인공지능 팀 프로젝트

SIA

(1) Semantic Segmentation

01

팀 소개 & 문제 설명

팀 소개

- **팀 이름 :** 미로
- **팀원 별 역할 소개 :**

  김민지 (팀장) : 베이스 라인 분석, 데이터 분석, JSON 파일로부터 학습/ 테스트를 위한 label 데이터 셋을 구축하는 코드 작성, 모델 훈련 및 성능 개선, ppt 작성, 발표

  이민주 : 베이스 라인 분석, FPS 측정 코드 작성, 학습 코드/ 테스트 코드 분리

  정인상 : 베이스 라인 분석, 데이터 전처리, 데이터 셋 추가 확보를 위한 자료조사

  정지용 : 베이스라인 분석, JSON 포맷에서 데이터를 파싱한 뒤 마스킹한 이미지를 만드는 코드 작성

문제설명

해결해야 할 문제 : 인공위성 영상에서 건물과 도로를 탐지한다.

→ **건물과 도로를 동시에 검출하는** 모델을 개발해야 함

02

데이터 분석

데이터 분석

Image :

건물과 도로가 같이 있는 이미지로 png 파일로 존재

label :

건물과 도로가 같이 있는 이미지에 대한 label이 png 파일로 존재하지 않기 때문에 json 파일을 열어 **학습/ 테스트를 위한 label 데이터 셋을 구축**해야 함.

데이터 분석

```
for json_name in json_list:

    print(Path(json_name).stem)

    with open(os.path.join(train_building, json_name), "r") as b_json_data,

        mask = np.zeros((1024, 1024,3), dtype="uint8")

        b_json_object = json.load(b_json_data)
        r_json_object = json.load(r_json_data)

        for b_element in b_json_object["features"]:

          b_imcoords = b_element["properties"]["building_imcoords"]

          if b_imcoords == "":
            continue

          b_splited_imcoords = b_imcoords.split(",")
          b_divided_imcoords = list_chunk(b_splited_imcoords, 2)

          b_polygon = np.array(b_divided_imcoords)
          b_polygon = np.array(b_polygon, np.float64)
          b_polygon = np.array(b_polygon, np.int32)

          mask1 = cv2.fillPoly(mask, [b_polygon], (153,51,153))
```

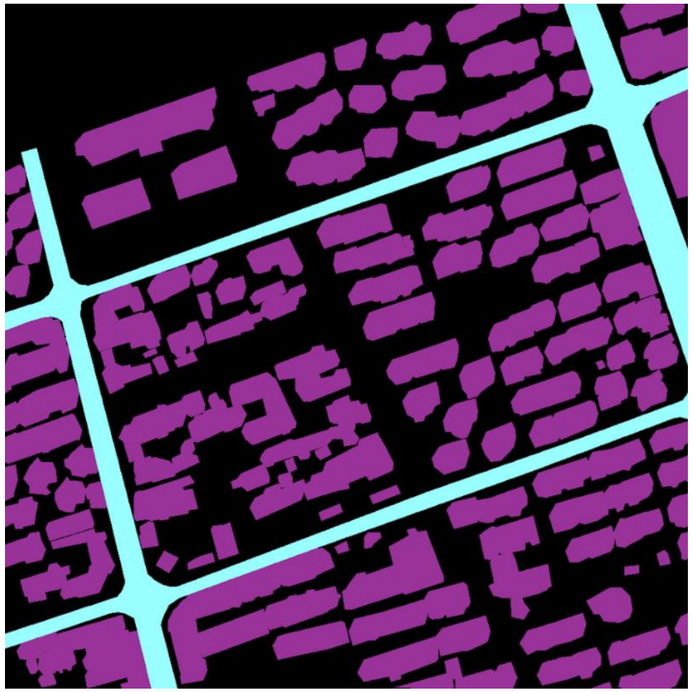
건물 Jason 파일을 열어 건물 좌표 값에 해당하는 **'building_imcoords' 값을 추출**하여 마스킹 이미지를 만들어 줌

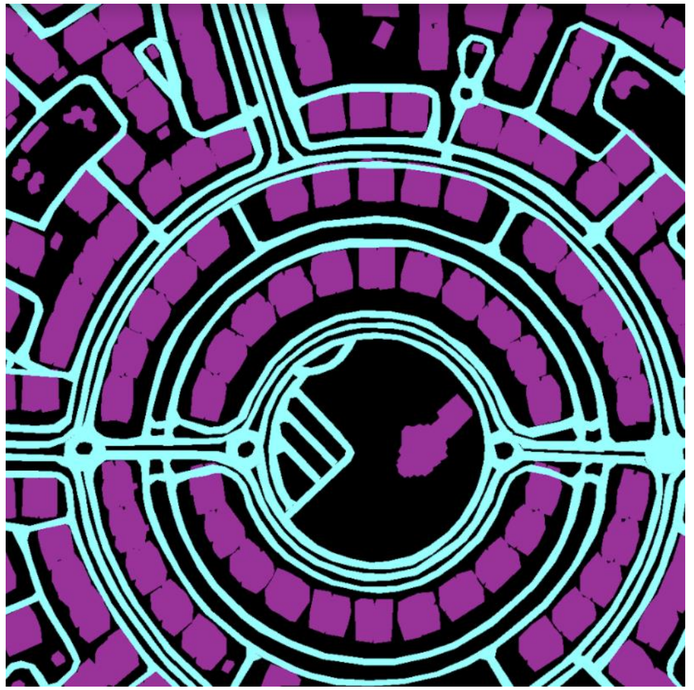
데이터 분석

```
for r_element in r_json_object["features"]:

  r_imcoords = r_element["properties"]["road_imcoords"]

  if r_imcoords == "":
    continue

  r_splited_imcoords = r_imcoords.split(",")
  r_divided_imcoords = list_chunk(r_splited_imcoords, 2)

  r_polygon = np.array(r_divided_imcoords)
  r_polygon = np.array(r_polygon, np.float64)
  r_polygon = np.array(r_polygon, np.int32)

  mask2 = cv2.fillPoly(mask1, [r_polygon], (255,255,153))

plt.imshow(mask2)
plt.show()
cv2.imwrite(os.path.join(n_train, Path(json_name).stem+ '.png'), mask2)
```

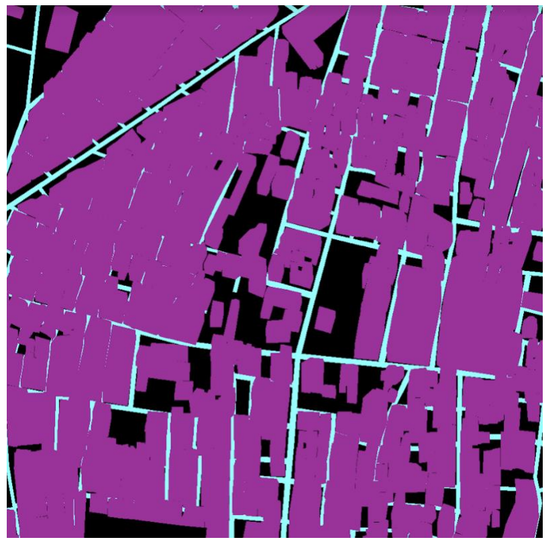
도로 Jason 파일을 열어 건물 좌표 값에 해당하는 **'road_imcoords' 값을 추출**하여 마스킹 이미지를 만들어 줌

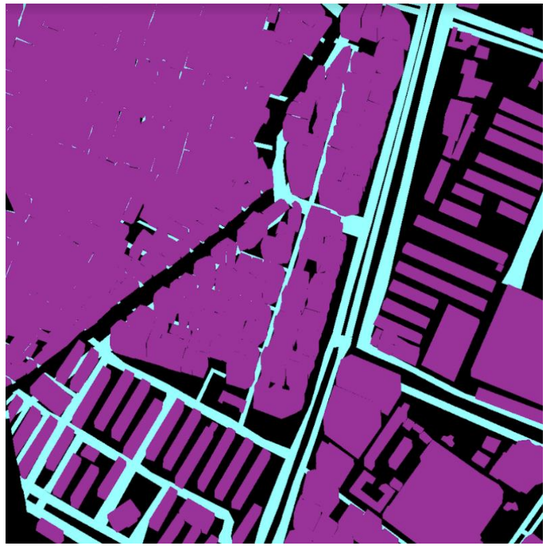
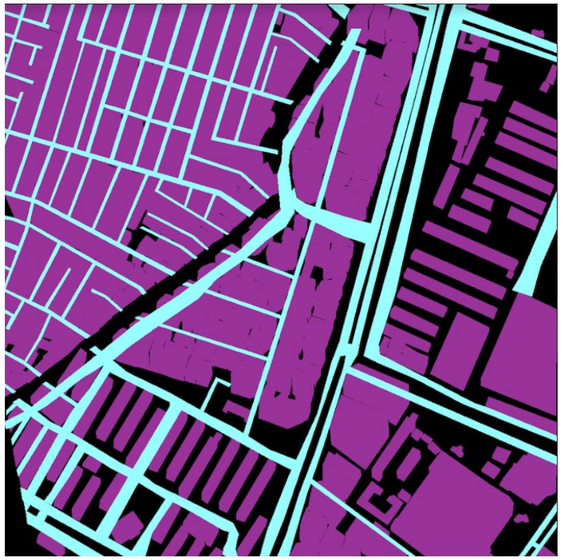
데이터 분석





데이터 분석



건물과 도로를 **그리는 순서에 따라 차이 발생**

위와 같은 이미지의 경우 **도로가 건물에 가려져 도로에 대한 학습이 잘 되지 않을 가능성**이 있음.

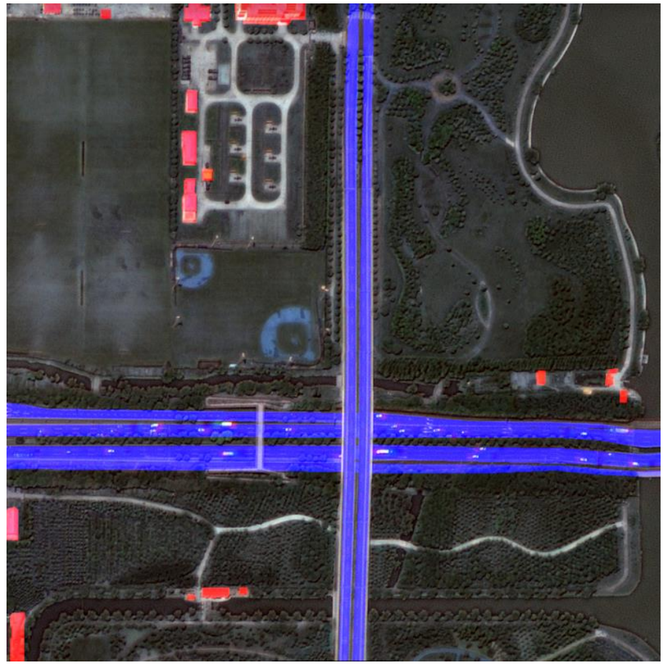
이를 방지하기 위해 건물을 먼저 그리고 그 위에 도로를 그리도록 함.

데이터 분석

샘플 이미지, 레이블 시각화 :

샘플 이미지 위에 레이블 정보 겹쳐 보이도록 시각화

데이터 분석



도로에 대한 부정확한 레이블링 발견:

도로처럼 보이는 곳이 레이블링 되어 있지 않거나 도로 레이블링이 중간에 끊긴 것처럼 보임.

03

데이터 전처리

데이터 전처리

Dataset 설정 :

dataset 을 이용하여 data를 손쉽게 불러서 사용할 수 있음

```
def __init__(
        self,
        images_dir,
        masks_dir,
        classes=None,
        augmentation=None,
        preprocessing=None,
):
    self.ids = os.listdir(images_dir)
    self.images_fps = [os.path.join(images_dir, image_id) for image_id in self.ids]
    self.masks_fps = [os.path.join(masks_dir, image_id) for image_id in self.ids]

    # convert str names to class values on masks
    self.class_values = [self.CLASSES.index(cls.lower()) for cls in classes]
```

__init__: 원본 이미지와 label이미지를 저장한 경로를 입력 받아 가져오고, 그 경로에 있는 이미지 파일들을 리스트로 만들어줌, Class 를 설정해줌

데이터 전처리

__getitem__: 이미지 파일 리스트로부터 해당하는 index의 이미지를 가져옴

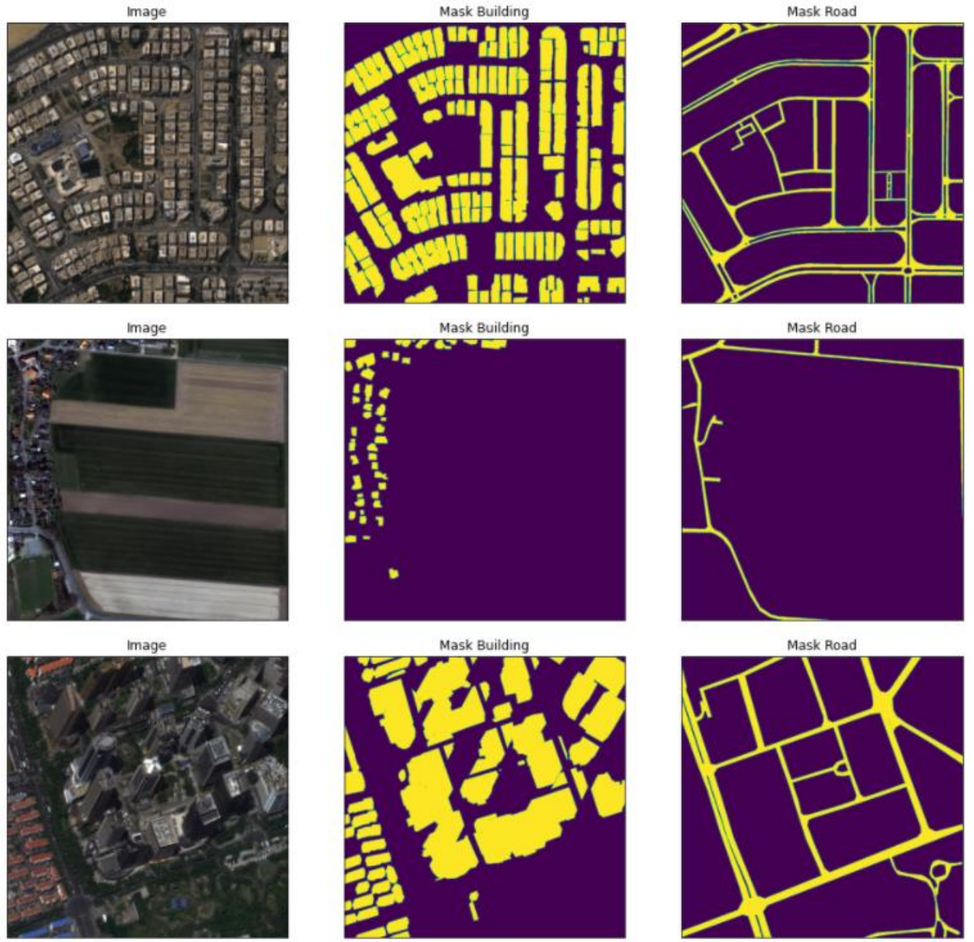
```
def __getitem__(self, i):

    # read data
    image = cv2.imread(self.images_fps[i])
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    mask = cv2.imread(self.masks_fps[i], 0)
```

__len__: 데이터 셋의 길이, 즉 총 샘플 수를 리턴

```
def __len__(self):
    return len(self.ids)
```

데이터 시각화

가지고 있는 데이터를 살펴보기 위한 시각화 :
랜덤으로 3개의 원본 이미지와 label 이미지를 가져와 시각화



데이터 증대

Albumentations library 사용:

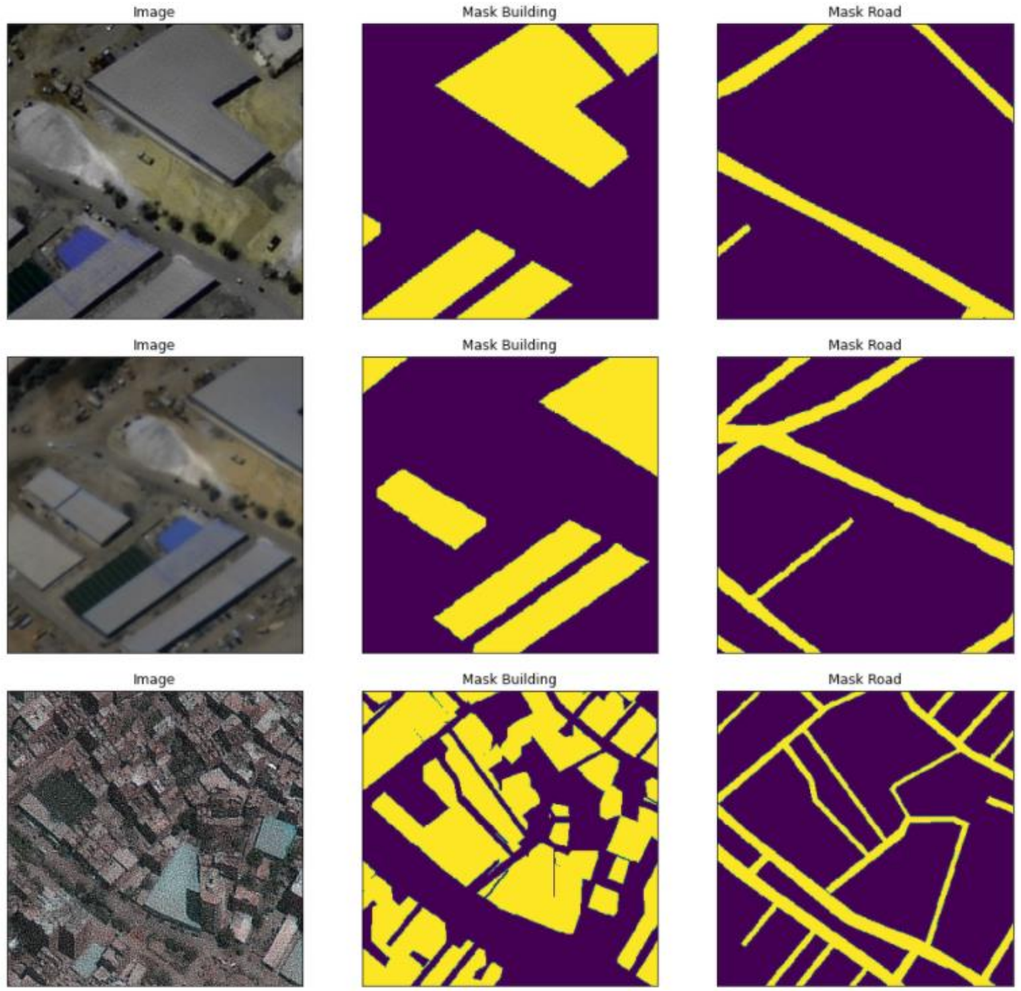
이미지를 상하좌우로 뒤집거나 자르는 방식 등 다양하게 데이터를 증대

```
def get_training_augmentation():
    train_transform = [

        albu.HorizontalFlip(p=0.5),

        albu.ShiftScaleRotate(scale_limit=0.5, rotate_limit=0, shift_limit=0.1, p=1, border_mode=0),

        albu.PadIfNeeded(min_height=320, min_width=320, always_apply=True, border_mode=0),
        albu.RandomCrop(height=320, width=320, always_apply=True),

        albu.IAAAdditiveGaussianNoise(p=0.2),
        albu.IAAPerspective(p=0.5),

        albu.OneOf(
            [
                albu.CLAHE(p=1),
                albu.RandomBrightness(p=1),
                albu.RandomGamma(p=1),
            ],
            p=0.9,
        ),
```

데이터 증대

증대한 데이터 시각화:

상하좌우로 뒤집히거나 잘리는 등
다양한 방식으로 변화된 것을 볼 수 있음



04

네트워크 설계 & 학습방법

모델 생성 및 훈련

네트워크 구조:

U-Net 모델사용, ImageNet 데이터 세트로 학습된 SE-ResNet을 Backbone으로 사용

네트워크 선택 이유:

처음 고려 대상이 되었던 모델은 **U-net, DeepLap V3**.

비교를 위해 **두 가지 모델에 대해 동일한 조건으로 학습하여 성능 평가**.

DeepLabV3 mIOU : 0.499, U-Net mIOU : 0.536

모델 생성 및 훈련

학습방법:

```
train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True, num_workers=2)
valid_loader = DataLoader(valid_dataset, batch_size=2, shuffle=False, num_workers=1)
```

```
loss = smp.utils.losses.DiceLoss()
metrics = [
    smp.utils.metrics.IoU(threshold=0.5),
]

optimizer = torch.optim.Adam([
    dict(params=model.parameters(), lr=0.0001),
])
```

```
for i in range(0, 500):
    print('\nEpoch: {}'.format(i))
    train_logs = train_epoch.run(train_loader)

    start_time = time.time()
    valid_logs = valid_epoch.run(valid_loader)
    end_time = time.time()
```

모델 생성 및 훈련

학습방법:

마스킹 이미지 색깔에 따른 차이:

여러 가지 색깔로 바꿔가며 비교

→ **흰색에 가까울수록 더 잘 예측하는 것으로 판단**.

건물 - (200, 200, 200)

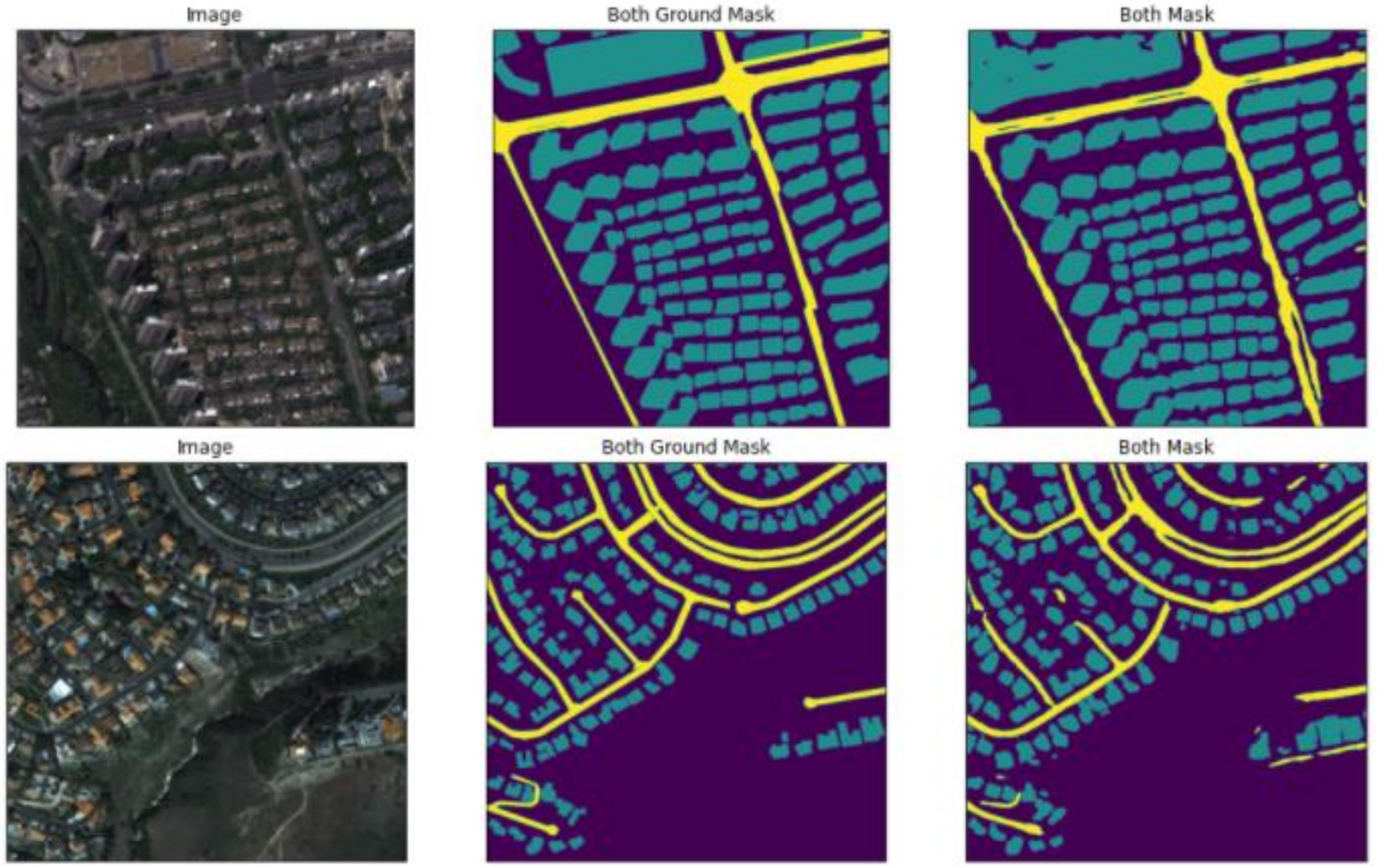
도로 - (255, 255, 255)

05

실험결과 분석

최종 결과:

```
max_score = 0.7026475781286627
```



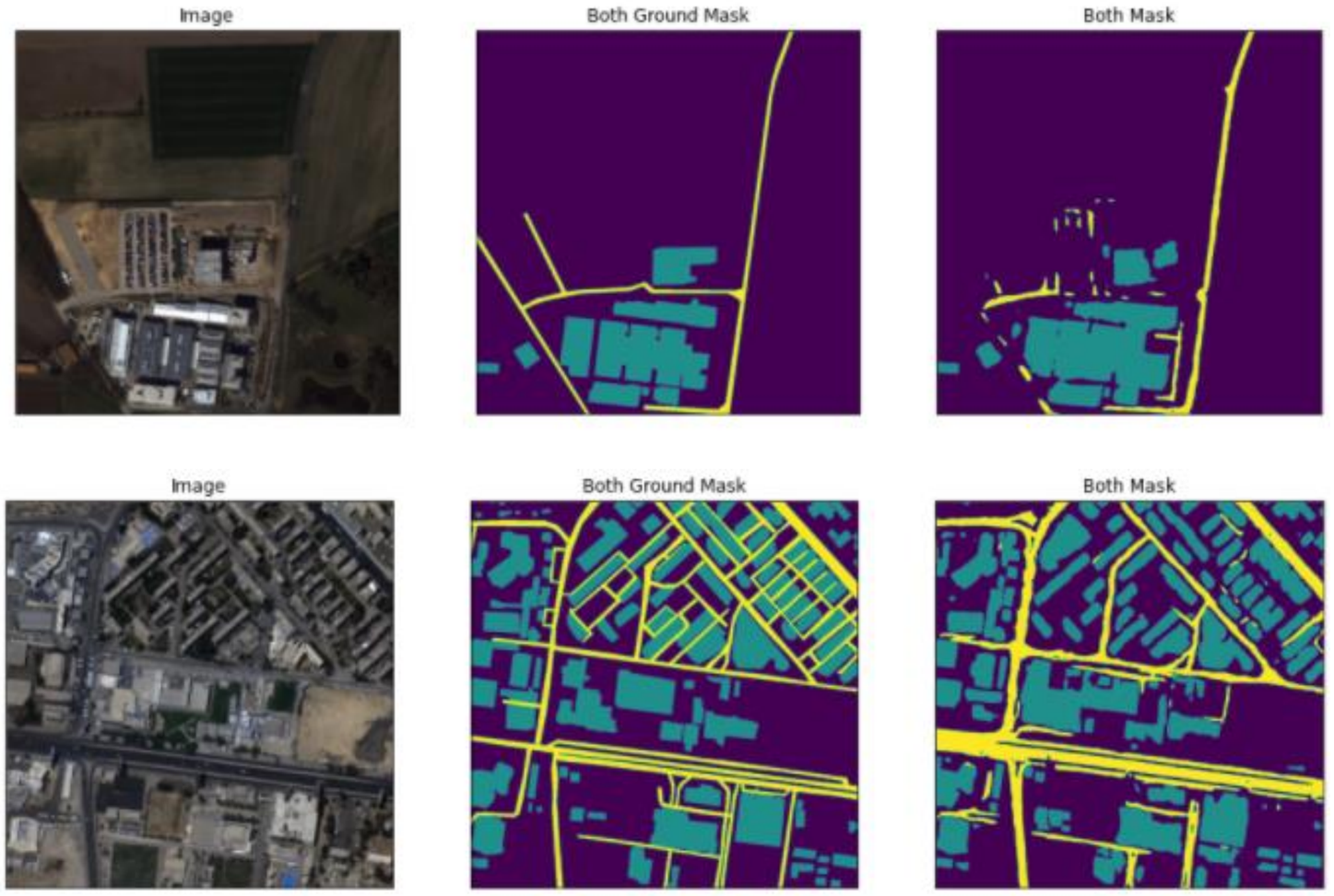


Image
Both Ground Mask
Both Mask
Image
Both Ground Mask
Both Mask

06

프로젝트 후기

프로젝트 후기

어려웠던 점 :

처음 접해보는 인공지능 개념과 OpenCV, Matplotlib 등을 활용하는데 수학적인 개념과 프로그래밍이 맞물리게 되어 더욱 어렵게 느껴졌던 것 같습니다. 또한 프로젝트를 수행하면서 많은 오류들에 맞닥뜨렸을 때 이를 해결하는 과정이 좀 힘들었습니다.

느낀 점 :

프로젝트를 통해서 여러 라이브러리를 한번에 다뤄볼 수 있는 좋은 기회였고, 이해하기 어려웠던 코드들을 한 줄 한 줄 분석하며 접근하는 과정에서 모델의 동작 원리를 이해하는데 도움이 된 좋은 경험이었습니다. 또한 프로젝트를 하면서 맞닥뜨리는 많은 오류들을 스스로 해결해 가는 과정을 통해 다양한 지식들을 얻을 수 있었고 한 단계 한 단계 씩 수행해 낼 때마다 성취감을 느낄 수 있었습니다.

배운 점 :

인공지능을 활용한 객체 분할, OpenCv와 Matplotlib 등을 활용한 데이터 시각화, 이미지 마스킹, 데이터 파싱

Thank you
감사합니다 :)