5. EC2와 S3를 이용한 AI 개발

3강. 클라우드 기반의 CNN 실습

학습목표

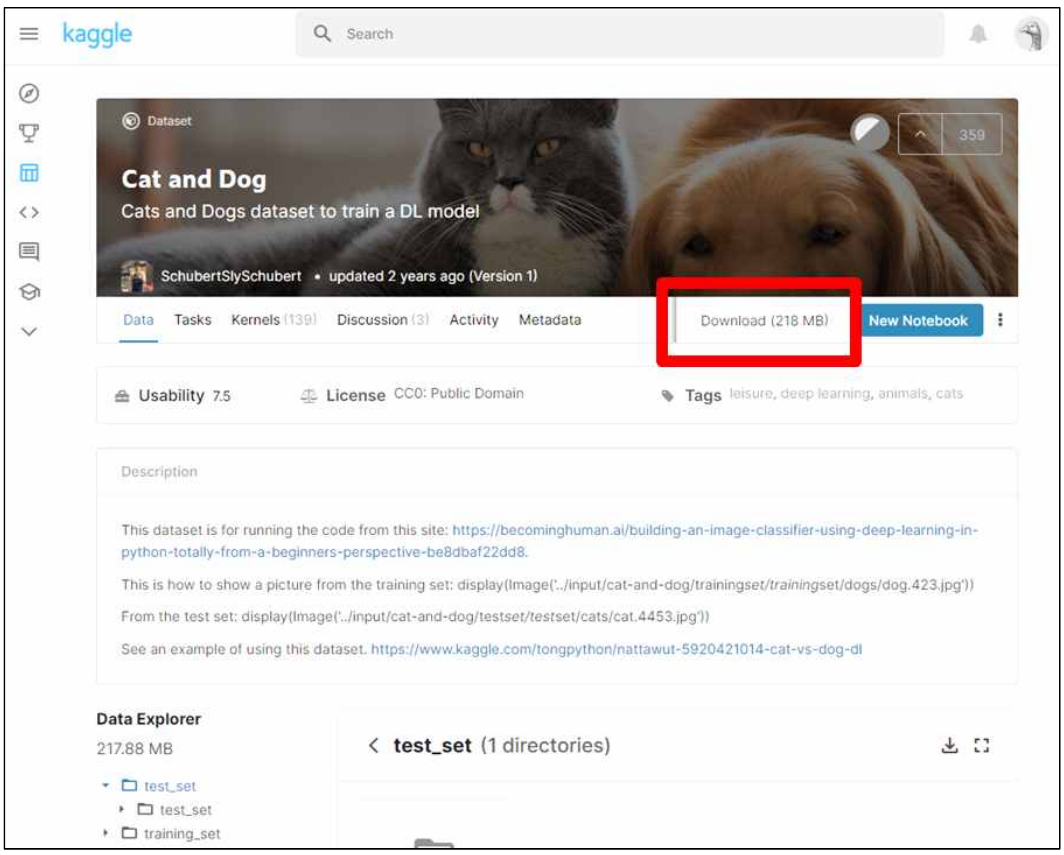
- EC2와 S3를 이용한 CNN 신경망 구축으로 학습 모델을 빌드하고 평가할 수 있다.

학습내용

- 데이터세트 준비하기
- CNN 학습모델 빌드하기
- CNN 학습모델 평가하기

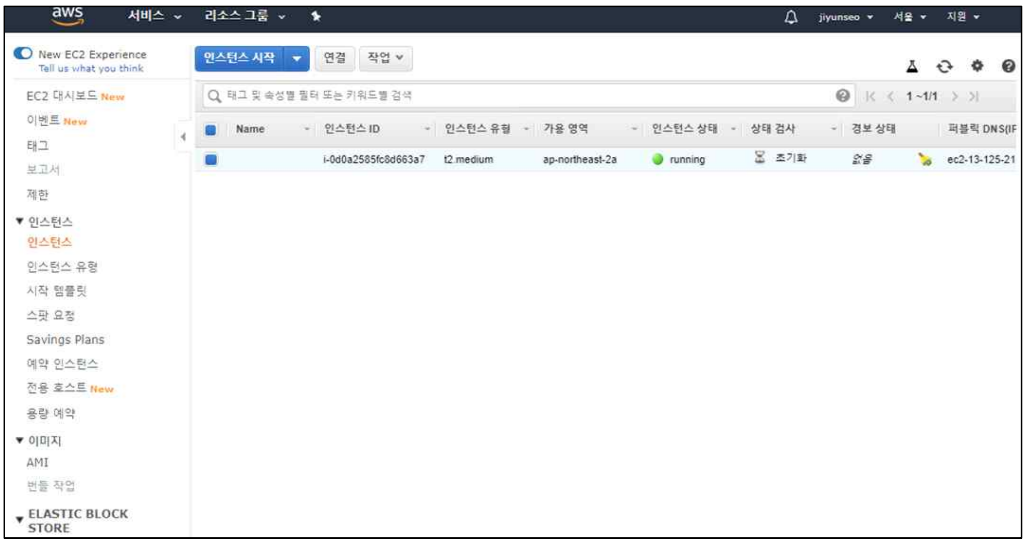
1. 데이터세트 준비하기

- 캐글 사이트 접속 후 cat and dog 데이터 다운로드

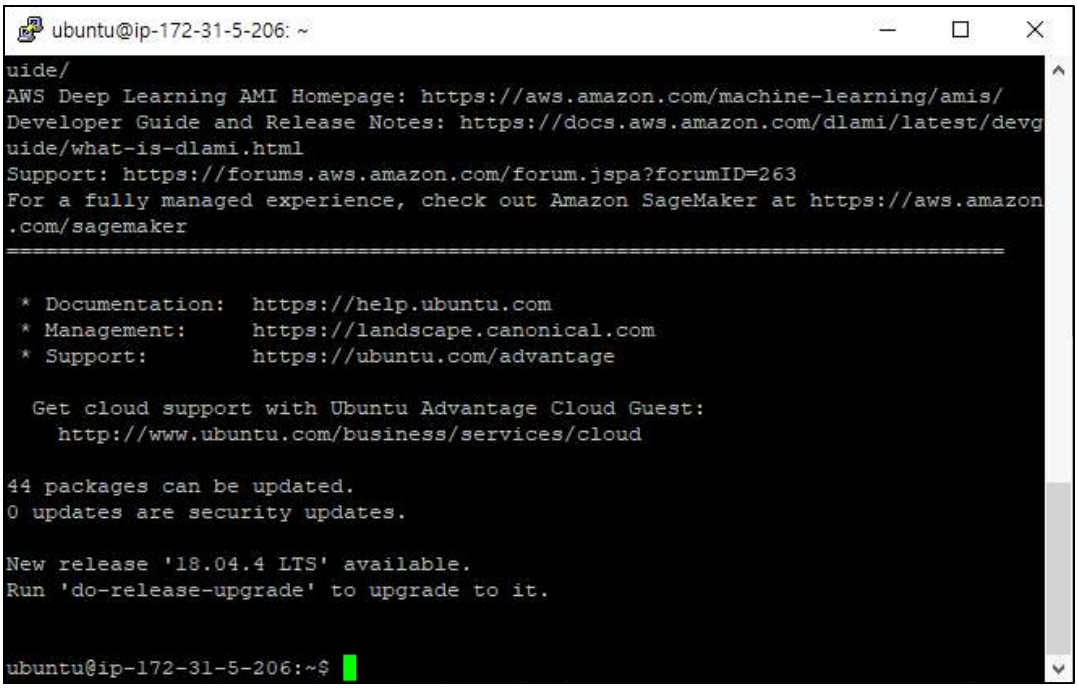


- 다운로드한 데이터 S3에 업로드

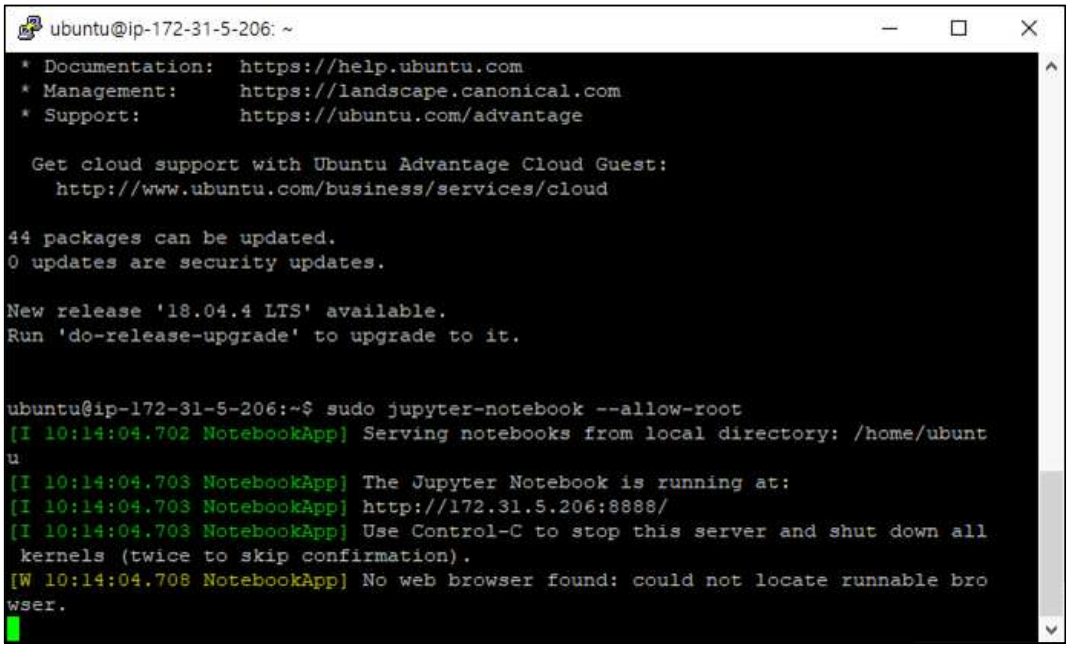
- AWS EC2 인스턴스 접속



• AWS EC2 접속 완료

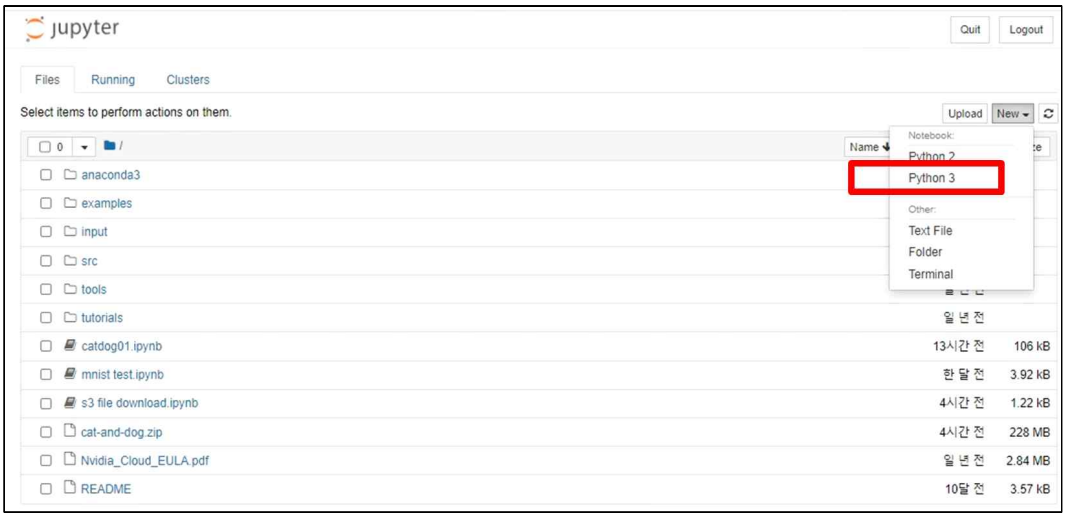


• 주피터 활성화



- 주피터 실행

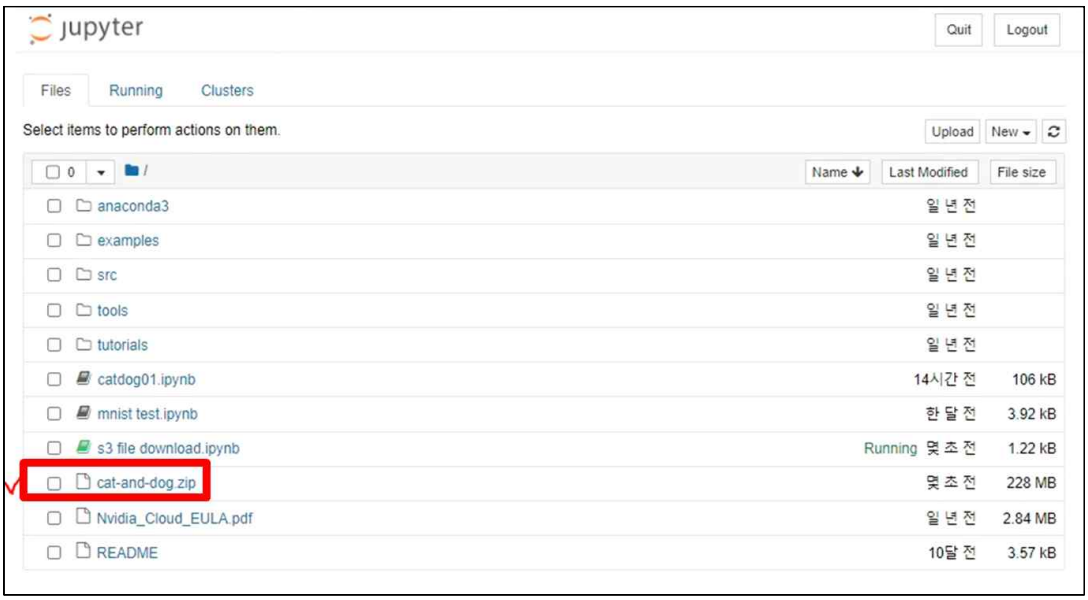
- 오른쪽 상단에 new 클릭 → python3 생성



- s3에서 파일 다운로드 명령어 입력 후 런(컴파일)

```
In [2]: import boto3
        import botocore

        bucket_name = 'bdu01'

        #name in s3
        in_file = 'cat-and-dog.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
```

- s3에서 파일 다운로드 완료



- 새 python3 생성 후 압축풀기 코드 입력

```
In [1]: import zipfile

        try:
            with zipfile.ZipFile("cat-and-dog.zip") as zf:
                zf.extractall()
                print("uncompress success")

        except:
            print("uncompress fail")

uncompress success
```

2. CNN 학습모델 생성하기

- 데이터 로드

```
#dataset https://www.kaggle.com/tongpython/cat-and-dog


import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

- 이미지 전처리

```
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

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

model = Sequential()
```

```
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))
```

```
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

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

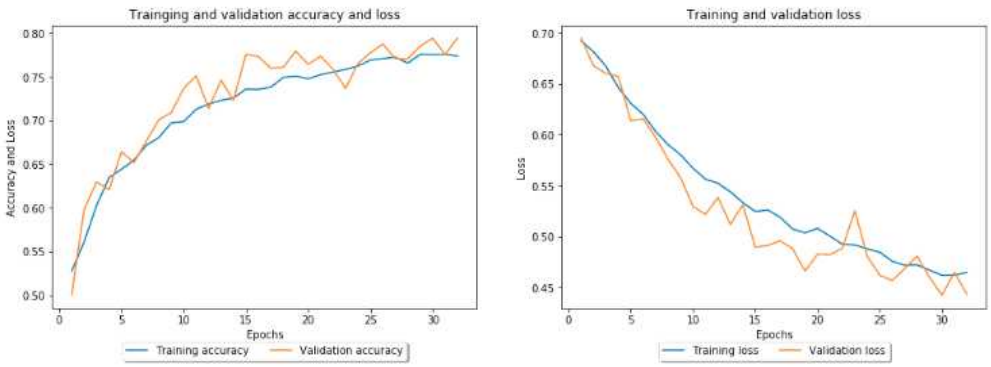
```
epochs = 32
history = model.fit_generator(
    trainGen,
    epochs=epochs,
    steps_per_epoch=trainGen.samples / epochs,
    validation_data=validationGen,
    validation_steps=trainGen.samples / epochs,
)
```

```
Epoch 24/32
225/225 [============================>.] - ETA: 0s - loss: 0.4632 - acc: 0.7794Epoch 1/32
226/225 [==============================] - 30s 134ms/step - loss: 0.4516 - acc: 0.7931
226/225 [==============================] - 66s 291ms/step - loss: 0.4647 - acc: 0.7783 - val_loss: 0.4533 - val_acc: 0.7931
Epoch 25/32
225/225 [============================>.] - ETA: 0s - loss: 0.4686 - acc: 0.7714Epoch 1/32
226/225 [==============================] - 30s 135ms/step - loss: 0.4487 - acc: 0.7890
226/225 [==============================] - 66s 291ms/step - loss: 0.4687 - acc: 0.7710 - val_loss: 0.4504 - val_acc: 0.7890
Epoch 26/32
225/225 [============================>.] - ETA: 0s - loss: 0.4707 - acc: 0.7683Epoch 1/32
226/225 [==============================] - 30s 134ms/step - loss: 0.4596 - acc: 0.7753
226/225 [==============================] - 65s 290ms/step - loss: 0.4708 - acc: 0.7681 - val_loss: 0.4613 - val_acc: 0.7753
Epoch 27/32
225/225 [============================>.] - ETA: 0s - loss: 0.4682 - acc: 0.7740Epoch 1/32
226/225 [==============================] - 30s 134ms/step - loss: 0.4558 - acc: 0.7830
226/225 [==============================] - 66s 290ms/step - loss: 0.4681 - acc: 0.7739 - val_loss: 0.4575 - val_acc: 0.7830
Epoch 28/32
225/225 [============================>.] - ETA: 0s - loss: 0.4604 - acc: 0.7813Epoch 1/32
226/225 [==============================] - 30s 134ms/step - loss: 0.4535 - acc: 0.7895
226/225 [==============================] - 66s 291ms/step - loss: 0.4614 - acc: 0.7805 - val_loss: 0.4552 - val_acc: 0.7895
Epoch 29/32
225/225 [============================>.] - ETA: 0s - loss: 0.4592 - acc: 0.7865Epoch 1/32
226/225 [==============================] - 30s 135ms/step - loss: 0.4319 - acc: 0.8019
226/225 [==============================] - 66s 291ms/step - loss: 0.4597 - acc: 0.7862 - val_loss: 0.4335 - val_acc: 0.8019
Epoch 30/32
225/225 [============================>.] - ETA: 0s - loss: 0.4543 - acc: 0.7857Epoch 1/32
226/225 [==============================] - 30s 134ms/step - loss: 0.4325 - acc: 0.7989
226/225 [==============================] - 66s 290ms/step - loss: 0.4535 - acc: 0.7861 - val_loss: 0.4341 - val_acc: 0.7989
Epoch 31/32
225/225 [============================>.] - ETA: 0s - loss: 0.4381 - acc: 0.7935Epoch 1/32
226/225 [==============================] - 30s 135ms/step - loss: 0.4360 - acc: 0.7951
226/225 [==============================] - 66s 291ms/step - loss: 0.4387 - acc: 0.7929 - val_loss: 0.4376 - val_acc: 0.7951
Epoch 32/32
225/225 [============================>.] - ETA: 0s - loss: 0.4398 - acc: 0.7887Epoch 1/32
226/225 [==============================] - 30s 135ms/step - loss: 0.4564 - acc: 0.7806
226/225 [==============================] - 66s 291ms/step - loss: 0.4395 - acc: 0.7888 - val_loss: 0.4581 - val_acc: 0.7806
```

3. CNN 학습모델 평가하기

- 학습결과 시각화

```
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))

    plt.subplot(121)
    plt.subplots_adjust(top=2)
    plt.plot(epochs, accuracy, label='Training accuracy')
    plt.plot(epochs, val_accuracy, label='Validation accuracy')
    plt.title('Trainging and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')

    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1),
               fancybox=True, shadow=True, ncol=5)
#     plt.legend(bbox_to_anchor=(1, -0.1))

    plt.subplot(122)
    plt.plot(epochs, loss, label='Training loss')
    plt.plot(epochs, val_loss, label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1),
           fancybox=True, shadow=True, ncol=5)
#     plt.legend(bbox_to_anchor=(1, 0))

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

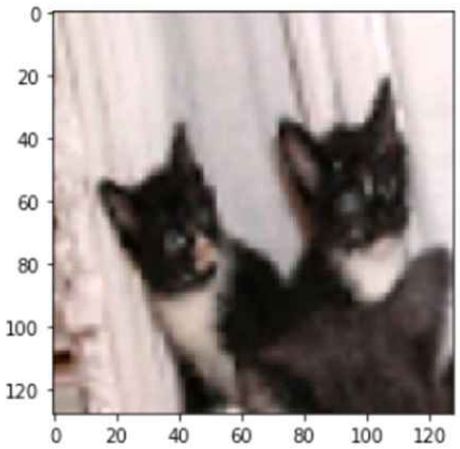


- 모델 평가

```
testGenerator = ImageDataGenerator(
    rescale=1./255
)

testGen = imageGenerator.flow_from_directory(
    os.path.join(rootPath, 'test_set'),
    target_size=(64, 64),
)

model.evaluate_generator(testGen)
```

```
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])]))
```

예측: 고양이
정답: 고양이



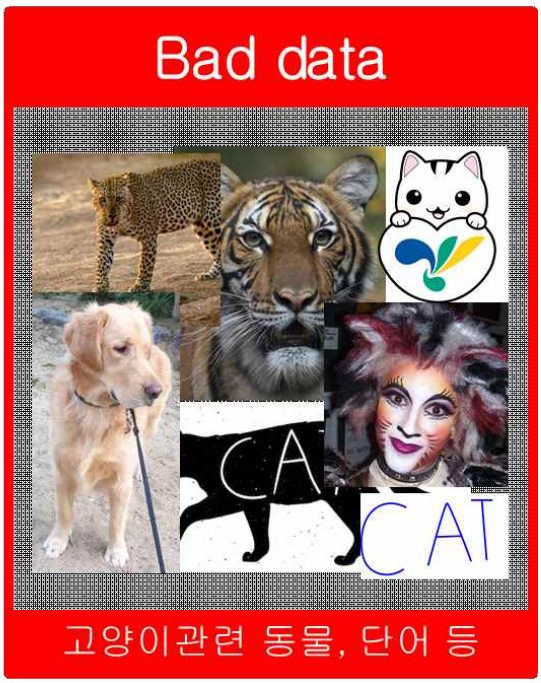
예측: 개
정답: 개

※ 나도 전문가다

- 학습률 향상을 위한 데이터 세트

  ✓ 고양이, 개 분류를 위하여 고양이 데이터 세트를 생성하는 경우

  - 캐글과 같은 검증된 사이트에서 공개된 데이터세트는 일반적으로 노이즈가 크게 없는 데이터들이지만 일반적으로 크롤링, 일부 오픈소스의 경우 노이즈가 포함된 데이터세트가 공유될 수 있음
  - 잘못된 데이터로 학습을 할 경우 좋지 못한 학습율을 얻을 수 있으며 학습율이 낮은 데이터 모델을 활용할 경우 부정적인 결과를 얻을 수 있음



평가하기

1. S3 데이터를 EC2로 다운로드 하기 위해 사전에 수행해야 할 작업이 아닌 것은?

① S3 버킷 생성

② S3 버킷에 데이터 업로드

③ 버킷정책 편집 및 입력

④ 모든 퍼블릭 액세스 차단 활성화

- **정답 : ④ 번**

해설 : S3 데이터를 EC2로 다운로드 하기 위하여 모든 퍼블릭 엑세스 차단은 비활성화 되야 합니다.

2. CNN 학습 모델 빌드 단계에서 비어있는 자리에 적절한 단계를 순서대로 나열하면?

데이터로드 → [ ] → [ ] → [ ] → 학습진행 → [ ]

① 이미지 전처리

② 학습 모델 시각화

③ 모델 설정

④ 데이터 분할

- **정답 : ① ④ ③ ② 번**

학습정리

1. **데이터세트 준비하기**
   - Kaagle에서 데이터세트 다운로드
   - S3에 데이터 업로드
   - EC2로 데이터 다운로드
2. **CNN 학습모델 빌드하기**
   - 데이터로드
   - 이미지 전처리
   - 데이터 분할
   - 모델 설정
   - 학습 진행
   - 학습 모델 시각화
3. **CNN 학습모델 평가하기**
   - 모델평가