```
import keras
keras.__version__
```

```
'2.4.3'
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
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: 1000
total training dog images: 1000
total validation cat images: 500
total validation dog images: 500
total test cat images: 499
total test dog images: 500
```

```
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 2000 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,)
```

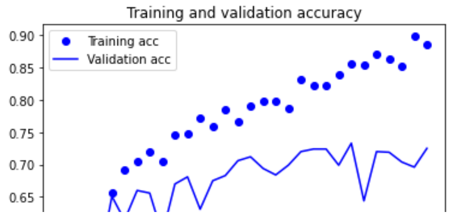
```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=50,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50)
```

```
50/50 [==============================] - 233s 5s/step - loss: 0.6821 - acc: 0.5631 - val_loss: 0.6769 - val_acc: 0.5430
Epoch 3/30
50/50 [==============================] - 109s 2s/step - loss: 0.6567 - acc: 0.6068 - val_loss: 0.6583 - val_acc: 0.5970
Epoch 4/30
50/50 [==============================] - 69s 1s/step - loss: 0.6426 - acc: 0.6104 - val_loss: 0.7008 - val_acc: 0.5490
Epoch 5/30
50/50 [==============================] - 62s 1s/step - loss: 0.6345 - acc: 0.6371 - val_loss: 0.6234 - val_acc: 0.6510
Epoch 6/30
50/50 [==============================] - 58s 1s/step - loss: 0.5992 - acc: 0.6813 - val_loss: 0.6455 - val_acc: 0.6140
Epoch 7/30
50/50 [==============================] - 57s 1s/step - loss: 0.5706 - acc: 0.7020 - val_loss: 0.6046 - val_acc: 0.6600
Epoch 8/30
50/50 [==============================] - 63s 1s/step - loss: 0.5407 - acc: 0.7370 - val_loss: 0.6036 - val_acc: 0.6560
Epoch 9/30
50/50 [==============================] - 56s 1s/step - loss: 0.5765 - acc: 0.6942 - val_loss: 0.7102 - val_acc: 0.5920
Epoch 10/30
50/50 [==============================] - 57s 1s/step - loss: 0.5207 - acc: 0.7393 - val_loss: 0.6179 - val_acc: 0.6700
Epoch 11/30
```

```
50/50 [==============================] - 56s 1s/step - loss: 0.5084 - acc: 0.7547 - val_loss: 0.6112 - val_acc: 0.6810
Epoch 12/30
50/50 [==============================] - 56s 1s/step - loss: 0.5020 - acc: 0.7706 - val_loss: 0.7002 - val_acc: 0.6310
Epoch 13/30
50/50 [==============================] - 56s 1s/step - loss: 0.4664 - acc: 0.7715 - val_loss: 0.5973 - val_acc: 0.6750
Epoch 14/30
50/50 [==============================] - 56s 1s/step - loss: 0.4631 - acc: 0.7958 - val_loss: 0.6052 - val_acc: 0.6830
Epoch 15/30
50/50 [==============================] - 56s 1s/step - loss: 0.4501 - acc: 0.7713 - val_loss: 0.5620 - val_acc: 0.7060
Epoch 16/30
50/50 [==============================] - 56s 1s/step - loss: 0.4181 - acc: 0.8057 - val_loss: 0.5727 - val_acc: 0.7120
Epoch 17/30
50/50 [==============================] - 56s 1s/step - loss: 0.4469 - acc: 0.7777 - val_loss: 0.5828 - val_acc: 0.6940
Epoch 18/30
50/50 [==============================] - 56s 1s/step - loss: 0.4103 - acc: 0.8068 - val_loss: 0.6017 - val_acc: 0.6840
Epoch 19/30
50/50 [==============================] - 56s 1s/step - loss: 0.4197 - acc: 0.7891 - val_loss: 0.5943 - val_acc: 0.6990
Epoch 20/30
50/50 [==============================] - 56s 1s/step - loss: 0.3893 - acc: 0.8356 - val_loss: 0.5682 - val_acc: 0.7200
Epoch 21/30
50/50 [==============================] - 56s 1s/step - loss: 0.3990 - acc: 0.8226 - val_loss: 0.5534 - val_acc: 0.7240
Epoch 22/30
50/50 [==============================] - 56s 1s/step - loss: 0.3587 - acc: 0.8419 - val_loss: 0.5600 - val_acc: 0.7240
Epoch 23/30
50/50 [==============================] - 56s 1s/step - loss: 0.3768 - acc: 0.8289 - val_loss: 0.5976 - val_acc: 0.6990
Epoch 24/30
50/50 [==============================] - 57s 1s/step - loss: 0.3454 - acc: 0.8585 - val_loss: 0.5433 - val_acc: 0.7330
Epoch 25/30
50/50 [==============================] - 56s 1s/step - loss: 0.3380 - acc: 0.8583 - val_loss: 0.7705 - val_acc: 0.6440
Epoch 26/30
50/50 [==============================] - 56s 1s/step - loss: 0.3208 - acc: 0.8751 - val_loss: 0.6051 - val_acc: 0.7200
Epoch 27/30
50/50 [==============================] - 57s 1s/step - loss: 0.3045 - acc: 0.8604 - val_loss: 0.5744 - val_acc: 0.7190
Epoch 28/30
50/50 [==============================] - 57s 1s/step - loss: 0.3256 - acc: 0.8625 - val_loss: 0.5919 - val_acc: 0.7040
Epoch 29/30
50/50 [==============================] - 56s 1s/step - loss: 0.2986 - acc: 0.8956 - val_loss: 0.6872 - val_acc: 0.6960
Epoch 30/30
50/50 [==============================] - 56s 1s/step - loss: 0.2789 - acc: 0.8851 - val_loss: 0.6095 - val_acc: 0.7250
```

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

```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(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()
```



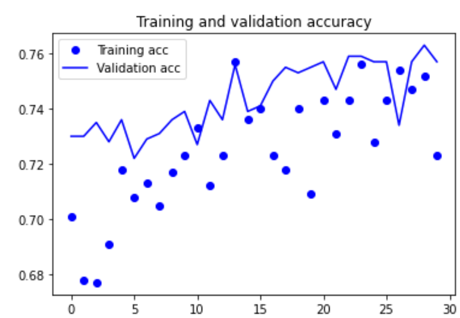

```
# Data Augmentation
from keras.preprocessing.image import ImageDataGenerator

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=50,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50,
      verbose=2)
```

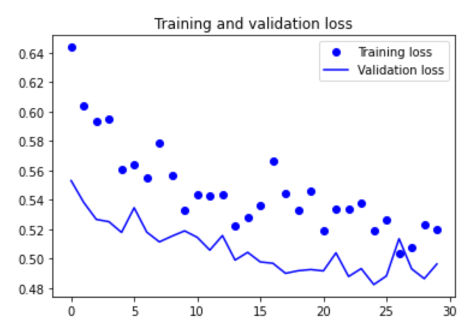
```
50/50 - 60s - loss: 0.6041 - acc: 0.6780 - val_loss: 0.5383 - val_acc: 0.7300
Epoch 3/30
50/50 - 60s - loss: 0.5933 - acc: 0.6770 - val_loss: 0.5267 - val_acc: 0.7350
Epoch 4/30
50/50 - 60s - loss: 0.5949 - acc: 0.6910 - val_loss: 0.5252 - val_acc: 0.7280
Epoch 5/30
50/50 - 61s - loss: 0.5609 - acc: 0.7180 - val_loss: 0.5179 - val_acc: 0.7360
Epoch 6/30
50/50 - 60s - loss: 0.5640 - acc: 0.7080 - val_loss: 0.5346 - val_acc: 0.7220
Epoch 7/30
50/50 - 60s - loss: 0.5547 - acc: 0.7130 - val_loss: 0.5181 - val_acc: 0.7290
Epoch 8/30
50/50 - 60s - loss: 0.5787 - acc: 0.7050 - val_loss: 0.5114 - val_acc: 0.7310
Epoch 9/30
50/50 - 60s - loss: 0.5562 - acc: 0.7170 - val_loss: 0.5154 - val_acc: 0.7360
Epoch 10/30
50/50 - 60s - loss: 0.5327 - acc: 0.7230 - val_loss: 0.5189 - val_acc: 0.7390
Epoch 11/30
50/50 - 60s - loss: 0.5436 - acc: 0.7330 - val_loss: 0.5145 - val_acc: 0.7270
Epoch 12/30
50/50 - 60s - loss: 0.5424 - acc: 0.7120 - val_loss: 0.5058 - val_acc: 0.7430
Epoch 13/30
50/50 - 60s - loss: 0.5432 - acc: 0.7230 - val_loss: 0.5158 - val_acc: 0.7360
Epoch 14/30
50/50 - 60s - loss: 0.5227 - acc: 0.7570 - val_loss: 0.4991 - val_acc: 0.7560
Epoch 15/30
50/50 - 60s - loss: 0.5280 - acc: 0.7360 - val_loss: 0.5043 - val_acc: 0.7390
Epoch 16/30
50/50 - 60s - loss: 0.5359 - acc: 0.7400 - val_loss: 0.4978 - val_acc: 0.7410

Epoch 17/30
50/50 - 60s - loss: 0.5667 - acc: 0.7230 - val_loss: 0.4969 - val_acc: 0.7500
Epoch 18/30
50/50 - 60s - loss: 0.5446 - acc: 0.7180 - val_loss: 0.4901 - val_acc: 0.7550
Epoch 19/30
50/50 - 60s - loss: 0.5326 - acc: 0.7400 - val_loss: 0.4918 - val_acc: 0.7530
Epoch 20/30
50/50 - 60s - loss: 0.5463 - acc: 0.7090 - val_loss: 0.4927 - val_acc: 0.7550
```

```
Epoch 21/30
50/50 - 60s - loss: 0.5194 - acc: 0.7430 - val_loss: 0.4917 - val_acc: 0.7570
Epoch 22/30
50/50 - 60s - loss: 0.5338 - acc: 0.7310 - val_loss: 0.5040 - val_acc: 0.7470
Epoch 23/30
50/50 - 60s - loss: 0.5337 - acc: 0.7430 - val_loss: 0.4879 - val_acc: 0.7590
Epoch 24/30
50/50 - 60s - loss: 0.5382 - acc: 0.7560 - val_loss: 0.4933 - val_acc: 0.7590
Epoch 25/30
50/50 - 60s - loss: 0.5191 - acc: 0.7280 - val_loss: 0.4825 - val_acc: 0.7570
Epoch 26/30
50/50 - 60s - loss: 0.5265 - acc: 0.7430 - val_loss: 0.4883 - val_acc: 0.7570
Epoch 27/30
50/50 - 60s - loss: 0.5037 - acc: 0.7540 - val_loss: 0.5136 - val_acc: 0.7340
Epoch 28/30
50/50 - 60s - loss: 0.5073 - acc: 0.7470 - val_loss: 0.4931 - val_acc: 0.7570
Epoch 29/30
50/50 - 60s - loss: 0.5234 - acc: 0.7520 - val_loss: 0.4865 - val_acc: 0.7630
Epoch 30/30
50/50 - 60s - loss: 0.5201 - acc: 0.7230 - val_loss: 0.4964 - val_acc: 0.7570
```

```
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()
```




```
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'))
```

```
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 2000 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,)
```

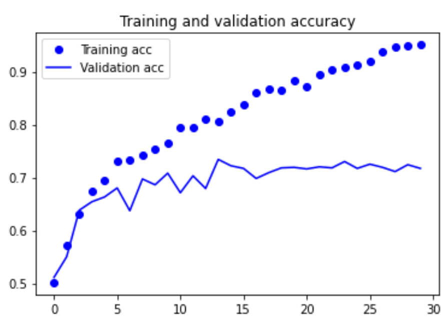
```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=75,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50)
```

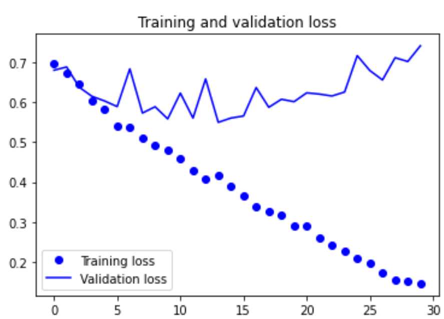
```
75/75 [==============================] - 77s 1s/step - loss: 0.6801 - acc: 0.5488 - val_loss: 0.6877 - val_acc: 0.5500
Epoch 3/30
75/75 [==============================] - 77s 1s/step - loss: 0.6569 - acc: 0.6134 - val_loss: 0.6370 - val_acc: 0.6380
Epoch 4/30
75/75 [==============================] - 77s 1s/step - loss: 0.6104 - acc: 0.6725 - val_loss: 0.6145 - val_acc: 0.6540
Epoch 5/30
75/75 [==============================] - 77s 1s/step - loss: 0.5741 - acc: 0.7102 - val_loss: 0.6028 - val_acc: 0.6630
Epoch 6/30
75/75 [==============================] - 77s 1s/step - loss: 0.5375 - acc: 0.7285 - val_loss: 0.5888 - val_acc: 0.6800
Epoch 7/30
75/75 [==============================] - 77s 1s/step - loss: 0.5454 - acc: 0.7328 - val_loss: 0.6830 - val_acc: 0.6370
Epoch 8/30
75/75 [==============================] - 77s 1s/step - loss: 0.5286 - acc: 0.7300 - val_loss: 0.5727 - val_acc: 0.6970
Epoch 9/30
75/75 [==============================] - 77s 1s/step - loss: 0.5050 - acc: 0.7565 - val_loss: 0.5884 - val_acc: 0.6860
Epoch 10/30
75/75 [==============================] - 77s 1s/step - loss: 0.4848 - acc: 0.7619 - val_loss: 0.5580 - val_acc: 0.7080
Epoch 11/30
75/75 [==============================] - 77s 1s/step - loss: 0.4728 - acc: 0.7840 - val_loss: 0.6224 - val_acc: 0.6710
Epoch 12/30
75/75 [==============================] - 81s 1s/step - loss: 0.4294 - acc: 0.7934 - val_loss: 0.5601 - val_acc: 0.7030
Epoch 13/30
75/75 [==============================] - 78s 1s/step - loss: 0.4264 - acc: 0.8023 - val_loss: 0.6578 - val_acc: 0.6790
Epoch 14/30
75/75 [==============================] - 78s 1s/step - loss: 0.4109 - acc: 0.8088 - val_loss: 0.5494 - val_acc: 0.7340
Epoch 15/30
75/75 [==============================] - 78s 1s/step - loss: 0.3823 - acc: 0.8308 - val_loss: 0.5601 - val_acc: 0.7220
Epoch 16/30
75/75 [==============================] - 78s 1s/step - loss: 0.3764 - acc: 0.8383 - val_loss: 0.5652 - val_acc: 0.7170

Epoch 17/30
75/75 [==============================] - 78s 1s/step - loss: 0.3322 - acc: 0.8599 - val_loss: 0.6363 - val_acc: 0.6980
Epoch 18/30
75/75 [==============================] - 78s 1s/step - loss: 0.3294 - acc: 0.8677 - val_loss: 0.5870 - val_acc: 0.7090
Epoch 19/30
75/75 [==============================] - 78s 1s/step - loss: 0.2942 - acc: 0.8846 - val_loss: 0.6071 - val_acc: 0.7180
```

```
Epoch 20/30
75/75 [==============================] - 78s 1s/step - loss: 0.2935 - acc: 0.8730 - val_loss: 0.6010 - val_acc: 0.7190
Epoch 21/30
75/75 [==============================] - 78s 1s/step - loss: 0.2742 - acc: 0.8765 - val_loss: 0.6231 - val_acc: 0.7160
Epoch 22/30
75/75 [==============================] - 78s 1s/step - loss: 0.2626 - acc: 0.8937 - val_loss: 0.6200 - val_acc: 0.7200
Epoch 23/30
75/75 [==============================] - 78s 1s/step - loss: 0.2343 - acc: 0.9096 - val_loss: 0.6152 - val_acc: 0.7180
Epoch 24/30
75/75 [==============================] - 77s 1s/step - loss: 0.2224 - acc: 0.9087 - val_loss: 0.6251 - val_acc: 0.7300
Epoch 25/30
75/75 [==============================] - 78s 1s/step - loss: 0.2100 - acc: 0.9050 - val_loss: 0.7161 - val_acc: 0.7170
Epoch 26/30
75/75 [==============================] - 78s 1s/step - loss: 0.1896 - acc: 0.9312 - val_loss: 0.6788 - val_acc: 0.7250
Epoch 27/30
75/75 [==============================] - 78s 1s/step - loss: 0.1561 - acc: 0.9545 - val_loss: 0.6553 - val_acc: 0.7190
Epoch 28/30
75/75 [==============================] - 78s 1s/step - loss: 0.1497 - acc: 0.9529 - val_loss: 0.7110 - val_acc: 0.7110
Epoch 29/30
75/75 [==============================] - 78s 1s/step - loss: 0.1374 - acc: 0.9560 - val_loss: 0.7013 - val_acc: 0.7240
Epoch 30/30
75/75 [==============================] - 78s 1s/step - loss: 0.1311 - acc: 0.9540 - val_loss: 0.7407 - val_acc: 0.7170
```

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

```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(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()
```




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

train_datagen = ImageDataGenerator(
    rescale=1./255,
```

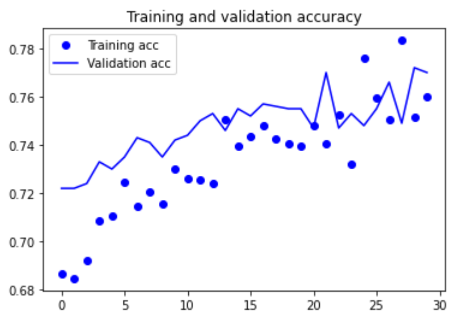
```
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

# 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=75,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50,
      verbose=2)
```

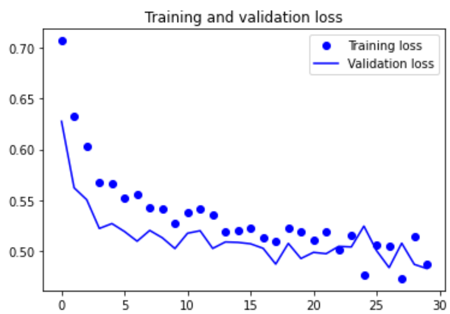
```
75/75 - 85s - loss: 0.6331 - acc: 0.6847 - val_loss: 0.5622 - val_acc: 0.7220
Epoch 3/30
75/75 - 84s - loss: 0.6025 - acc: 0.6920 - val_loss: 0.5505 - val_acc: 0.7240
Epoch 4/30
75/75 - 84s - loss: 0.5676 - acc: 0.7087 - val_loss: 0.5222 - val_acc: 0.7330
Epoch 5/30
75/75 - 84s - loss: 0.5669 - acc: 0.7107 - val_loss: 0.5270 - val_acc: 0.7300
Epoch 6/30
75/75 - 84s - loss: 0.5517 - acc: 0.7247 - val_loss: 0.5192 - val_acc: 0.7350
Epoch 7/30
75/75 - 84s - loss: 0.5557 - acc: 0.7147 - val_loss: 0.5097 - val_acc: 0.7430
Epoch 8/30
75/75 - 84s - loss: 0.5424 - acc: 0.7207 - val_loss: 0.5203 - val_acc: 0.7410
Epoch 9/30
75/75 - 84s - loss: 0.5411 - acc: 0.7153 - val_loss: 0.5131 - val_acc: 0.7350
Epoch 10/30
75/75 - 84s - loss: 0.5270 - acc: 0.7300 - val_loss: 0.5025 - val_acc: 0.7420
Epoch 11/30
75/75 - 84s - loss: 0.5385 - acc: 0.7260 - val_loss: 0.5175 - val_acc: 0.7440
Epoch 12/30
75/75 - 84s - loss: 0.5417 - acc: 0.7253 - val_loss: 0.5200 - val_acc: 0.7500
Epoch 13/30
75/75 - 84s - loss: 0.5361 - acc: 0.7240 - val_loss: 0.5026 - val_acc: 0.7530
Epoch 14/30
75/75 - 84s - loss: 0.5187 - acc: 0.7507 - val_loss: 0.5090 - val_acc: 0.7460
Epoch 15/30
75/75 - 84s - loss: 0.5203 - acc: 0.7393 - val_loss: 0.5085 - val_acc: 0.7550
Epoch 16/30
75/75 - 84s - loss: 0.5220 - acc: 0.7433 - val_loss: 0.5072 - val_acc: 0.7520
Epoch 17/30
75/75 - 84s - loss: 0.5126 - acc: 0.7480 - val_loss: 0.5027 - val_acc: 0.7570
Epoch 18/30
75/75 - 84s - loss: 0.5093 - acc: 0.7427 - val_loss: 0.4872 - val_acc: 0.7560
Epoch 19/30
75/75 - 84s - loss: 0.5229 - acc: 0.7407 - val_loss: 0.5075 - val_acc: 0.7550
Epoch 20/30
75/75 - 84s - loss: 0.5187 - acc: 0.7393 - val_loss: 0.4927 - val_acc: 0.7550
Epoch 21/30
75/75 - 84s - loss: 0.5106 - acc: 0.7480 - val_loss: 0.4987 - val_acc: 0.7470
Epoch 22/30
75/75 - 83s - loss: 0.5191 - acc: 0.7407 - val_loss: 0.4974 - val_acc: 0.7700
Epoch 23/30
75/75 - 83s - loss: 0.5008 - acc: 0.7527 - val_loss: 0.5046 - val_acc: 0.7470
Epoch 24/30
75/75 - 83s - loss: 0.5154 - acc: 0.7320 - val_loss: 0.5040 - val_acc: 0.7530
Epoch 25/30
75/75 - 83s - loss: 0.4762 - acc: 0.7760 - val_loss: 0.5245 - val_acc: 0.7480
Epoch 26/30
75/75 - 83s - loss: 0.5062 - acc: 0.7593 - val_loss: 0.5005 - val_acc: 0.7550
Epoch 27/30
75/75 - 83s - loss: 0.5047 - acc: 0.7507 - val_loss: 0.4837 - val_acc: 0.7660
Epoch 28/30
75/75 - 83s - loss: 0.4730 - acc: 0.7833 - val_loss: 0.5078 - val_acc: 0.7490
Epoch 29/30
```

```
Epoch 29/30
75/75 - 83s - loss: 0.5142 - acc: 0.7513 - val_loss: 0.4869 - val_acc: 0.7720
Epoch 30/30
75/75 - 83s - loss: 0.4868 - acc: 0.7600 - val_loss: 0.4828 - val_acc: 0.7700
```

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




```
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'))
```

```
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 2000 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,)
```

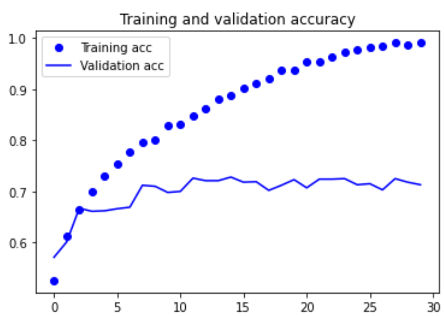
```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50)
```

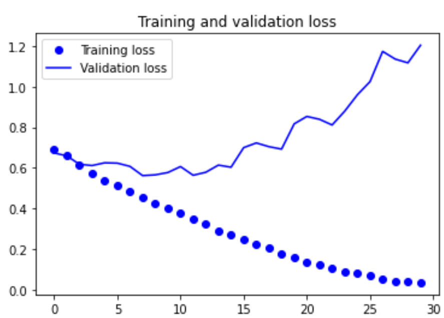
```
100/100 [==============================] - 99s 994ms/step - loss: 0.6687 - acc: 0.6151 - val_loss: 0.6593 - val_acc: 0.6010
Epoch 3/30
100/100 [==============================] - 100s 997ms/step - loss: 0.6230 - acc: 0.6575 - val_loss: 0.6180 - val_acc: 0.6670
Epoch 4/30
100/100 [==============================] - 100s 998ms/step - loss: 0.5829 - acc: 0.7049 - val_loss: 0.6117 - val_acc: 0.6610
Epoch 5/30
100/100 [==============================] - 100s 1s/step - loss: 0.5388 - acc: 0.7293 - val_loss: 0.6251 - val_acc: 0.6620
Epoch 6/30
100/100 [==============================] - 100s 1s/step - loss: 0.5179 - acc: 0.7433 - val_loss: 0.6235 - val_acc: 0.6660
Epoch 7/30
100/100 [==============================] - 100s 1s/step - loss: 0.4838 - acc: 0.7792 - val_loss: 0.6076 - val_acc: 0.6690
Epoch 8/30
100/100 [==============================] - 100s 999ms/step - loss: 0.4497 - acc: 0.7916 - val_loss: 0.5622 - val_acc: 0.7120
Epoch 9/30
100/100 [==============================] - 100s 999ms/step - loss: 0.4293 - acc: 0.7958 - val_loss: 0.5659 - val_acc: 0.7100
Epoch 10/30
100/100 [==============================] - 100s 998ms/step - loss: 0.3960 - acc: 0.8378 - val_loss: 0.5773 - val_acc: 0.6980
Epoch 11/30
100/100 [==============================] - 100s 997ms/step - loss: 0.3805 - acc: 0.8239 - val_loss: 0.6066 - val_acc: 0.7000
Epoch 12/30
100/100 [==============================] - 100s 996ms/step - loss: 0.3462 - acc: 0.8456 - val_loss: 0.5639 - val_acc: 0.7260
Epoch 13/30
100/100 [==============================] - 99s 994ms/step - loss: 0.3186 - acc: 0.8705 - val_loss: 0.5786 - val_acc: 0.7210
Epoch 14/30
100/100 [==============================] - 99s 993ms/step - loss: 0.2872 - acc: 0.8781 - val_loss: 0.6139 - val_acc: 0.7210
Epoch 15/30
100/100 [==============================] - 100s 999ms/step - loss: 0.2718 - acc: 0.8965 - val_loss: 0.6034 - val_acc: 0.7280
Epoch 16/30
100/100 [==============================] - 99s 994ms/step - loss: 0.2594 - acc: 0.8915 - val_loss: 0.6998 - val_acc: 0.7180
Epoch 17/30
100/100 [==============================] - 99s 994ms/step - loss: 0.2265 - acc: 0.9134 - val_loss: 0.7229 - val_acc: 0.7190
Epoch 18/30
100/100 [==============================] - 100s 999ms/step - loss: 0.2011 - acc: 0.9297 - val_loss: 0.7045 - val_acc: 0.7020
Epoch 19/30
100/100 [==============================] - 100s 1s/step - loss: 0.1739 - acc: 0.9369 - val_loss: 0.6925 - val_acc: 0.7120
Epoch 20/30
100/100 [==============================] - 100s 999ms/step - loss: 0.1458 - acc: 0.9508 - val_loss: 0.8168 - val_acc: 0.7230
Epoch 21/30
100/100 [==============================] - 102s 1s/step - loss: 0.1430 - acc: 0.9559 - val_loss: 0.8532 - val_acc: 0.7070
Epoch 22/30
100/100 [==============================] - 100s 1s/step - loss: 0.1106 - acc: 0.9629 - val_loss: 0.8392 - val_acc: 0.7240
Epoch 23/30
100/100 [==============================] - 100s 1s/step - loss: 0.0992 - acc: 0.9677 - val_loss: 0.8111 - val_acc: 0.7240
Epoch 24/30
100/100 [==============================] - 101s 1s/step - loss: 0.0896 - acc: 0.9716 - val_loss: 0.8794 - val_acc: 0.7250
Epoch 25/30
100/100 [==============================] - 101s 1s/step - loss: 0.1070 - acc: 0.9717 - val_loss: 0.9596 - val_acc: 0.7130
Epoch 26/30
100/100 [==============================] - 101s 1s/step - loss: 0.0739 - acc: 0.9774 - val_loss: 1.0243 - val_acc: 0.7150
Epoch 27/30
100/100 [==============================] - 101s 1s/step - loss: 0.0521 - acc: 0.9860 - val_loss: 1.1731 - val_acc: 0.7030
```

```
Epoch 28/30
100/100 [==============================] - 101s 1s/step - loss: 0.0398 - acc: 0.9917 - val_loss: 1.1352 - val_acc: 0.7250
Epoch 29/30
100/100 [==============================] - 101s 1s/step - loss: 0.0461 - acc: 0.9863 - val_loss: 1.1168 - val_acc: 0.7180
Epoch 30/30
100/100 [==============================] - 101s 1s/step - loss: 0.0326 - acc: 0.9919 - val_loss: 1.2036 - val_acc: 0.7130
```

```
model.save('cats_and_dogs_small_5.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()
```




```
# 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
```

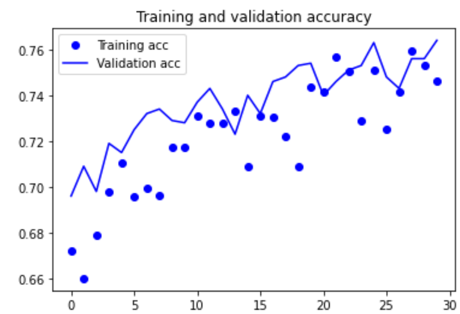
```
        # All images will be resized to 150x150
        target_size=(150, 150),
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

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)
```

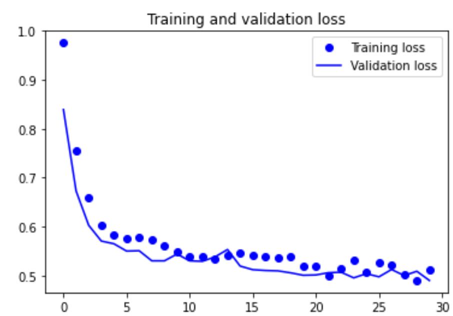
```
Epoch 2/30
100/100 - 108s - loss: 0.7553 - acc: 0.6600 - val_loss: 0.6731 - val_acc: 0.7090
Epoch 3/30
100/100 - 107s - loss: 0.6592 - acc: 0.6790 - val_loss: 0.6028 - val_acc: 0.6980
Epoch 4/30
100/100 - 107s - loss: 0.6027 - acc: 0.6980 - val_loss: 0.5706 - val_acc: 0.7190
Epoch 5/30
100/100 - 108s - loss: 0.5843 - acc: 0.7105 - val_loss: 0.5650 - val_acc: 0.7150
Epoch 6/30
100/100 - 107s - loss: 0.5748 - acc: 0.6955 - val_loss: 0.5501 - val_acc: 0.7250
Epoch 7/30
100/100 - 107s - loss: 0.5784 - acc: 0.6995 - val_loss: 0.5508 - val_acc: 0.7320
Epoch 8/30
100/100 - 107s - loss: 0.5724 - acc: 0.6960 - val_loss: 0.5302 - val_acc: 0.7340
Epoch 9/30
100/100 - 109s - loss: 0.5621 - acc: 0.7175 - val_loss: 0.5302 - val_acc: 0.7290
Epoch 10/30
100/100 - 107s - loss: 0.5479 - acc: 0.7175 - val_loss: 0.5441 - val_acc: 0.7280
Epoch 11/30
100/100 - 107s - loss: 0.5389 - acc: 0.7310 - val_loss: 0.5301 - val_acc: 0.7370
Epoch 12/30
100/100 - 107s - loss: 0.5396 - acc: 0.7280 - val_loss: 0.5291 - val_acc: 0.7430
Epoch 13/30
100/100 - 107s - loss: 0.5346 - acc: 0.7280 - val_loss: 0.5380 - val_acc: 0.7340
Epoch 14/30
100/100 - 108s - loss: 0.5402 - acc: 0.7330 - val_loss: 0.5533 - val_acc: 0.7230
Epoch 15/30
100/100 - 107s - loss: 0.5460 - acc: 0.7090 - val_loss: 0.5195 - val_acc: 0.7400
Epoch 16/30
100/100 - 107s - loss: 0.5408 - acc: 0.7310 - val_loss: 0.5120 - val_acc: 0.7320
Epoch 17/30
100/100 - 109s - loss: 0.5378 - acc: 0.7305 - val_loss: 0.5104 - val_acc: 0.7460
Epoch 18/30
100/100 - 109s - loss: 0.5369 - acc: 0.7220 - val_loss: 0.5096 - val_acc: 0.7480
Epoch 19/30
100/100 - 108s - loss: 0.5387 - acc: 0.7090 - val_loss: 0.5056 - val_acc: 0.7530
Epoch 20/30
100/100 - 109s - loss: 0.5190 - acc: 0.7435 - val_loss: 0.5007 - val_acc: 0.7540
Epoch 21/30
100/100 - 108s - loss: 0.5193 - acc: 0.7415 - val_loss: 0.5013 - val_acc: 0.7400
Epoch 22/30
100/100 - 108s - loss: 0.5001 - acc: 0.7570 - val_loss: 0.5059 - val_acc: 0.7460
Epoch 23/30
100/100 - 108s - loss: 0.5143 - acc: 0.7505 - val_loss: 0.5070 - val_acc: 0.7510
Epoch 24/30
100/100 - 108s - loss: 0.5321 - acc: 0.7290 - val_loss: 0.4954 - val_acc: 0.7530
Epoch 25/30
100/100 - 108s - loss: 0.5071 - acc: 0.7510 - val_loss: 0.5039 - val_acc: 0.7630
Epoch 26/30
100/100 - 109s - loss: 0.5270 - acc: 0.7250 - val_loss: 0.4975 - val_acc: 0.7480
Epoch 27/30
100/100 - 109s - loss: 0.5229 - acc: 0.7415 - val_loss: 0.5124 - val_acc: 0.7430
Epoch 28/30
100/100 - 108s - loss: 0.5018 - acc: 0.7595 - val_loss: 0.5000 - val_acc: 0.7560
Epoch 29/30
100/100 - 108s - loss: 0.4899 - acc: 0.7530 - val_loss: 0.5088 - val_acc: 0.7560
Epoch 30/30
100/100 - 107s - loss: 0.5109 - acc: 0.7465 - val_loss: 0.4900 - val_acc: 0.7640
```

```
model.save('cats_and_dogs_small_5.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']
```

```
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()
```




```
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-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 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_1"
_________________________________________________________________
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: 4
```

```
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)
```

```
        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=50,
      epochs=6,
      validation_data=validation_generator,
      validation_steps=50,
      verbose=2)
```

```
Found 2000 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:1844: UserWarning: `Model.fit_generator` is deprecated and will be rem
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/6
50/50 - 434s - loss: 0.3503 - acc: 0.8460 - val_loss: 0.2703 - val_acc: 0.8920
Epoch 2/6
50/50 - 433s - loss: 0.3392 - acc: 0.8590 - val_loss: 0.2733 - val_acc: 0.8910
Epoch 3/6
50/50 - 433s - loss: 0.3473 - acc: 0.8480 - val_loss: 0.2726 - val_acc: 0.8880
Epoch 4/6
50/50 - 433s - loss: 0.3528 - acc: 0.8490 - val_loss: 0.2673 - val_acc: 0.8890
Epoch 5/6
50/50 - 433s - loss: 0.3490 - acc: 0.8360 - val_loss: 0.2622 - val_acc: 0.8940
Epoch 6/6
50/50 - 433s - loss: 0.3113 - acc: 0.8690 - val_loss: 0.2624 - val_acc: 0.8960
```

```
model.save('cats_and_dogs_small_7.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()
```

Training and validation accuracy

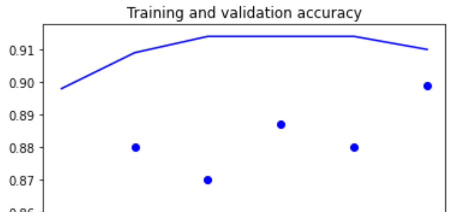
```
conv_base.trainable = True

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=50,
      epochs=6,
      validation_data=validation_generator,
      validation_steps=50)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be rem
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/6
50/50 [==============================] - 481s 10s/step - loss: 0.3742 - acc: 0.8324 - val_loss: 0.2474 - val_acc: 0.8980
Epoch 2/6
50/50 [==============================] - 478s 10s/step - loss: 0.2838 - acc: 0.8849 - val_loss: 0.2116 - val_acc: 0.9090
Epoch 3/6
50/50 [==============================] - 477s 10s/step - loss: 0.2770 - acc: 0.8686 - val_loss: 0.2080 - val_acc: 0.9140
Epoch 4/6
50/50 [==============================] - 477s 10s/step - loss: 0.2708 - acc: 0.8856 - val_loss: 0.2124 - val_acc: 0.9140
Epoch 5/6
50/50 [==============================] - 477s 10s/step - loss: 0.2572 - acc: 0.8686 - val_loss: 0.2117 - val_acc: 0.9140
Epoch 6/6
50/50 [==============================] - 482s 10s/step - loss: 0.2990 - acc: 0.8944 - val_loss: 0.2092 - val_acc: 0.9100
```

```
model.save('cats_and_dogs_small_8.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()
```



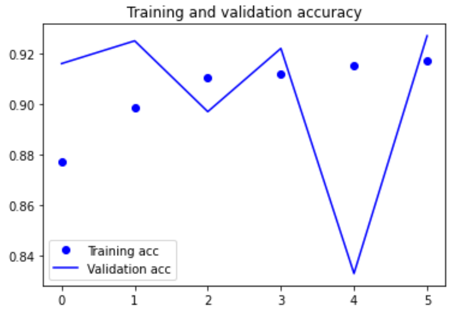

```
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

# 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=75,
      epochs=6,
      validation_data=validation_generator,
      validation_steps=50,
      verbose=2)
```

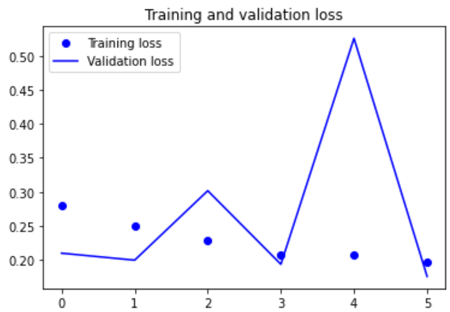
```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be rem
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/6
75/75 - 621s - loss: 0.2795 - acc: 0.8773 - val_loss: 0.2099 - val_acc: 0.9160
Epoch 2/6
75/75 - 628s - loss: 0.2492 - acc: 0.8987 - val_loss: 0.1996 - val_acc: 0.9250
Epoch 3/6
75/75 - 620s - loss: 0.2291 - acc: 0.9107 - val_loss: 0.3017 - val_acc: 0.8970
Epoch 4/6
75/75 - 621s - loss: 0.2080 - acc: 0.9120 - val_loss: 0.1936 - val_acc: 0.9220
Epoch 5/6
75/75 - 623s - loss: 0.2073 - acc: 0.9153 - val_loss: 0.5256 - val_acc: 0.8330
Epoch 6/6
75/75 - 620s - loss: 0.1963 - acc: 0.9173 - val_loss: 0.1757 - val_acc: 0.9270
```

```
model.save('cats_and_dogs_small_9.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()
```



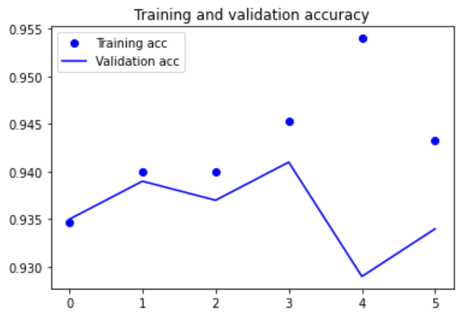

```
conv_base.trainable = True

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=75,
      epochs=6,
      validation_data=validation_generator,
      validation_steps=50)
```

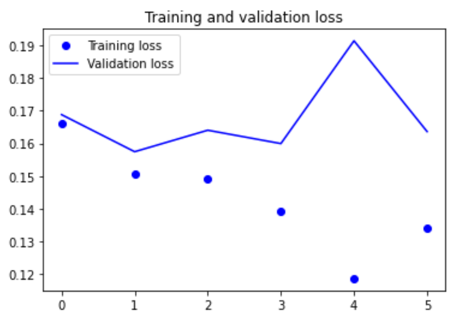
```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be rem
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/6
75/75 [==============================] - 618s 8s/step - loss: 0.1855 - acc: 0.9283 - val_loss: 0.1688 - val_acc: 0.9350
Epoch 2/6
75/75 [==============================] - 617s 8s/step - loss: 0.1608 - acc: 0.9387 - val_loss: 0.1574 - val_acc: 0.9390
Epoch 3/6
75/75 [==============================] - 617s 8s/step - loss: 0.1449 - acc: 0.9450 - val_loss: 0.1640 - val_acc: 0.9370
Epoch 4/6
75/75 [==============================] - 617s 8s/step - loss: 0.1353 - acc: 0.9407 - val_loss: 0.1599 - val_acc: 0.9410
Epoch 5/6
75/75 [==============================] - 617s 8s/step - loss: 0.1123 - acc: 0.9590 - val_loss: 0.1915 - val_acc: 0.9290
Epoch 6/6
75/75 [==============================] - 617s 8s/step - loss: 0.1520 - acc: 0.9298 - val_loss: 0.1636 - val_acc: 0.9340
```

```
model.save('cats_and_dogs_small_10.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()
```




```
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_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               2097408
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 9,177,089
Non-trainable params: 7,635,264
_________________________________________________________________
```

```
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: 10
```

```
conv_base.trainable = False
```

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

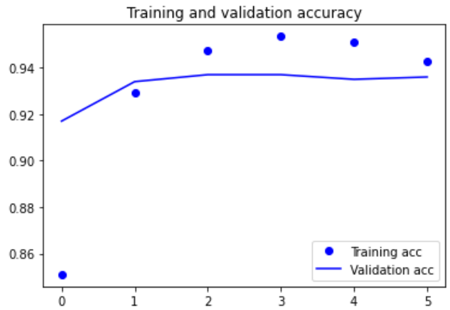
```
This is the number of trainable weights after freezing the conv base: 4
```

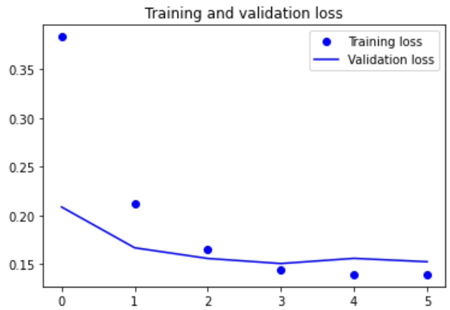
```
from keras.preprocessing.image import ImageDataGenerator

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=6,
      validation_data=validation_generator,
      validation_steps=50,
      verbose=2)
```

```
Found 2000 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:1844: UserWarning: `Model.fit_generator` is deprecated and will be rem
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/6
100/100 - 659s - loss: 0.3833 - acc: 0.8510 - val_loss: 0.2088 - val_acc: 0.9170
Epoch 2/6
100/100 - 658s - loss: 0.2121 - acc: 0.9290 - val_loss: 0.1670 - val_acc: 0.9340
Epoch 3/6
100/100 - 658s - loss: 0.1649 - acc: 0.9475 - val_loss: 0.1560 - val_acc: 0.9370
Epoch 4/6
100/100 - 660s - loss: 0.1449 - acc: 0.9535 - val_loss: 0.1508 - val_acc: 0.9370
Epoch 5/6
100/100 - 656s - loss: 0.1396 - acc: 0.9510 - val_loss: 0.1561 - val_acc: 0.9350
Epoch 6/6
100/100 - 655s - loss: 0.1398 - acc: 0.9425 - val_loss: 0.1527 - val_acc: 0.9360
```

```
model.save('cats_and_dogs_small_11.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()
```



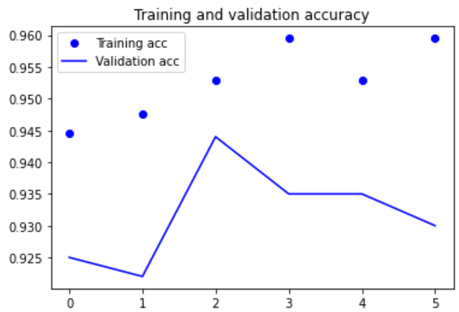

```
conv_base.trainable = True

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=6,
      validation_data=validation_generator,
      validation_steps=50)
```

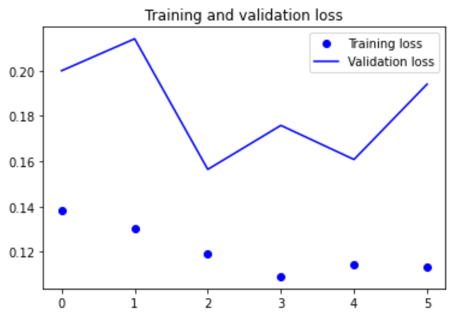
```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be rem
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/6
100/100 [==============================] - 748s 7s/step - loss: 0.1362 - acc: 0.9445 - val_loss: 0.1999 - val_acc: 0.9250
Epoch 2/6
100/100 [==============================] - 745s 7s/step - loss: 0.1357 - acc: 0.9457 - val_loss: 0.2141 - val_acc: 0.9220
Epoch 3/6
100/100 [==============================] - 745s 7s/step - loss: 0.1179 - acc: 0.9512 - val_loss: 0.1564 - val_acc: 0.9440
Epoch 4/6
100/100 [==============================] - 746s 7s/step - loss: 0.1261 - acc: 0.9514 - val_loss: 0.1758 - val_acc: 0.9350
Epoch 5/6
100/100 [==============================] - 746s 7s/step - loss: 0.1226 - acc: 0.9508 - val_loss: 0.1608 - val_acc: 0.9350
Epoch 6/6
100/100 [==============================] - 747s 7s/step - loss: 0.1081 - acc: 0.9598 - val_loss: 0.1940 - val_acc: 0.9300
```

```
model.save('cats_and_dogs_small_12.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()
```




At 1,000 dataset: validation accuracy was 72%, when we increased the dataset to 1,500, there was no change to validation accuracy. When we applied data augmentation and regularization, validation accuracy went up to 75% (4% increase) When we used a pretrained network, validation accuracy was 90% (25% increase) When we applied data augmentation and regularization to the pretrained network, validation accuracy went up further to 92% (28% increase)

Validation loss was also affected, we started with a validation loss of 0.55 @24epochs, when we increased our dataset to 1,500, there was no change in validation loss, when we applied data augmentation and regularization, validation loss reduced to 0.49 @24epochs, when we used a pretrained network, validation loss further reduced to 0.26 @5epochs and when we applied data augmentation and regularization to the pre-trained network, validation loss dropped further to 0.1 @2epochs.

At 1,500 dataset: validation accuracy was 72%, when we increased the dataset to 2,000, there was no change to validation accuracy. When we applied data augmentation and regularization, validation accuracy went up to 77% (7% increase) When we used a pretrained network, validation accuracy was 93% (29% increase) When we applied data augmentation and regularization to the pretrained network, validation accuracy went up further to 94% (31% increase)

Validation loss was also affected, we started with a validation loss of 0.55 @14epochs, when we increased our dataset to 2,000, there was no change in validation loss, when we applied data augmentation and regularization, validation loss reduced to 0.4 @24epochs, when we used a pretrained network, validation loss further reduced to 0.15 @6epochs and when we applied data augmentation and regularization to the pre-trained network, validation loss dropped further to 0.1 @5epochs.

At 2,000 dataset: validation accuracy was 72% When we applied data augmentation and regularization, validation accuracy went up to 77% (7% increase) When we used a pretrained network, validation accuracy was 93% (29% increase) When we applied data augmentation and regularization to the pretrained network, validation accuracy went up further to 94.5% (31.25% increase)

Validation loss was also affected, we started with a validation loss of 0.55 @7epochs, when we applied data augmentation and regularization, validation loss reduced to 0.4 @23epochs, when we used a pretrained network, validation loss further reduced to 0.15 @3epochs and when we applied data augmentation and regularization to the pre-trained network, validation loss dropped further to 0.1 @3epochs

Based on the above, data augmentation and regularization is more effective than increase in dataset for training a network, however, pretrained networks are most effective for classification models.

Double-click (or enter) to edit

✓ 16s completed at 9:30 AM