

5. EC2와 S3를 이용한 AI 개발

2강. 클라우드 기반의 CNN 구현

학습목표

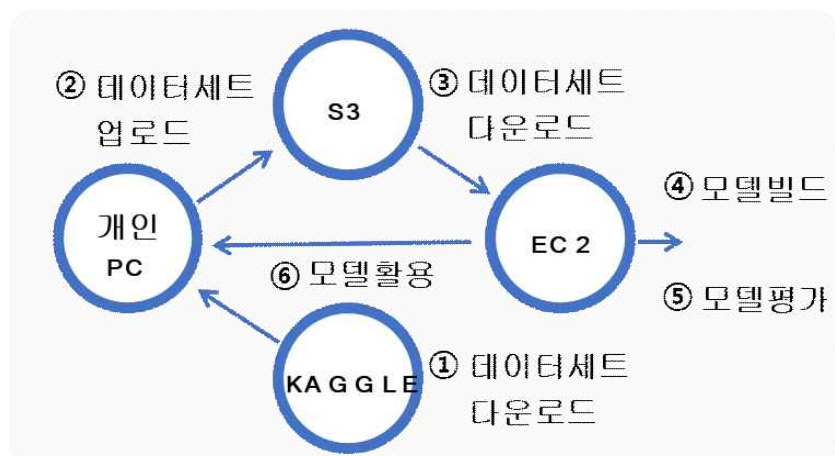
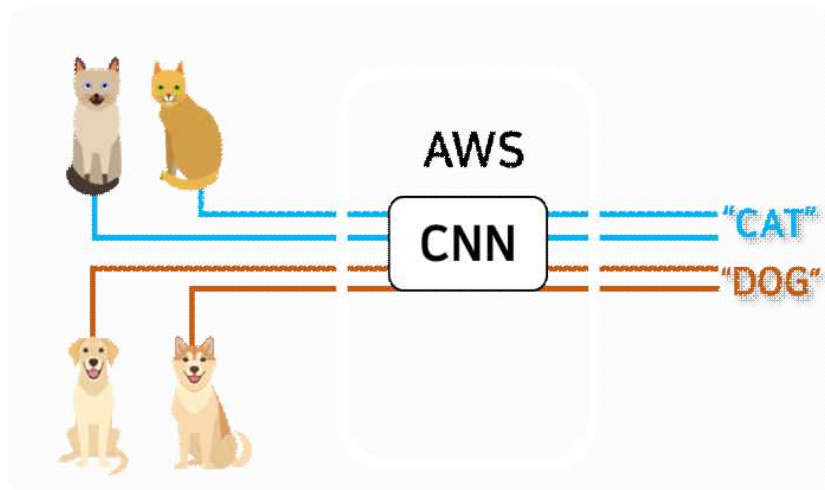
- EC2, S3 를 이용한 CNN 신경망 구축 단계 및 방법에 대해 설명할 수 있다.

학습내용

- 데이터세트 준비하기
- AWS를 이용한 CNN 모델 구축 단계
- CNN 학습모델 빌드 및 평가

■ 세상을 잇(IT)다!

- 고양이 vs 개를 분류해보자



1. 데이터세트 준비하기

- ① Kaggle에서 오픈 데이터 세트 다운로드
- ② AWS S3에 데이터 업로드



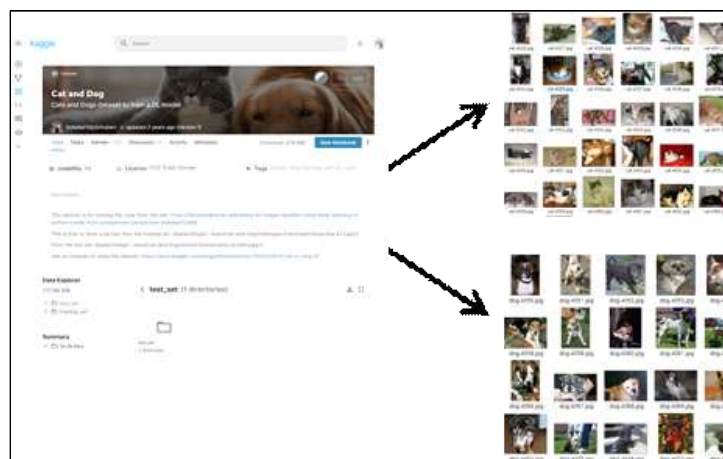
생각해보기

- 캐글(Kaggle)
 - ✓ 2010년에 설립된 예측 모델 및 분석 대회 플랫폼으로 딥러닝 및 머신러닝에 필요한 데이터세트를 무료로 제공



- 데이터 분석/예측
- 빅데이터 경험
- 경연
- 데이터 과학자들과 소통

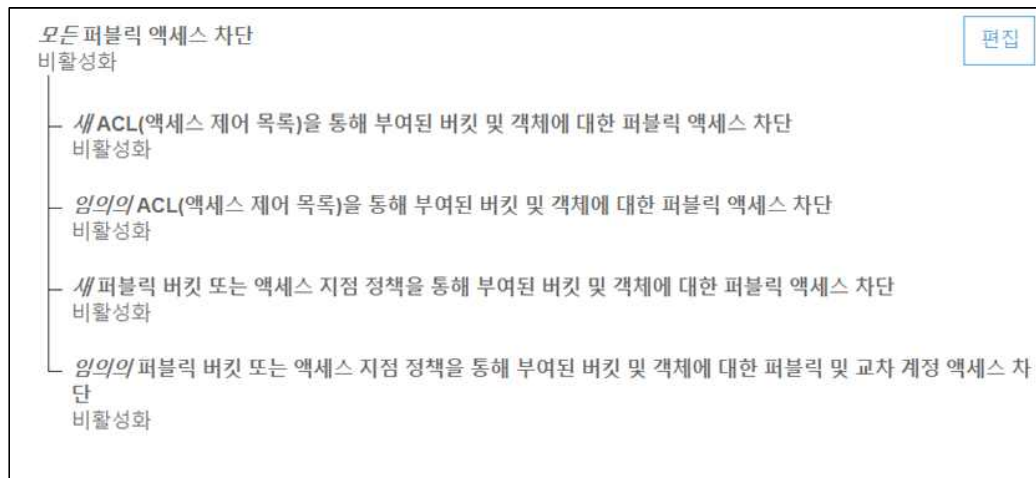
- Cat and Dog
 - ✓ Kaggle의 cat and dog에는 다양한 사이즈의 고양이와 개 이미지 파일이 포함되어 있음



[출처] Kaggle(<https://www.kaggle.com/tongpython/cat-and-dog>)

2. AWS를 이용한 CNN 모델 구축 단계

- ① AWS 버킷생성하기
 - ② AWS S3 버킷에 파일 업로드
 - ③ 권한부여
 - AWS 버킷 클릭
 - 모든 퍼블릭 액세스 차단 비활성화 활성화하기
 - 버킷 정책 입력
- 모든 퍼블릭 액세스 차단 비활성화
 - ✓ S3 접속 → [버킷이름] → [권한] → [퍼블릭 액세스 차단]



- S3 버킷 정책
 - ✓ S3 접속 → [버킷이름] → [권한] → [버킷정책]



```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AddCannedAcl",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<계정ID Num.>:root"
      },
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::<bucket name>/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "public-read"
        }
      }
    }
  ]
}

```

↑
내 계정에서 확인

- EC2 데이터 다운로드 단계
 - ① 주피터 실행
 - ② S3 데이터(.zip) 다운로드
 - ③ 다운로드 된 데이터 압축풀기

- 데이터(파일) 다운로드

```
import boto3
import botocore

bucket_name = 'bucket name'
# ex) bdu01
#name in s3
in_file = 'filename+extension'
#ex) dog-and-cat.zip
out_file = in_file

s3 = boto3.resource('s3')
try :
s3.Bucket(bucket_name).download_file(in_file, out_file)
except botocore.exceptions.ClientError as e:
    if e.response['Error']['Code'] == '404':
        print('해당 파일이 없습니다.')
    else:
        raise
```

- 데이터 압축풀기

```
import zipfile

try:
    with zipfile.ZipFile("filename+extension") as zf:
        zf.extractall()
        print("uncompress success")

except:
    print("uncompress fail")
```

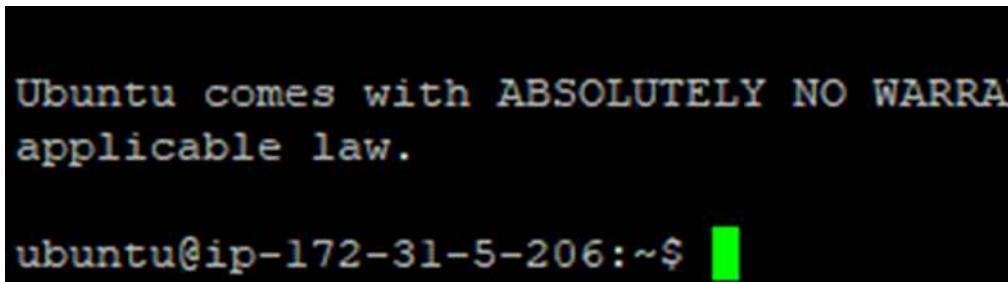
3. CNN 학습모델 빌드 및 평가

- CNN 학습모델 구축 단계

- ① 필요라이브러리 추가
- ② 주피터 실행
- ③ 모델 빌드
- ④ 모델 평가

- 필요 라이브러리 추가

- ✓ EC2 인스턴스 명령창에 명령어 입력



```

Ubuntu comes with ABSOLUTELY NO WARRANTY.
Ubuntu is provided AS IS, without any warranty, to the extent
permitted by applicable law.

ubuntu@ip-172-31-5-206:~$
  
```

- sudo pip3 install tensorflow==1.15
- sudo pip3 install keras

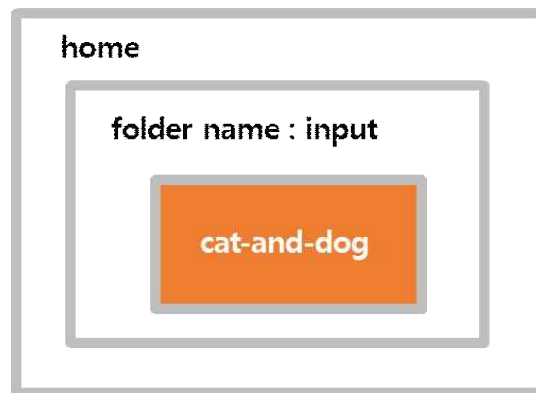
- 모델 빌드 단계

- ① 데이터 로드
- ② 이미지 전처리
- ③ 데이터 분할
- ④ 모델 설정
- ⑤ 학습진행
- ⑥ 학습 결과 시각화

- 데이터 로드

```

import os
rootPath = './input/cat-and-dog'
  
```



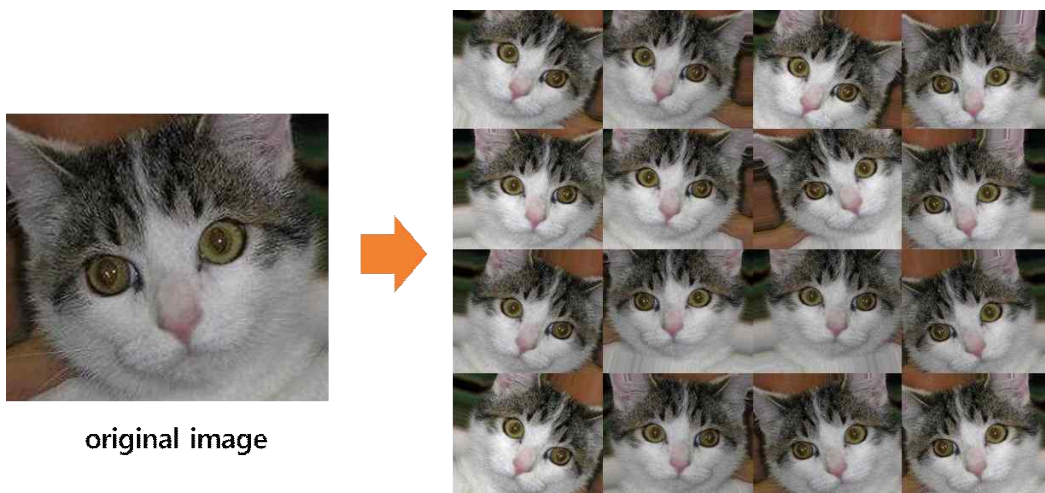
- 이미지 전처리
 - ✓ 모델강화(학습량 증가)를 위해 데이터를 회전, 좌우반전, 이동, 역전등으로 이미지를 바꿈

```

from tensorflow.keras.preprocessing.image import
ImageDataGenerator

imageGenerator = ImageDataGenerator(rescale=1./255,
                                     rotation_range=20,
                                     width_shift_range=0.1,
                                     height_shift_range=0.1,

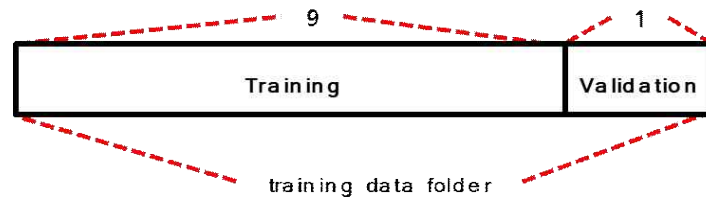
                                     brightness_range=[.2,.2],
                                     horizontal_flip=True,
                                     validation_split=.1)
  
```



- 데이터 분할

```
trainGen =
imageGenerator.flow_from_directory(os.path.join(rootPath,'training_set'),
                                   target_size=(64,64),
                                   subset='training')

validationGen =
imageGenerator.flow_from_directory(os.path.join(rootPath,'training_set'),
                                   target_size=(64,64),
                                   subset='validation')
```



- 모델 설정 - 1

```
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers

model = Sequential()
```

- 모델 설정 - 2

```
model.add(layers.InputLayer(input_shape=(64,64,3)))
model.add(layers.Conv2D(16,(3,3),(1,1),'same',activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(rate=0.3))

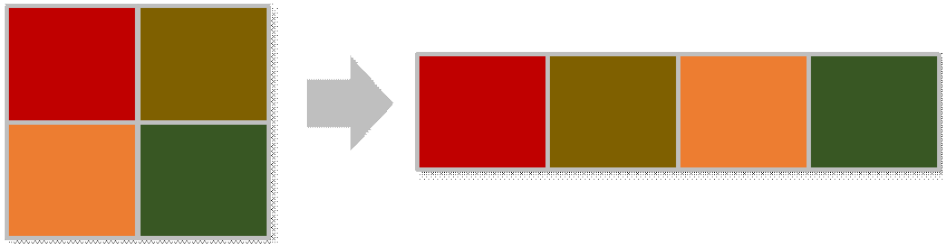
model.add(layers.Conv2D(32, (3, 3), (1, 1), 'same', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(rate=0.3))

model.add(layers.Conv2D(64, (3, 3), (1, 1), 'same', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(rate=0.3))
```


- 모델 설정 - 3

```
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(2, activation='sigmoid'))

model.summary()
```



- 학습 진행 - 1

- ✓ Adam은 Momentum과 RMSprop를 합친 경사하강법

```
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['acc'],
)
```

- 학습 진행 - 2

```
epochs = 32
history = model.fit_generator(
    trainGen,
    epochs=epochs,
    steps_per_epoch=trainGen.samples / epochs,
    validation_data=validationGen,
    validation_steps=trainGen.samples / epochs,
)
```

- 학습 결과 시각화 - 1

```
import matplotlib.pyplot as plt

def show_graph(history_dict):
    accuracy = history_dict['acc']
    val_accuracy = history_dict['val_acc']
    loss = history_dict['loss']
    val_loss = history_dict['val_loss']

    epochs = range(1, len(loss) + 1)

    plt.figure(figsize=(16, 2))
```

- 학습 결과 시각화 - 2

```
plt.subplot(121)
    plt.subplots_adjust(top=2)
    plt.plot(epochs, accuracy, label='Training accuracy')
    plt.plot(epochs, val_accuracy, label='Validation accuracy')

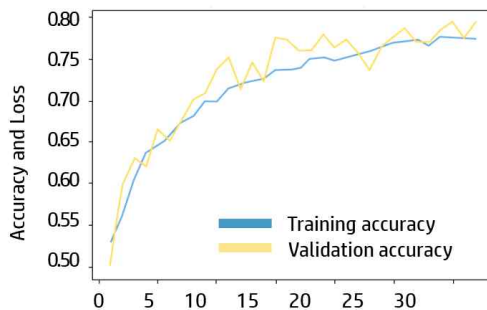
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1),
              fancybox=True, shadow=True, ncol=5)

plt.subplot(122)
    plt.plot(epochs, loss, label='Training loss')
    plt.plot(epochs, val_loss, label='Validation loss')
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1),
              fancybox=True, shadow=True, ncol=5)

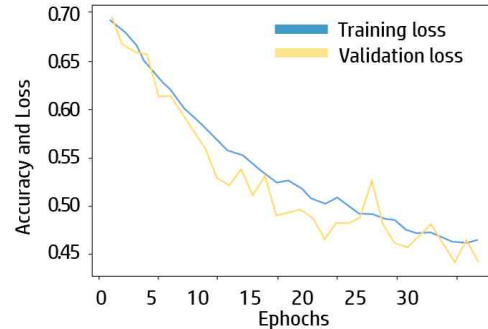
plt.show()
show_graph(history.history)
```

- 학습 결과 시각화 - 3
 - ✓ 실험 결과는 일부 달라질 수 있음

Training and validation accuracy and loss



Training and validation loss



- 모델 평가 - 1

```
testGenerator = ImageDataGenerator(rescale=1./255)
testGen = imageGenerator.flow_from_directory(
    os.path.join(rootPath, 'test_set'),
    target_size=(64, 64), )

model.evaluate_generator(testGen)
```

- 모델 평가 - 2

```
from tensorflow.keras.preprocessing.image import array_to_img
import numpy as np

cls_index = ['고양이', '개']

imgs = testGen.next()
arr = imgs[0][0]
img = array_to_img(arr).resize((128, 128))
plt.imshow(img)
result = model.predict_classes(arr.reshape(1, 64, 64, 3))
print('예측: {}'.format(cls_index[result[0]]))
print('정답: {}'.format(cls_index[np.argmax(imgs[1][0])]))
```

• 모델 평가 - 3



평가하기

1. 'Kaggle'은 딥러닝 및 머신러닝에 필요한 데이터셋을 유료로 제공한다.(O/X)

- 정답 : X

해설 : Kaggle에서는 무료로 딥러닝 및 머신러닝에 필요한 데이터셋을 제공합니다.

2. 모델 빌드 시 모델강화를 위해 데이터를 회전, 좌우반전, 이동, 역전 등으로 이미지를 바꾸는 단계는?

- ① 이미지 전처리
- ② 데이터 로드
- ③ 모델 설정
- ④ 학습 진행

- 정답 : ①번

해설 : 데이터 학습량을 늘려서 데이터가 변조되었을 때 모델이 이를 잘 캐치할 수 있도록 케라스에서 제공하는 이미지 제너레이터를 이용하여 전처리 과정을 수행합니다.

학습정리

1. 데이터셋 준비하기

- Kaggle : 2010년에 설립된 예측 모델 및 분석 대회 플랫폼, 딥러닝 및 머신러닝에 필요한 데이터셋을 무료로 제공

2. AWS를 이용한 CNN 모델 구축 단계

- Kaggle에서 데이터셋 다운로드
- 데이터셋 S3 업로드
- 데이터셋 EC2 다운로드

3. CNN 학습모델 빌드 및 평가

- 데이터 로드
- 이미지 전처리
- 데이터 분할
- 모델 설정
- 학습 진행
- 학습결과 시각화
- 학습결과 평가