강아지 vs 고양이 분류하기(1) - 캐글노트북

학습 내용

- 개와 고양이의 이미지를 준비합니다.
- 신경망을 구축 후, 학습을 수행합니다.
- 모델 학습 후, 이를 저장하는 것을 대해 알아봅니다.

환경

- 캐글 노트북

데이터(강아지 vs 고양이)

- url : https://www.kaggle.com/c/dogs-vs-cats/data (https://www.kaggle.com/c/dogs-vs-cats/data)
- test1.zip : 271.15MB
- train.zip : 543.16MB
- sampleSubmission.csv

목차

01. 데이터 준비하기

목차로 이동하기







- url : https://s3.amazonaws.com/book.keras.io/img/ch5/cats_vs_dogs_samples.jpg (https://s3.amazonaws.com/book.keras.io/img/ch5/cats_vs_dogs_samples.jpg)

- 25,000개의 강아지와 고양이 이미지
- 클래스마다 12,500개를 담고 있다. 압축(543MB크기)
- 클래스마다 1,000개의 샘플로 이루어진 훈련 세트
- 클래스마다 500개의 샘플로 이루어진 검증 세트
- 클래스마다 500개의 샘플로 이루어진 테스트 세트

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/dogs-vs-cats/test1.zip
/kaggle/input/dogs-vs-cats/train.zip
/kaggle/input/dogs-vs-cats/sampleSubmission.csv
```

압축을 풀고, 데이터 준비

In [2]:

```
os.getcwd()
```

Out[2]:

```
'/kaggle/working'
```

In [3]:

```
# 데이터 셋 복사 및 확인
!cp -r '/kaggle/input/dogs-vs-cats/'  '/kaggle/working/'
!ls -ls '/kaggle/working/'
```

```
total 36
32 ---------- 1 root root 32624 Oct 27 02:14 __notebook__.ipynb
 4 drwxr-xr-x 2 root root  4096 Oct 27 02:14 dogs-vs-cats
```

In [4]:

```
!rm -rf '/kaggle/working/datasets/'
!unzip '/kaggle/working/dogs-vs-cats/test1.zip' -d '/kaggle/working/datasets/'
!unzip '/kaggle/working/dogs-vs-cats/train.zip' -d '/kaggle/working/datasets/'
```

```
Archive:  /kaggle/working/dogs-vs-cats/test1.zip
   creating: /kaggle/working/datasets/test1/
  inflating: /kaggle/working/datasets/test1/1.jpg
  inflating: /kaggle/working/datasets/test1/10.jpg
  inflating: /kaggle/working/datasets/test1/100.jpg
  inflating: /kaggle/working/datasets/test1/1000.jpg
  inflating: /kaggle/working/datasets/test1/10000.jpg
  inflating: /kaggle/working/datasets/test1/10001.jpg
  inflating: /kaggle/working/datasets/test1/10002.jpg
  inflating: /kaggle/working/datasets/test1/10003.jpg
  inflating: /kaggle/working/datasets/test1/10004.jpg
  inflating: /kaggle/working/datasets/test1/10005.jpg
  inflating: /kaggle/working/datasets/test1/10006.jpg
  inflating: /kaggle/working/datasets/test1/10007.jpg
  inflating: /kaggle/working/datasets/test1/10008.jpg
  inflating: /kaggle/working/datasets/test1/10009.jpg
  inflating: /kaggle/working/datasets/test1/1001.jpg
  inflating: /kaggle/working/datasets/test1/10010.jpg
  inflating: /kaggle/working/datasets/test1/10011.jpg
  inflating: /kaggle/working/datasets/test1/10012.jpg
```

In [5]:

```
!ls -al '/kaggle/working/datasets/train' | head -5
!ls -l '/kaggle/working/datasets/train' | grep ^- | wc -l
!ls -al '/kaggle/working/datasets/test1' | head -5
!ls -l '/kaggle/working/datasets/test1' | grep ^- | wc -l
```

```
total 609268
drwxr-xr-x 2 root root 774144 Sep 20  2013 .
drwxr-xr-x 4 root root   4096 Oct 27 02:14 ..
-rw-r--r-- 1 root root  12414 Sep 20  2013 cat.0.jpg
-rw-r--r-- 1 root root  16880 Sep 20  2013 cat.1.jpg
ls: write error: Broken pipe
25000
total 304268
drwxr-xr-x 2 root root 274432 Sep 20  2013 .
drwxr-xr-x 4 root root   4096 Oct 27 02:14 ..
-rw-r--r-- 1 root root  24178 Sep 20  2013 1.jpg
-rw-r--r-- 1 root root  22194 Sep 20  2013 10.jpg
ls: write error: Broken pipe
12500
```

In [6]:

```
# 원본 데이터셋을 압축 해제한 디렉터리 경로(학습)
ori_dataset_dir = '/kaggle/working/datasets/train'

# 학습을 작게 하기 위한 소규모 데이터셋을 저장할 디렉터리
base_dir = '/kaggle/working/datasets/cats_and_dogs_small'
```

In [7]:

```
# 반복실행을 위해 디렉터리 삭제 및 생성
if os.path.exists(base_dir):
    shutil.rmtree(base_dir)
os.mkdir(base_dir)
```

In [8]:

```
# 훈련, 검증, 테스트 분할을 위한 디렉터리
train_dir = os.path.join(base_dir, 'train')
os.mkdir(train_dir)

val_dir = os.path.join(base_dir, 'validation')
os.mkdir(val_dir)

test_dir = os.path.join(base_dir, 'test')
os.mkdir(test_dir)
```

In [9]:

```
# 훈련용 고양이 사진 디렉터리
train_cats_dir = os.path.join(train_dir, 'cats')
os.mkdir(train_cats_dir)

# 훈련용 강아지 사진 디렉터리
train_dogs_dir = os.path.join(train_dir, 'dogs')
os.mkdir(train_dogs_dir)

# 검증용 고양이 사진 디렉터리
val_cats_dir = os.path.join(val_dir, 'cats')
os.mkdir(val_cats_dir)

# 검증용 강아지 사진 디렉터리
val_dogs_dir = os.path.join(val_dir, 'dogs')
os.mkdir(val_dogs_dir)

# 테스트용 고양이 사진 디렉터리
test_cats_dir = os.path.join(test_dir, 'cats')
os.mkdir(test_cats_dir)

# 테스트용 강아지 사진 디렉터리
test_dogs_dir = os.path.join(test_dir, 'dogs')
os.mkdir(test_dogs_dir)
```

In [10]:

```
!ls -al './datasets/cats_and_dogs_small/train' | head -5
!ls -al './datasets/cats_and_dogs_small/validation/' | head -5
!ls -al './datasets/cats_and_dogs_small/test/' | head -5
```

```
total 16
drwxr-xr-x 4 root root 4096 Oct 27 02:14 .
drwxr-xr-x 5 root root 4096 Oct 27 02:14 ..
drwxr-xr-x 2 root root 4096 Oct 27 02:14 cats
drwxr-xr-x 2 root root 4096 Oct 27 02:14 dogs
total 16
drwxr-xr-x 4 root root 4096 Oct 27 02:14 .
drwxr-xr-x 5 root root 4096 Oct 27 02:14 ..
drwxr-xr-x 2 root root 4096 Oct 27 02:14 cats
drwxr-xr-x 2 root root 4096 Oct 27 02:14 dogs
total 16
drwxr-xr-x 4 root root 4096 Oct 27 02:14 .
drwxr-xr-x 5 root root 4096 Oct 27 02:14 ..
drwxr-xr-x 2 root root 4096 Oct 27 02:14 cats
drwxr-xr-x 2 root root 4096 Oct 27 02:14 dogs
```

In [11]:

```
datasize = 2000

# 처음 2,000개의 고양이 이미지를 train_cats_dir에 복사합니다
fnames = ['cat.{}.jpg'.format(i) for i in range(datasize)]
for fname in fnames:
    src = os.path.join(ori_dataset_dir, fname)
    dst = os.path.join(train_cats_dir, fname)
    shutil.copyfile(src, dst)

# 다음 500개 고양이 이미지를 validation_cats_dir에 복사합니다
fnames = ['cat.{}.jpg'.format(i) for i in range(datasize, datasize+500)]
for fname in fnames:
    src = os.path.join(ori_dataset_dir, fname)
    dst = os.path.join(val_cats_dir, fname)
    shutil.copyfile(src, dst)

# 다음 500개 고양이 이미지를 test_cats_dir에 복사합니다
fnames = ['cat.{}.jpg'.format(i) for i in range(datasize+500, datasize+1000)]
for fname in fnames:
    src = os.path.join(ori_dataset_dir, fname)
    dst = os.path.join(test_cats_dir, fname)
    shutil.copyfile(src, dst)

# 처음 2,000개의 강아지 이미지를 train_dogs_dir에 복사합니다
fnames = ['dog.{}.jpg'.format(i) for i in range(datasize)]
for fname in fnames:
    src = os.path.join(ori_dataset_dir, fname)
    dst = os.path.join(train_dogs_dir, fname)
    shutil.copyfile(src, dst)

# 다음 500개 강아지 이미지를 validation_dogs_dir에 복사합니다
fnames = ['dog.{}.jpg'.format(i) for i in range(datasize, datasize+500)]
for fname in fnames:
    src = os.path.join(ori_dataset_dir, fname)
    dst = os.path.join(val_dogs_dir, fname)
    shutil.copyfile(src, dst)

# 다음 500개 강아지 이미지를 test_dogs_dir에 복사합니다
fnames = ['dog.{}.jpg'.format(i) for i in range(datasize+500, datasize+1000)]
for fname in fnames:
    src = os.path.join(ori_dataset_dir, fname)
    dst = os.path.join(test_dogs_dir, fname)
    shutil.copyfile(src, dst)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_23/3611519122.py in <module>
      6     src = os.path.join(ori_dataset_dir, fname)
      7     dst = os.path.join(train_cats_dir, fname)
----> 8     shutil.copyfile(src, dst)
      9
     10 # 다음 500개 고양이 이미지를 validation_cats_dir에 복사합니다

NameError: name 'shutil' is not defined
```

In [ ]:

```
print('훈련용 고양이 이미지 전체 개수:', len(os.listdir(train_cats_dir)))
print('훈련용 강아지 이미지 전체 개수:', len(os.listdir(train_dogs_dir)))

print('검증용 고양이 이미지 전체 개수:', len(os.listdir(val_cats_dir)))
print('검증용 강아지 이미지 전체 개수:', len(os.listdir(val_dogs_dir)))

print('테스트용 고양이 이미지 전체 개수:', len(os.listdir(test_cats_dir)))
print('테스트용 강아지 이미지 전체 개수:', len(os.listdir(test_dogs_dir)))
```

02. 데이터 전처리

목차로 이동하기

- 사진 파일을 읽기
- JPEG 콘텐츠를 RGB픽셀로 디코딩
- 부동 소수 타입의 텐서로 변환
- 픽셀 값(0~255)의 스케일을 [0,1]사이로 조정

In [ ]:

```
from keras.preprocessing.image import ImageDataGenerator

# 모든 이미지를 1/255로 스케일을 조정합니다
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,  # 타깃 디렉터리
        target_size=(150, 150),  # 모든 이미지를 150 × 150 크기로 변경
        batch_size=20,
        # binary_crossentropy 손실을 사용하기 때문에 이진 레이블이 필요
        class_mode='binary')

val_generator = test_datagen.flow_from_directory(
        val_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
```

In [ ]:

```
for data_batch, labels_batch in train_generator:
    print('배치 데이터 크기:', data_batch.shape)
    print('배치 레이블 크기:', labels_batch.shape)
    break
```

03. 신경망 구성하기

목차로 이동하기

- Conv2D (3x3) filter:32 - Maxpooling2D (2x2) stride 1
- Conv2D (3x3) filter:64 - Maxpooling2D (2x2) stride 1
- Conv2D (3x3) filter:128 - Maxpooling2D (2x2) stride 1
- Conv2D (3x3) filter:128 - Maxpooling2D (2x2) stride 1

In [ ]:

```
from tensorflow.keras import layers
from tensorflow.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.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

- 150 x 150 크기에서 7 x 7 크기의 특성 맵으로 줄어든다.

최적화 알고리즘, 손실 함수 선택

In [ ]:

```
from tensorflow.keras import optimizers

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

제너레이터를 사용한 데이터의 모델 훈련

- fit_generator 메서드는 fit 메서드와 동일하고 데이터 제너레이터를 사용 가능.
  - 데이터가 끝없이 생성되기에 케라스 모델에 하나의 에포크를 정의하기 위해 제너레이터로부터 얼마나 많은 샘플을 뽑을지 알려 주어야 한다.(steps_per_epoch 이를 설정)
  - validation_data 매개변수를 사용 가능.
  - validation_steps에서 얼마나 많은 배치를 추출할지 평가할지 validation_steps 매개변수에 지정한다.

In [ ]:

```
## 경로에 이미지 데이터의 개수
num_cats_tr = len(os.listdir(train_cats_dir))
num_dogs_tr = len(os.listdir(train_dogs_dir))

num_cats_val = len(os.listdir(val_cats_dir))
num_dogs_val = len(os.listdir(val_dogs_dir))

total_train = num_cats_tr + num_dogs_tr
total_val = num_cats_val + num_dogs_val

print("학습용 데이터 : ", total_train)
print("검증용 데이터 : ", total_val)

batch_size = 20
epochs = 30
```

04. 모델 학습

목차로 이동하기

In [ ]:

```
%%time

history = model.fit(   # model.fit_gerator -> model.fit() 으로 최근 추가됨
      train_generator,
      steps_per_epoch = total_train // batch_size,
      epochs=30,
      validation_data=val_generator,
      validation_steps = total_val // batch_size )
```

05. 훈련 후, 모델 저장

목차로 이동하기

In [ ]:

```
model.save('cats_and_dogs_small_4000.h5')
```

In [ ]:

```
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()
```