2주차

- **2주차 목표 : 프로젝트 구체화**

- **프로젝트 이슈**
  - 2019~2023년 사이에 다양한 지역에서 포트홀(도로 파임)이 2만건에 육박하는 것으로 나타남.
  - 포트홀 건수 : 2019년 → 3717건, 2020년 → 4440건, 2021년 → 4285건, 2022년 → 4509건, 2023년 → 2474건을 기록함.
  - 총 피해 배상도 → 1737건, 배상액 → 35억원에 육박함.
  - 도로 파손으로 인해 3년간 배상액 약 100억원임.

도로위의 지뢰 '포트홀', 고속도로서 5년간 1만9000여 건 발생

최근 5년간 고속도로에서 발생한 도로위의 지뢰밭이라 불리는 '포트홀'(도로파임)이 2만건에 육박하는 것으로 나타났다. 또 이...

https://www.sedaily.com/NewsView/29TGUW2LND

[단독] 기후의 역습...'도로의 지뢰' 포트홀 12월에 2배 급증 | 중앙일보

지난해 12월 서울 시내에서 발생한 포트홀이 전년 동월 대비 2배 수준으로 늘었다. 하지만 지난해 12월에는 유독 평년 대비 포트홀 발생 건수가 많았다. 서울시 관계자는 "지난해 12월엔 봄 같은 날씨를 보이다 급격히 영하권으로 기온이 곤

https://www.joongang.co.kr/article/25224782#home

[단독] 도로공사, 도로포장 파손으로 매년 수십억 배상

[디스커버리뉴스=강성덕 기자] 한국도로공사(사장 함진규)가 도로 포장 훼손으로 인해 지난해 11월까지 배상한 비용이 41억원이 넘는 것으로 확인됐다. 배상건수별로는 2832건으로 '22년에는 1737건에 약 34

https://www.discoverynews.kr/news/articleView.html?idxno=1038710

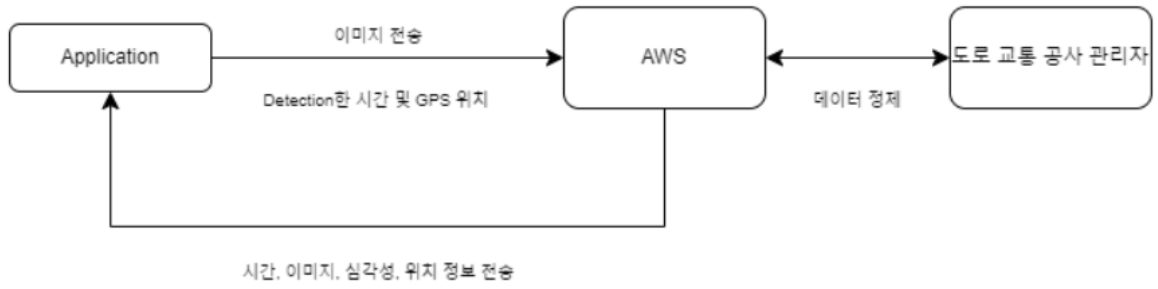
→ 기후변화로 인한 이상 기상현상 발생 및 도로노후화로 인하여 포트홀 증가, 도로시설물 파손 증가

안전, 비용 상의 문제로 보수공사의 필요성

차량 전방 이미지 촬영을 통하여 도로 및 시설물 파손을 감지하는 “도로 노후화 및 파손 감지 AI 서비스” 기획

- **프로젝트 이슈 해결**
  - Saveway 애플리케이션을 통해 실시간으로 도로 파손을 감지하여, 포트홀이나 도로 균열 등 다양한 도로 정보 수집을 할 수 있다.
  - 실시간 감지 시간, 도로 파손 이미지, 도로 파손 위치를 알 수 있기 때문에, 차량 사고가 나기 전에 빠르게 도로 보수를 할 수 있다.

- **프로젝트 구체화**



- **팀원 역할**

팀원	역할
팽준호	Application 개발
오수만	서버 개발
박근태	데이터 수집 및 전처리
최민혁	서버 개발
박서영	데이터 수집 및 전처리
김재영	YOLO V9 모델 개발 서버 개발
정지연	서버 AI 모델 개발

- 데이터 수집
  - AI-HUB에서의 이미지와 이미지 크롤링을 통해 데이터 셋 구축

Application

- Frontend : Android
- Backend : AWS, Flask
- AI 모델 : Tensorflow, Pytorch

페이지 구성

1. 기능
   - 카메라를 통한 감지
   - 실시간 객체 탐지
   - 지도를 통해 위치 정보 저장 및 확인
2. 메인 화면
3. 결과
   - 대시 보드 형태로 도로 상태 정보와 월, 연도 별로 도로 파손으로 인한 배상액 등을 나타냄.

서버AI

서버 측 인공지능 모델 목적

1. Application에서 YOLOv9를 사용하여 1차적으로 Object detection을 수행한 이미지 데이터에서 개별적인 object를 cropping 및 warping 하여 서버로 전송한다.
2. 서버에 Classification을 위한 인공지능 모델을 구축한다.
3. Image classification을 위한 여러가지 모델들을 구축한다. 실제 데이터셋을 적용하여 성능을 비교하고 평가한 후, 도로 노후화 및 파손 감지를 위한 최적의 모델을 선택한다.

2주차 진행 목표

- 기본적인 CNN 모델 (AlexNet) 구축 및 훈련과 성능 평가
- 우선 지자체 도로부속시설물 파손 이미지를 AI HUB에서 다운로드 하고, 불필요한 데이터셋을 삭제하고 중복을 제거한 상태이다.

1. Class를 PE드럼 파손, PE방호벽 파손, PE안내봉 파손, PE입간판 파손, PE휀스 파손, 도로 갈라짐 ,라바콘 파손, 시선유도봉 파손, 제설함 파손 총 8가지로 나누어야 한다.
2. CNN 전체 구조를 만들고 train-test set을 7:3으로 나눈다. 마지막 최종 예측 테스트를 위하여 single image를 클래스 별로 1장씩 빼놓는다.
3. CNN을 컴파일링 하고 train set을 가지고 훈련하며 test set으로 성능 평가를 한다.

2주차 진행 상황

```

train_datagen = ImageDataGenerator(
    rescale = 1./255, # 신경망이 이미지 데이터 픽셀값 0부터 255 값을 잘 처리 못할 수도 있어서 0부터 1 사이의 값으로 만들기 위해서 255로 나눠준 것이다.
    shear_range = 0.2, # shear (전단) 변환 : 이미지를 일정 각도로 밀어 왜곡하는 변환이다. x축에서 밀수도 있고 y축에서 밀 수도 있다.
    zoom_range = 0.2,
    horizontal_flip = True
)

```

훈련 데이터셋 생성 → 지자체 도로부속시설물 파손 이미지 데이터셋을 훈련에 알맞도록 전 처리 한다.

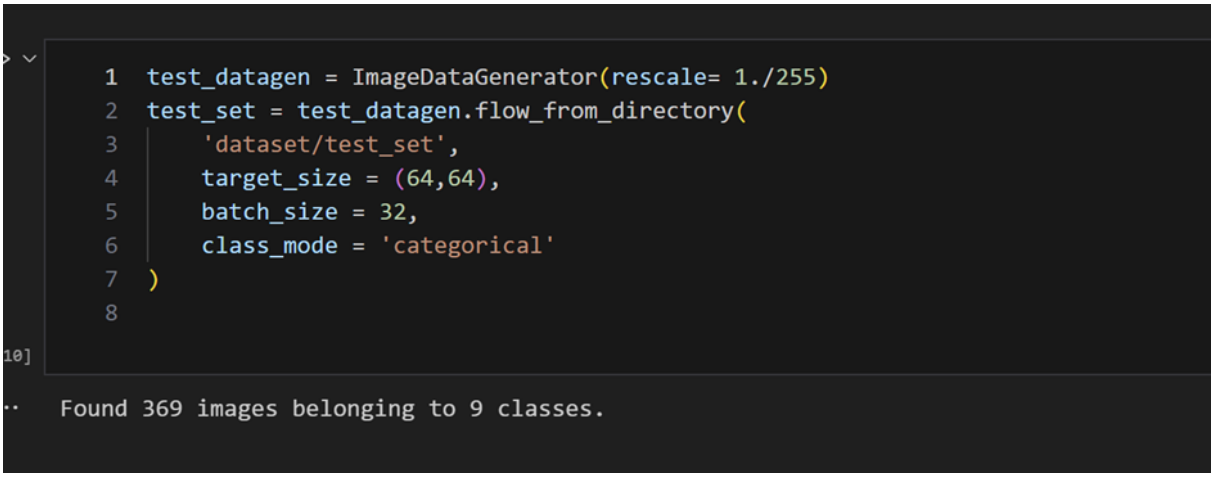

```

train_datagen = ImageDataGenerator(
    rescale = 1./255, # 신경망이 이미지 데이터 픽셀값 0부터 255 값을 잘 처리 못할 수도 있어서 0부터 1 사이의 값으로 만들기 위해서 255로 나눠준 것이다.
    shear_range = 0.2, # shear (전단) 변환 : 이미지를 일정 각도로 밀어 왜곡하는 변환이다. x축에서 밀수도 있고 y축에서 밀 수도 있다.
    zoom_range = 0.2,
    horizontal_flip = True
)

```

전처리한 데이터 셋을 convolution network의 input image로 들어갈 수 있게 조정한다.

해당 모델에서는 input image의 size를 (64,64)로 지정해주었고, batch size는 32로 설정하였다.

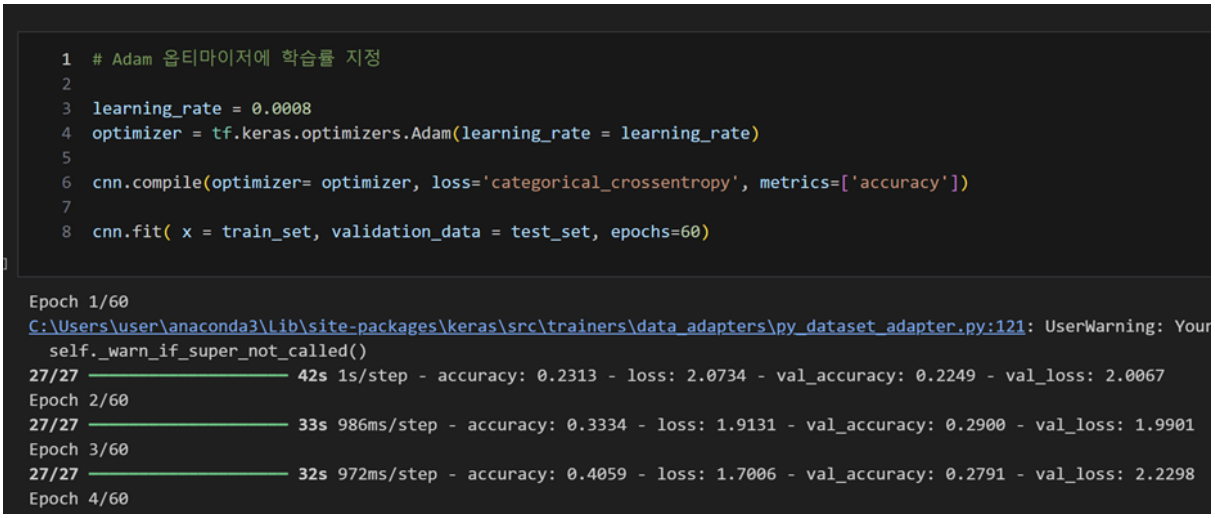
현재 모델이 분류해야 하는 클래스가 2가지가 아닌 PE드럼 파손, PE방호벽 파손, PE안내봉 파손, PE입간판 파손, PE휀스 파손, 도로 갈라짐 ,라바콘 파손, 시선유도봉 파손, 제설함 파손, 총 8가지 이므로 다중분류를 위한 class_mode를 categorical로 설정하였다.

Train과 test dataset의 비율은 70:30으로 설정해주었다.



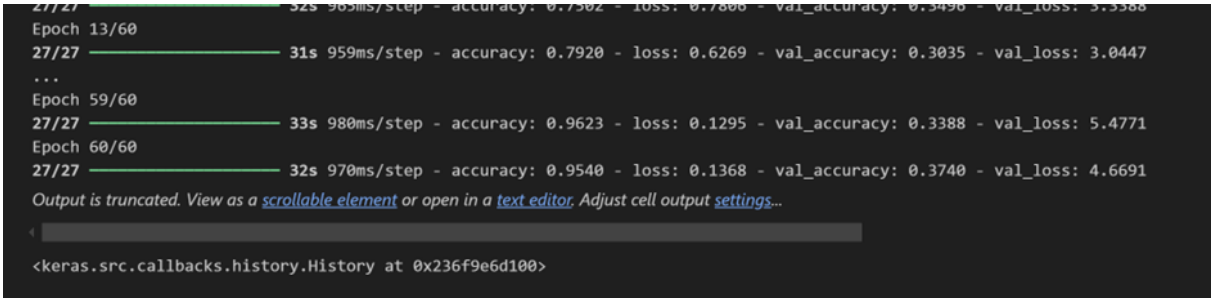
```
test_datagen = ImageDataGenerator(rescale= 1./255)
test_set = test_datagen.flow_from_directory(
    'dataset/test_set',
    target_size = (64,64),
    batch_size = 32,
    class_mode = 'categorical'
)
```

```
Found 369 images belonging to 9 classes.
```



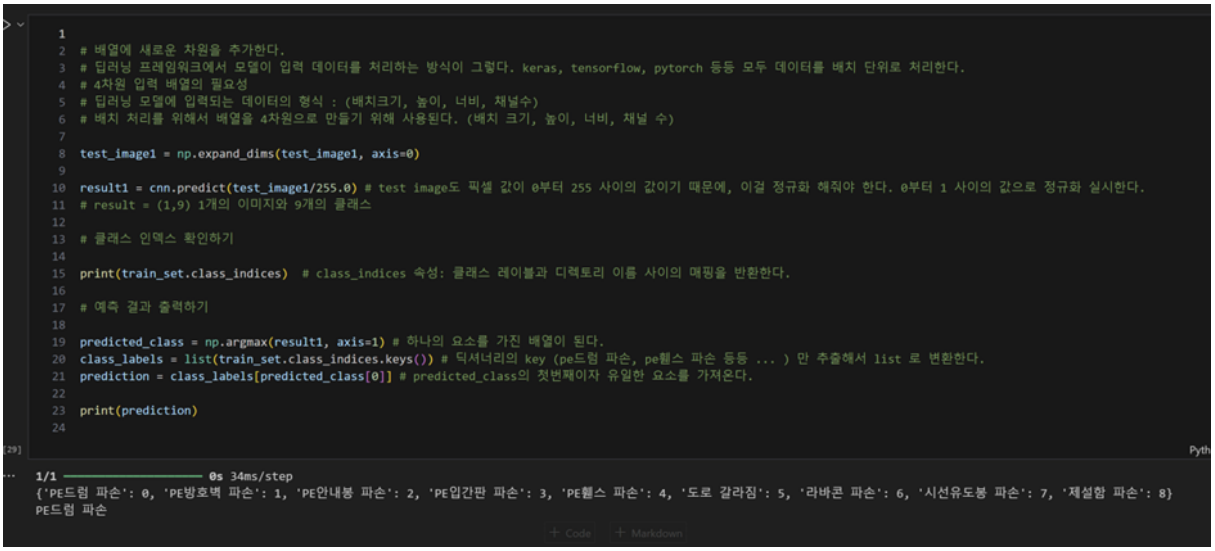
```
# Adam 옵티마이저에 학습률 지정

learning_rate = 0.0008
optimizer = tf.keras.optimizers.Adam(learning_rate = learning_rate)

cnn.compile(optimizer= optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

cnn.fit( x = train_set, validation_data = test_set, epochs=60)
```

```
Epoch 1/60
C:\Users\user\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your
  self._warn_if_super_not_called()
27/27 ━━━━━━━━━━━━━━━━━━━━ 42s 1s/step - accuracy: 0.2313 - loss: 2.0734 - val_accuracy: 0.2249 - val_loss: 2.0067
Epoch 2/60
27/27 ━━━━━━━━━━━━━━━━━━━━ 33s 986ms/step - accuracy: 0.3334 - loss: 1.9131 - val_accuracy: 0.2900 - val_loss: 1.9901
Epoch 3/60
27/27 ━━━━━━━━━━━━━━━━━━━━ 32s 972ms/step - accuracy: 0.4059 - loss: 1.7006 - val_accuracy: 0.2791 - val_loss: 2.2298
Epoch 4/60
```



동일한 구조의 convolution layer 및 pooling layer를 적용하였을 때, 하이퍼 파라미터를 다양하게 적용해 보았으나, 모두 비슷한 성능을 보였다.

훈련했던 모델 중에서 가장 accuracy가 높은 hyper parameter는 학습률=0.0008에 epoch=60 이었다. 그리고 optimizer는 adam을 사용하였다.



학습 및 테스트에 쓰이지 않은 별도의 데이터 셋을 class별로 저장해주고, 학습된 모델로 실제 서비스에 쓰이는 것처럼 테스트를 해보았다.

첨부된 이미지에서는 대표적으로 PE드럼 파손이 올바르게 Classification 되는 모습이다.

이외의 모든 class가 올바르게 평가되었다.

3주차 진행 목표

- CNN (AlexNet) 은 Classification에 쓰이는 가장 기본적인 모델이므로, 모바일 애플리케이션과 원활한 네트워크가 이루어지기에 무리가 있다고 판단하였다.
- R-CNN 및 Fast R-CNN, Faster R-CNN을 사용하려고 하였으나, 해당 모델은 복잡한 구조와 연산이 필요하다. R-CNN의 1 stage인 region proposal 단계를 거쳐서 제안된 영역에

대해서 모두 CNN을 실행하여 Classification을 진행하므로 매우 느리고 많은 계산 자원이 필요하다. 그리하여 프로젝트 주제인 실시간 객체 탐지와는 거리가 멀다고 판단하였다.

⇒ 비교적 경량화되어있는 ViT, PocketNet, MobileNet V4를 고려하고 있다.

백엔드

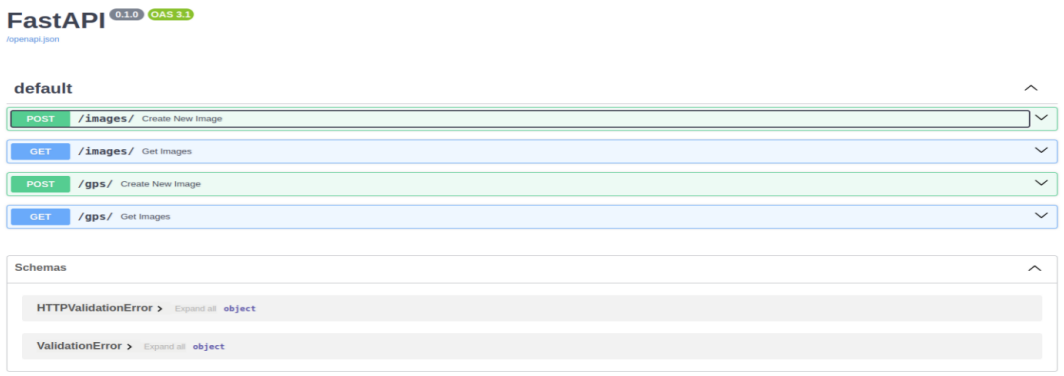
1. API

   서버에서 사용할 API를 로컬 환경에서 구현 및 테스트

   - 서버에서 사용할 API를 로컬 환경에서 먼저 구현한 후, 충분한 테스트를 통해 기능을 검증한다
   - 테스트 완료 후, 해당 API를 서버에 구축할 준비를 한다.

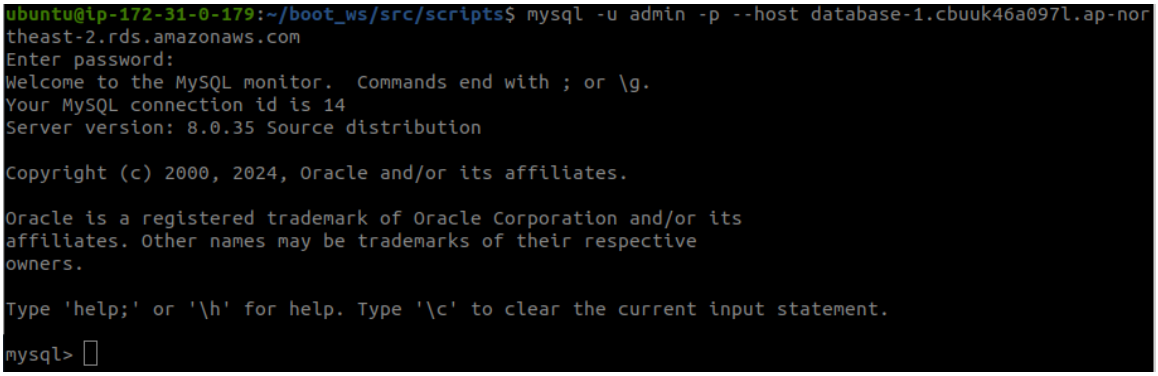
   1) Fast API 라이브러리 사용

   - image , gps를 비트맵 형식으로 저장, 업로드



2. 서버 구축

   - AWS EC2 및 RDS를 생성하고 연동, 연결 성공



- EC2 인스턴스를 생성하여 서버 환경을 구축하고, RDS를 통해 데이터베이스를 설정.
- 로컬에서 테스트한 API를 EC2 서버에 배포하고, RDS와 연동하여 데이터 저장 및 관리를 진행.

3. EC2 서버에 API 배포
    - EC2 서버에 ssh 원격 연결
    - ~/etc/nginx/sites-enabled 경로에 fastapi_nginx 생성 및 설정
    - EC2 의 퍼블릭 ipv4 주소를 사용하여 http로 접속 및 확인
        - https::는 434 포트, http:는 80포트이기 때문