

```
import keras
keras.__version__
```

```
'2.4.3' '2.4.3'
```

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

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

```
import os
import shutil
```

```
base_dir = '/content/gdrive/MyDrive/ML Assignment 2/cats_and_dogs_3'
!ls '/content/gdrive/MyDrive/ML Assignment 2/cats_and_dogs_3'
```

```
test train validation
```

## ▼ Read Cats and Dogs

```
# 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: 500
total training dog images: 500
total validation cat images: 500
total validation dog images: 500
total test cat images: 500
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\_1"

Layer (type)	Output Shape	Param #
=====		
conv2d_4 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_4 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_5 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_5 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_6 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_6 (MaxPooling2D)	(None, 17, 17, 128)	0

conv2d_7 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_7 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_2 (Dense)	(None, 512)	3211776
dense_3 (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 1000 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=10,
    validation_data=validation_generator,
    validation_steps=50)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844:
  warnings.warn("`Model.fit_generator` is deprecated and "
```

```
Epoch 1/10
50/50 [=====] - 49s 973ms/step - loss: 0.6970 - acc: 0.5034 - \
Epoch 2/10
50/50 [=====] - 48s 970ms/step - loss: 0.6843 - acc: 0.5704 - \
Epoch 3/10
50/50 [=====] - 48s 966ms/step - loss: 0.6516 - acc: 0.6100 - \
Epoch 4/10
50/50 [=====] - 48s 971ms/step - loss: 0.6187 - acc: 0.6818 - \
Epoch 5/10
50/50 [=====] - 48s 970ms/step - loss: 0.5903 - acc: 0.6764 - \
Epoch 6/10
50/50 [=====] - 48s 967ms/step - loss: 0.5661 - acc: 0.7122 - \
Epoch 7/10
50/50 [=====] - 48s 969ms/step - loss: 0.5455 - acc: 0.7072 - \
Epoch 8/10
50/50 [=====] - 48s 970ms/step - loss: 0.5220 - acc: 0.7607 - \
Epoch 9/10
50/50 [=====] - 48s 971ms/step - loss: 0.4815 - acc: 0.7774 - \
Epoch 10/10
50/50 [=====] - 48s 971ms/step - loss: 0.4568 - acc: 0.7741 - \
```

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

```
import matplotlib.pyplot as plt
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
epochs = range(len(acc))
```

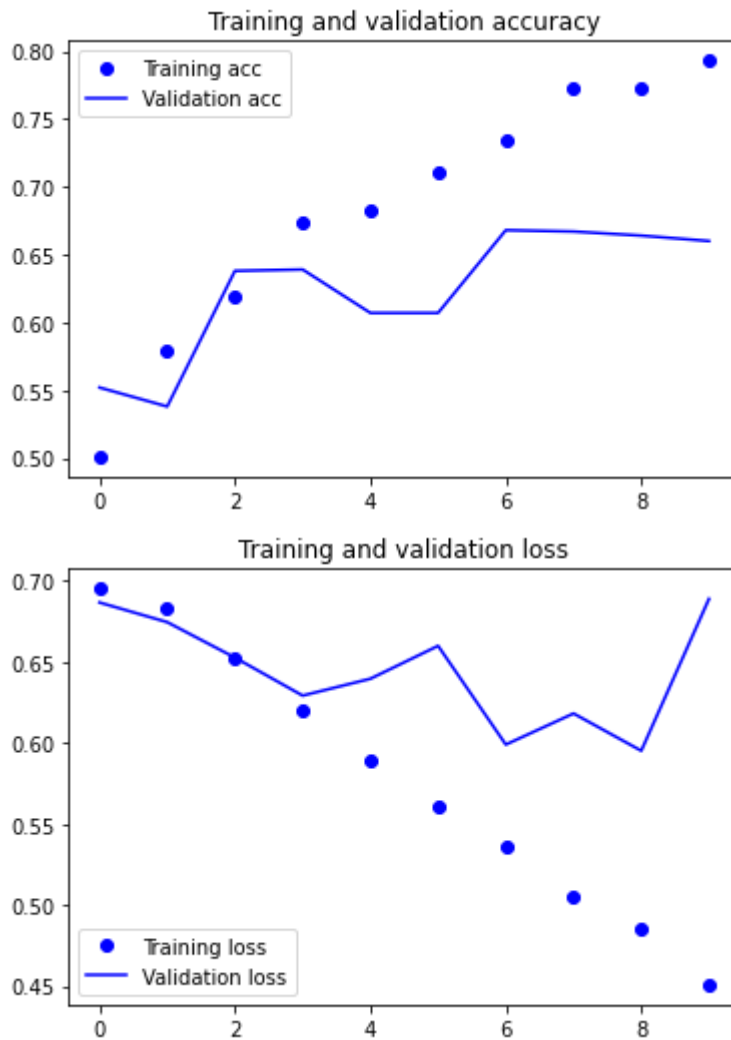
```
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
```

```
plt.figure()
```

```
plt.plot(epochs, loss, 'bo', label='Training loss')
```

```
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



## ▼ Data Augmentation

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

```
# This is module with image preprocessing utilities
from keras.preprocessing import image
```

```
from keras.preprocessing import image

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

# We pick one image to "augment"
img_path = fnames[3]

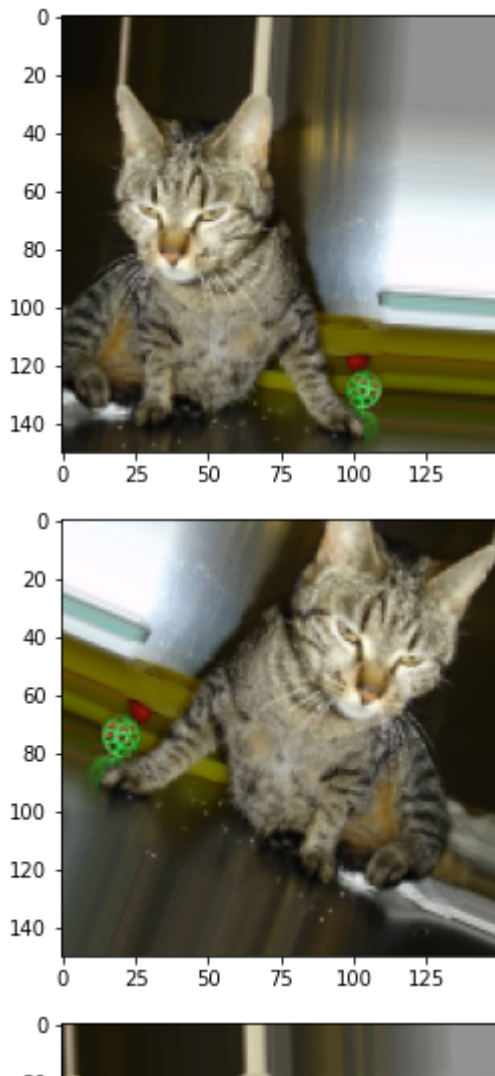
# Read the image and resize it
img = image.load_img(img_path, target_size=(150, 150))

# Convert it to a Numpy array with shape (150, 150, 3)
x = image.img_to_array(img)

# Reshape it to (1, 150, 150, 3)
x = x.reshape((1,) + x.shape)

# The .flow() command below generates batches of randomly transformed images.
# It will loop indefinitely, so we need to `break` the loop at some point!
i = 0
for batch in datagen.flow(x, batch_size=1):
    plt.figure(i)
    imgplot = plt.imshow(image.array_to_img(batch[0]))
    i += 1
    if i % 4 == 0:
        break

plt.show()
```



```

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

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

```

100

```

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=10
)

```

```
rotation_range=40,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,)
```

# 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=32,
    # 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=32,
    class_mode='binary')
```

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch=50,
    epochs=10,
    validation_data=validation_generator,
    validation_steps=50)
```

Found 1000 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:

warnings.warn("`Model.fit\_generator` is deprecated and "

Epoch 1/10

32/50 [=====>.....] - ETA: 22s - loss: 0.6933 - acc: 0.5200WARNING:te

WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that you

50/50 [=====] - 52s 1s/step - loss: 0.6933 - acc: 0.5200 - val\_

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

```
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()
```

