total training cat images: 1000

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 [ ] 56 1 / t l 0 5084 0 7547 l l 0 6112 l 0 6810

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

# Data Augmentation - acc: 0.8851 - val\_loss: 0.6095 - val\_acc: 0.7250

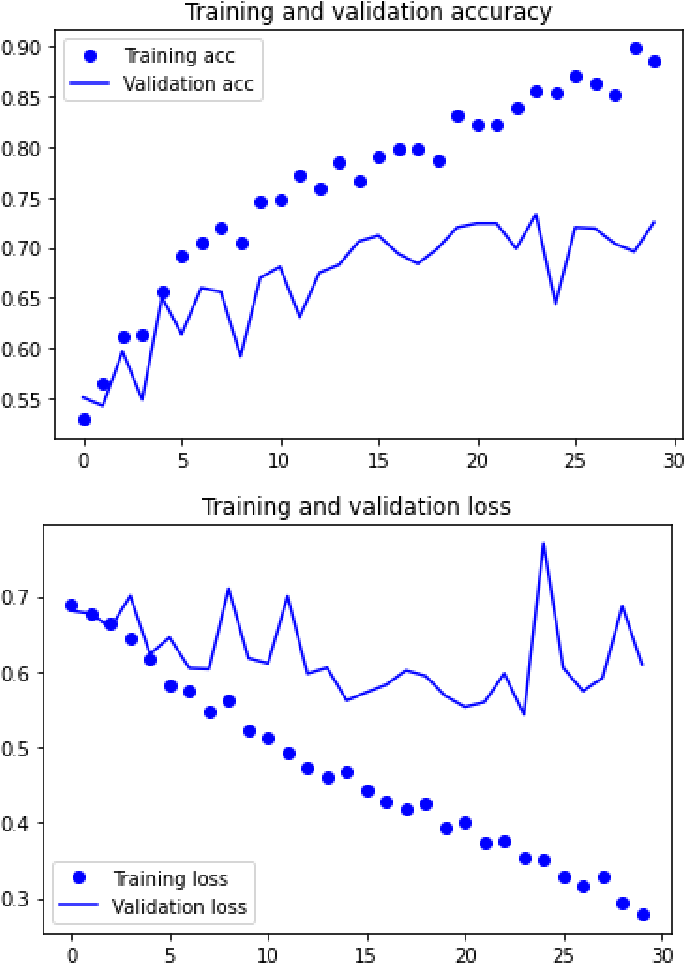
from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator( rescale=1 ./ 255 ,



rotation\_range=40 , model.save('cats\_and\_dogs\_small\_1.h5') width\_shift\_range=0.2 ,

height\_shift\_range=0.2 , import matplotlib.pyplot as plt shear\_range=0.2 , acc = history.history['acc'] val\_acc = history.history['val\_acc']

 zoom\_range=0.2 , loss = history.history['loss'] val\_loss = horizontal\_flip=True , history.history['val\_loss'] fill\_mode='nearest' ) epochs = range(len(acc))

# Note that the validation data should not be augmented! plt.plot(epochs, acc, 'bo', label='Training acc') plt.plot(epochs, test\_datagen = ImageDataGenerator(rescale=1./ 255 ) val\_acc, 'b', label='Validation acc') plt.title('Training and validation accuracy') plt.legend()

train\_generator = train\_datagen.flow\_from\_directory( plt.figure()

# This is the target directory 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\_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

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') 

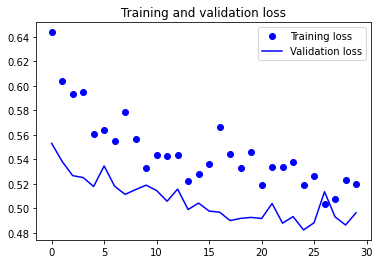
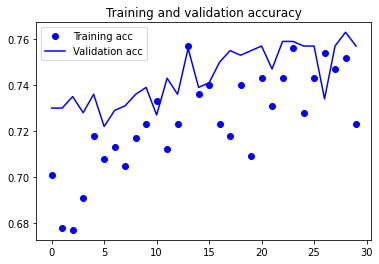
import matplotlib.pyplot as plt acc = history.history['acc'] val\_acc = history.history['val\_acc'] loss = history.history['loss'] val\_loss =

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

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



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']

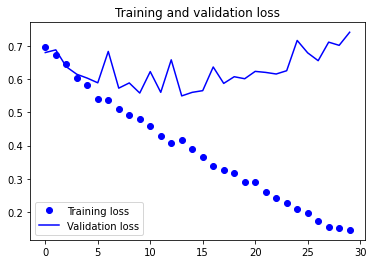
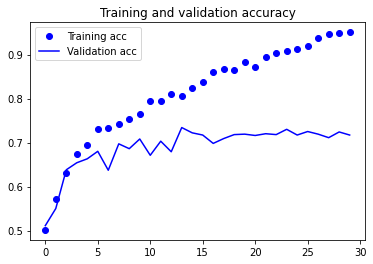
epochs = range(len(acc))

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()



# 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

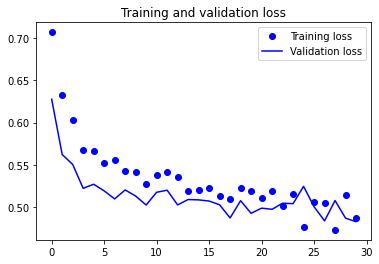
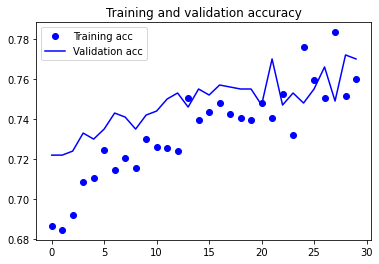
model.save('cats\_and\_dogs\_small\_4.h5') 

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

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 =

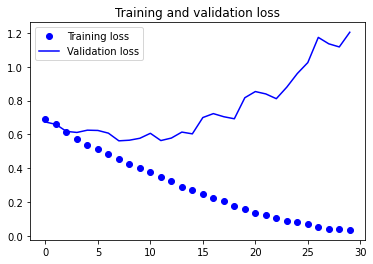
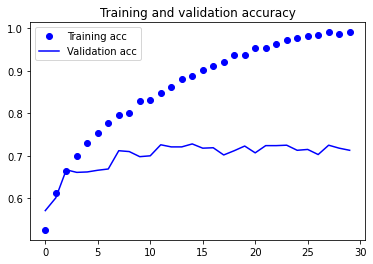
history.history['val\_loss']

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) poc /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

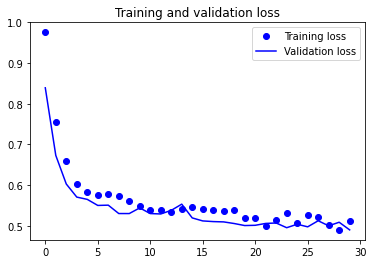
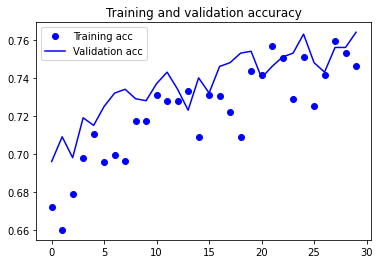
history['val loss'] val\_loss = history.history[ val\_loss ]

 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-](https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5)

[applications/vgg16/vgg16\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5](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

|  |  |
| --- | --- |
| 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') |  |

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 '

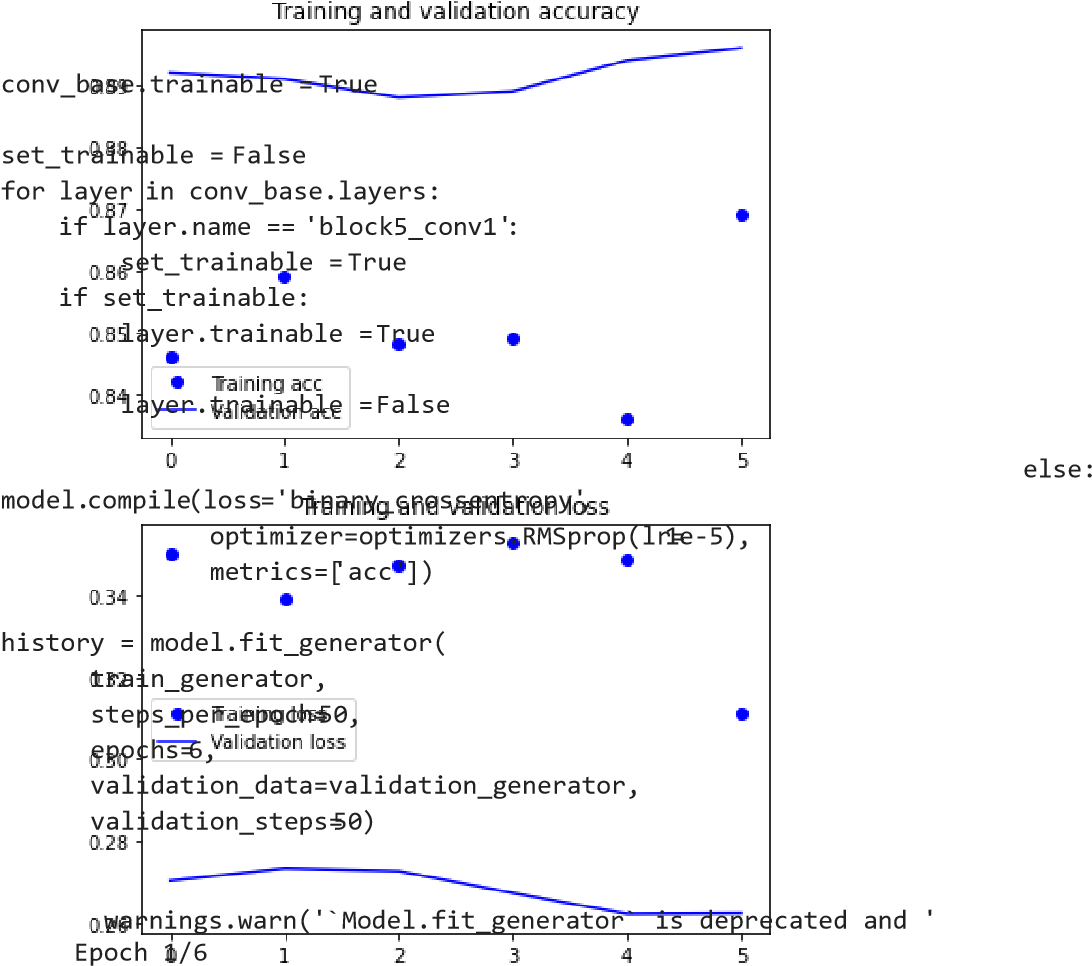
Epoch 1/6

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

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()

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit\_generator` is deprecated and will be rem

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 =

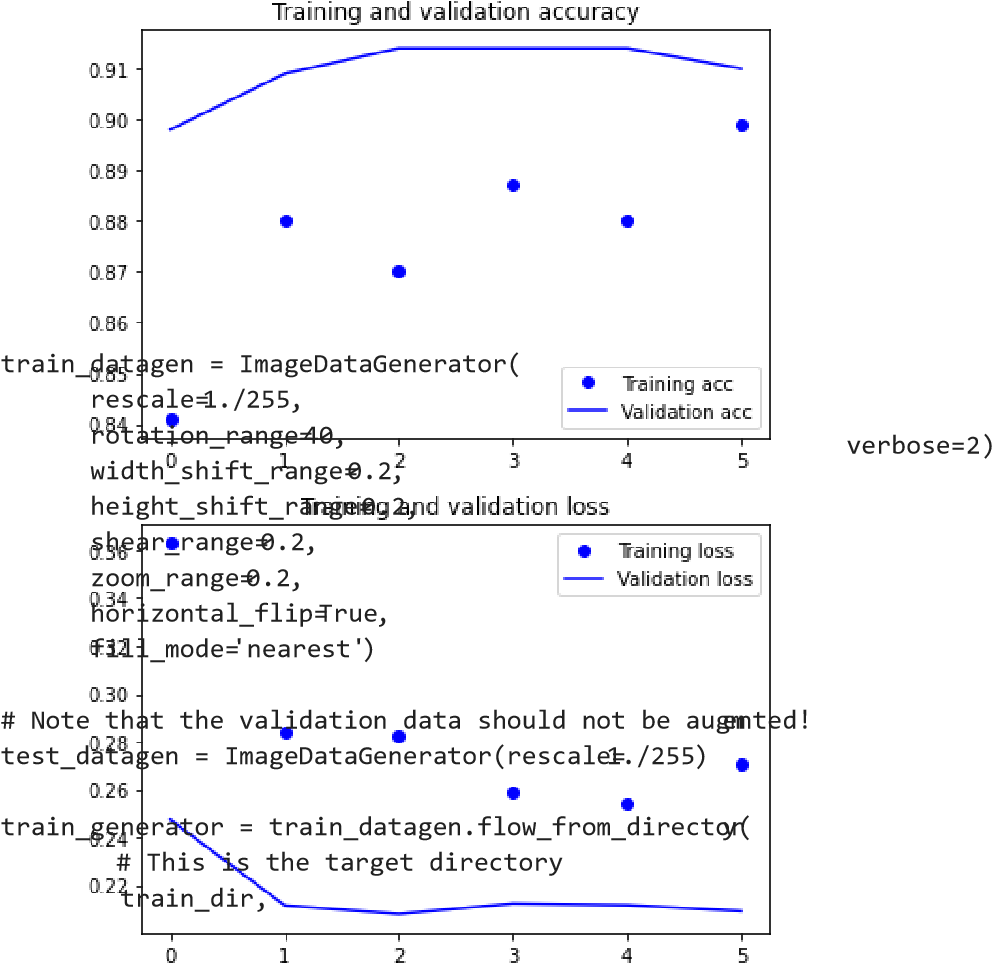
history.history['val\_loss']

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()

 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,

# All images will be resized to 150x150 target\_size=(150 , 150 ) ,

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']

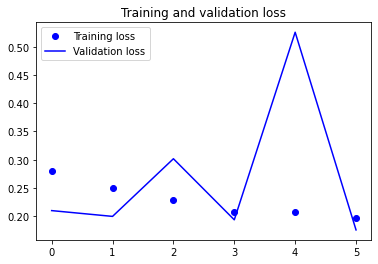
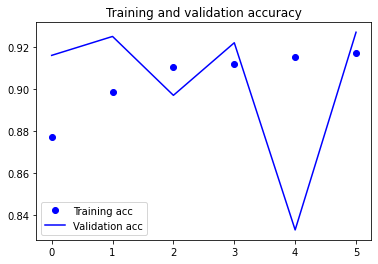
epochs = range(len(acc))

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 =

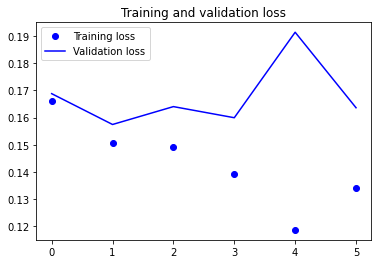
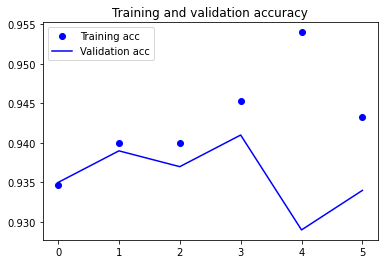


history.history['val\_loss']

h (l ( ))

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']

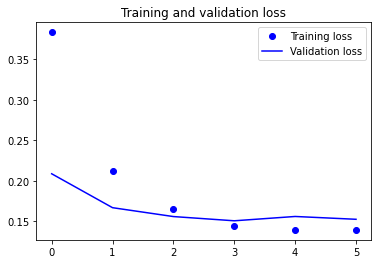
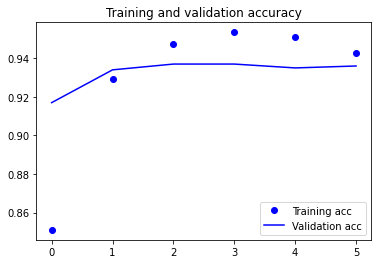
epochs = range(len(acc))

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 =



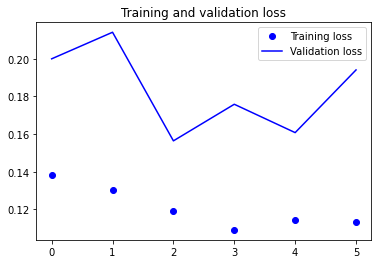
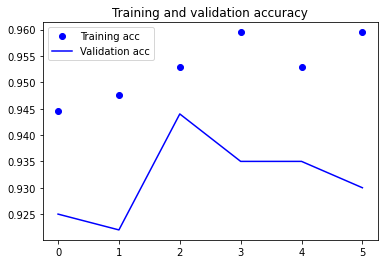
history.history['val\_loss']

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()



|  |  |
| --- | --- |
| The classification model's accuracy and validation loss were assessed using various techniques at different stages of training. When the dataset size increased from 1000 to 1500, the validation accuracy remained unchanged. However, after applying data augmentation and regularization, there was a noticeable 4% increase in accuracy. The use of a pretrained network led to a significant improvement in validation accuracy (25%), which further increased to 28% with data augmentation and regularization. The same trend was observed when the dataset size was increased to 2000, where data augmentation and regularization proved to be more effective than simply expanding the dataset. The most effective method for classification models was found to be using pretrained networks, resulting in a significant increase in validation accuracy (29%). Moreover, this improvement further increased to 31% with the use of data augmentation and regularization. Additionally, the validation loss decreased with each applied technique. Overall, the results suggest that data augmentation and regularization are more effective for training a network, but pretrained networks are most effective for classification models. | |
| The classification model's accuracy and validation loss were assessed using various techniques at different stages of training. When the dataset size increased from 1000 to 1500, the validation accuracy remained unchanged. However, after applying data augmentation and regularization, there was a noticeable 4% increase in accuracy. The use of a pretrained network led to a significant improvement in validation accuracy (25%), which further increased to 28% with data augmentation and regularization. The same trend was observed when the dataset size was increased to 2000, where data augmentation and regularization proved to be more effective than simply expanding the dataset. The most effective method for classification models was found to be using pretrained networks, resulting in a significant increase in validation accuracy (29%). Moreover, this improvement further increased to 31% with the use of data augmentation and regularization. Additionally, the validation loss decreased with each applied technique. Overall, the results suggest that data augmentation and regularization are more effective for training a network, but pretrained networks are most effective for classification models. | t |
| The classification model's accuracy and validation loss were assessed using various techniques at different stages of training. When the dataset size increased from 1000 to 1500, the validation accuracy remained unchanged. However, after applying data augmentation and regularization, there was a noticeable 4% increase in accuracy. The use of a pretrained network led to a significant improvement in validation accuracy (25%), which further increased to 28% with data augmentation and regularization. The same trend was observed when the dataset size was increased to 2000, where data augmentation and regularization proved to be more effective than simply expanding the dataset. The most effective method for classification models was found to be using pre-trained networks, resulting in a significant increase in validation accuracy (29%). Moreover, this improvement further increased to 31% with the use of data augmentation and regularization. Additionally, the validation loss decreased with each applied technique. Overall, the results suggest that data augmentation and regularization are more effective for training a network, but pre-trained networks are most effective for classification models. | |
| The classification model's accuracy and validation loss was assessed using various techniques at different stages of training. When the dataset size increased from 1000 to 1500, the validation accuracy remained unchanged. However, after applying data augmentation and regularization, there was a noticeable 4% increase in accuracy. The use of a pre-trained network led to a significant improvement in validation accuracy (25%), which further increased to 28% with data augmentation and regularization. The same trend was observed when the dataset size was increased to 2000, where data augmentation and regularization proved to be more effective than simply expanding the dataset. The most effective method for classification models was found to be using pre-trained networks resulting in a significant increase in validation accuracy (29%). Moreover, this improvement further increased to 31% with the use of data augmentation and regularization. Additionally, the validation loss decreased with each applied technique. Overall, the results suggest that data augmentation and regularization are more effective for training a network, but pre-trained networks are most effective for classification models. | , |
| The classification model's accuracy and validation loss was assessed using various techniques at different stages of training. When the dataset size increased from 1000 to 1500, the validation accuracy remained unchanged. However, after applying data augmentation and regularization, there was a noticeable 4% increase in accuracy. The use of a pre-trained network led to a significant improvement in validation accuracy (25%), which further increased to 28% with data augmentation and regularization. The same trend was observed when the dataset size was increased to 2000, where data augmentation and regularization proved to be more effective than simply expanding the dataset. The most effective method for classification models was found to be using pre-trained networks, resulting in a significant increase in validation accuracy (29%). Moreover, this improvement further increased to 31% with the use of data augmentation and regularization. Additionally, the validation loss decreased with each applied technique. Overall, the results suggest that data augmentation and regularization are more effective for training a network, but pre-trained networks are most effective for classification models. | |
| The classification model's accuracy and validation loss was assessed using various techniques at different stages of training. When the dataset size increased from 1000 to 1500, the validation accuracy remained unchanged. However, after applying data augmentation and regularization, there was a noticeable 4% increase in accuracy. The use of a pre-trained network led to a significant improvement in validation accuracy (25%), which further increased to 28% with data augmentation and regularization. The same trend was observed when the dataset size was increased to 2000, where data augmentation and regularization proved to be more effective than simply expanding the dataset. The most effective method for classification models was found to be using pre-trained networks, resulting in a significant increase in validation accuracy (29%). Moreover, this improvement further increased to 31% with the use of data augmentation and regularization. Additionally, the validation loss decreased with each applied technique. Overall, the results suggest that data augmentation and regularization are more effective for training a network, but pre-trained networks are most effective for classification models. |  |

