epoch 27/30
100/100 - 106s - loss: 0.4871 - acc: 0.7693 - val_loss: 0.4726 - val_acc: 0.7730
Epoch 28/30
```

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

```
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(len(acc))

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()
```





Thanks to data augmentation, we are no longer overfitting: the training curves are rather closely tracking the validation curves. We are now able to reach an accuracy of 82%, a 15% relative improvement over the non-regularized model

Using Pre-trained *model*

it would prove very difficult to go any higher just by training our own convnet from scratch, simply because we have so little data to work with. As a next step to improve our accuracy on this problem, we will have to leverage a pre-trained model

We will use the VGG16 architecture, developed by Karen Simonyan and Andrew Zisserman in 2014, a simple and widely used convnet architecture for ImageNet

```
from keras.applications import VGG16

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
58892288/58889256 [==============================] - 0s 0us/step
```

```
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
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
_________________________________________________________________
```

```
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'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 256)               2097408
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0
_________________________________________________________________
```

```
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 base: 30
```

```
conv_base.trainable = False
```

```
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 base: 4
```

```
from keras.preprocessing.image import ImageDataGenerator
from keras import models
from keras import layers
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')

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 150x150
```

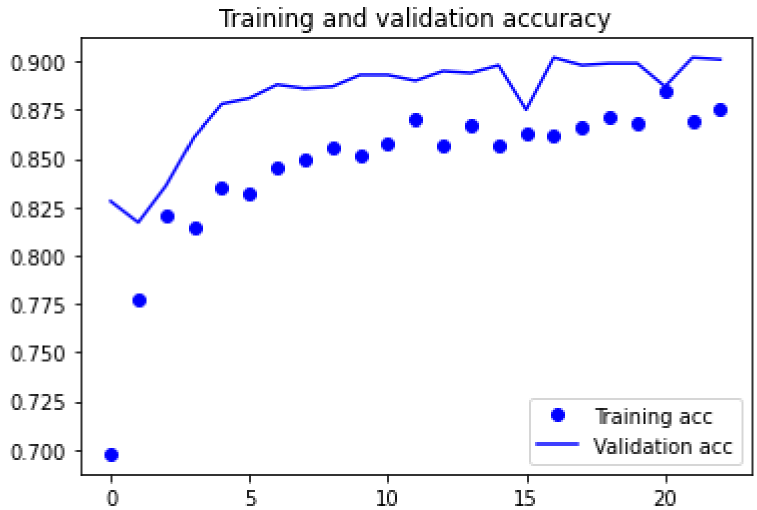
```
        target_size=(150, 150),
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

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=23,
      validation_data=validation_generator,
      validation_steps=50,
      verbose=2)
```

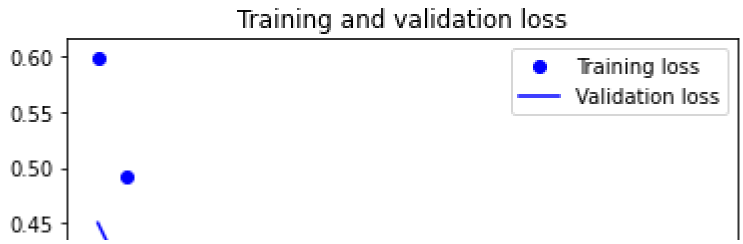
```
Found 2078 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/23
100/100 - 1016s - loss: 0.5978 - acc: 0.6977 - val_loss: 0.4504 - val_acc: 0.828
Epoch 2/23
100/100 - 661s - loss: 0.4914 - acc: 0.7768 - val_loss: 0.3991 - val_acc: 0.8170
Epoch 3/23
100/100 - 661s - loss: 0.4254 - acc: 0.8208 - val_loss: 0.3592 - val_acc: 0.8360
Epoch 4/23
100/100 - 660s - loss: 0.4035 - acc: 0.8143 - val_loss: 0.3127 - val_acc: 0.8610
Epoch 5/23
100/100 - 659s - loss: 0.3815 - acc: 0.8353 - val_loss: 0.2951 - val_acc: 0.8780
Epoch 6/23
100/100 - 658s - loss: 0.3752 - acc: 0.8323 - val_loss: 0.2869 - val_acc: 0.8810
Epoch 7/23
100/100 - 662s - loss: 0.3565 - acc: 0.8453 - val_loss: 0.2789 - val_acc: 0.8880
Epoch 8/23
100/100 - 661s - loss: 0.3504 - acc: 0.8493 - val_loss: 0.2754 - val_acc: 0.8860
Epoch 9/23
100/100 - 659s - loss: 0.3448 - acc: 0.8554 - val_loss: 0.2715 - val_acc: 0.8870
Epoch 10/23
100/100 - 658s - loss: 0.3397 - acc: 0.8514 - val_loss: 0.2624 - val_acc: 0.8930
Epoch 11/23
100/100 - 661s - loss: 0.3334 - acc: 0.8579 - val_loss: 0.2608 - val_acc: 0.8930
Epoch 12/23
100/100 - 659s - loss: 0.3126 - acc: 0.8704 - val_loss: 0.2574 - val_acc: 0.8900
Epoch 13/23
100/100 - 662s - loss: 0.3280 - acc: 0.8564 - val_loss: 0.2533 - val_acc: 0.8950
Epoch 14/23
100/100 - 662s - loss: 0.3039 - acc: 0.8669 - val_loss: 0.2520 - val_acc: 0.8940
```

```
Epoch 15/23
100/100 - 663s - loss: 0.3130 - acc: 0.8564 - val_loss: 0.2526 - val_acc: 0.8980
Epoch 16/23
100/100 - 663s - loss: 0.3131 - acc: 0.8634 - val_loss: 0.2851 - val_acc: 0.8750
Epoch 17/23
100/100 - 664s - loss: 0.3093 - acc: 0.8624 - val_loss: 0.2459 - val_acc: 0.9020
Epoch 18/23
100/100 - 668s - loss: 0.3026 - acc: 0.8664 - val_loss: 0.2437 - val_acc: 0.8980
Epoch 19/23
100/100 - 664s - loss: 0.2984 - acc: 0.8714 - val_loss: 0.2431 - val_acc: 0.8990
Epoch 20/23
100/100 - 663s - loss: 0.3016 - acc: 0.8679 - val_loss: 0.2477 - val_acc: 0.8990
Epoch 21/23
100/100 - 661s - loss: 0.2873 - acc: 0.8849 - val_loss: 0.2554 - val_acc: 0.8870
Epoch 22/23
100/100 - 662s - loss: 0.2987 - acc: 0.8694 - val_loss: 0.2433 - val_acc: 0.9020
Epoch 23/23
100/100 - 663s - loss: 0.2861 - acc: 0.8749 - val_loss: 0.2423 - val_acc: 0.9010
```

```
model.save('cats_and_dogs_small_3.h5')
```

```
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(len(acc))

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()
```





Fine-tuning


```
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


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)
```

```
100/100 [==============================] - 762s 8s/step - loss: 0.3485 - acc: 0.
Epoch 2/30
100/100 [==============================] - 755s 8s/step - loss: 0.2626 - acc: 0.
```

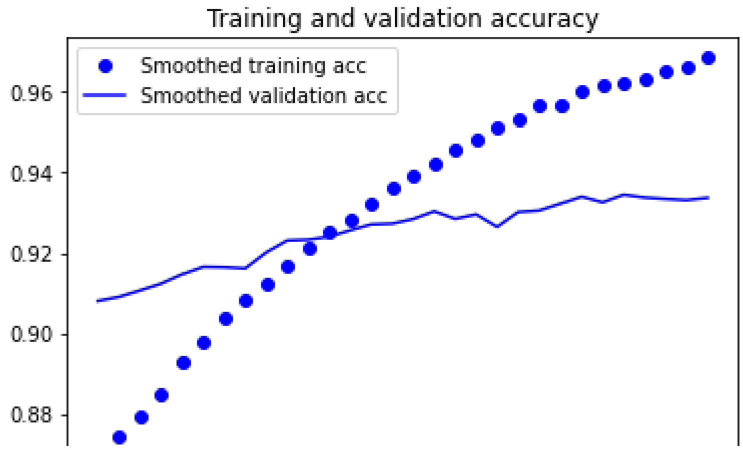
```
Epoch 3/30
100/100 [==============================] - 761s 8s/step - loss: 0.2316 - acc: 0.9
Epoch 4/30
100/100 [==============================] - 756s 8s/step - loss: 0.2348 - acc: 0.9
Epoch 5/30
100/100 [==============================] - 756s 8s/step - loss: 0.1878 - acc: 0.9
Epoch 6/30
100/100 [==============================] - 756s 8s/step - loss: 0.1980 - acc: 0.9
Epoch 7/30
100/100 [==============================] - 757s 8s/step - loss: 0.1654 - acc: 0.9
Epoch 8/30
100/100 [==============================] - 758s 8s/step - loss: 0.1683 - acc: 0.9
Epoch 9/30
100/100 [==============================] - 759s 8s/step - loss: 0.1527 - acc: 0.9
Epoch 10/30
100/100 [==============================] - 754s 8s/step - loss: 0.1768 - acc: 0.9
Epoch 11/30
100/100 [==============================] - 756s 8s/step - loss: 0.1508 - acc: 0.9
Epoch 12/30
100/100 [==============================] - 757s 8s/step - loss: 0.1480 - acc: 0.9
Epoch 13/30
100/100 [==============================] - 759s 8s/step - loss: 0.1774 - acc: 0.9
Epoch 14/30
100/100 [==============================] - 754s 8s/step - loss: 0.1497 - acc: 0.9
Epoch 15/30
100/100 [==============================] - 757s 8s/step - loss: 0.1229 - acc: 0.9
Epoch 16/30
100/100 [==============================] - 754s 8s/step - loss: 0.1385 - acc: 0.9
Epoch 17/30
100/100 [==============================] - 755s 8s/step - loss: 0.1177 - acc: 0.9
Epoch 18/30
100/100 [==============================] - 759s 8s/step - loss: 0.0999 - acc: 0.9
Epoch 19/30
100/100 [==============================] - 770s 8s/step - loss: 0.1021 - acc: 0.9
Epoch 20/30
100/100 [==============================] - 754s 8s/step - loss: 0.0866 - acc: 0.9
Epoch 21/30
100/100 [==============================] - 755s 8s/step - loss: 0.1192 - acc: 0.9
Epoch 22/30
100/100 [==============================] - 765s 8s/step - loss: 0.0724 - acc: 0.9
Epoch 23/30
100/100 [==============================] - 782s 8s/step - loss: 0.0992 - acc: 0.9
Epoch 24/30
100/100 [==============================] - 758s 8s/step - loss: 0.0745 - acc: 0.9
Epoch 25/30
100/100 [==============================] - 749s 8s/step - loss: 0.0849 - acc: 0.9
Epoch 26/30
100/100 [==============================] - 751s 8s/step - loss: 0.0870 - acc: 0.9
Epoch 27/30
100/100 [==============================] - 751s 8s/step - loss: 0.1006 - acc: 0.9
Epoch 28/30
100/100 [==============================] - 749s 8s/step - loss: 0.0715 - acc: 0.9
Epoch 29/30
100/100 [==============================] - 751s 8s/step - loss: 0.0812 - acc: 0.9

Epoch 30/30
```

```
model.save('cats_and_dogs_small_4.h5')
```

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


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(len(acc))

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()
```

These curves look very noisy. To make them more readable, we can smooth them by replacing every loss and accuracy with exponential moving averages of these quantities.

```
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()
```



We can now finally evaluate this model on the test data:

Training and validation loss

```
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 1169 images belonging to 2 classes.
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.p
  warnings.warn('`Model.evaluate_generator` is deprecated and '
test acc: 0.9409999847412109
```

Here we get a test accuracy of 94.09%. I managed to reach this result using only a very small fraction of the training data available.

✓ 5m 41s completed at 4:10 PM