자동차 검출

01. 데이터

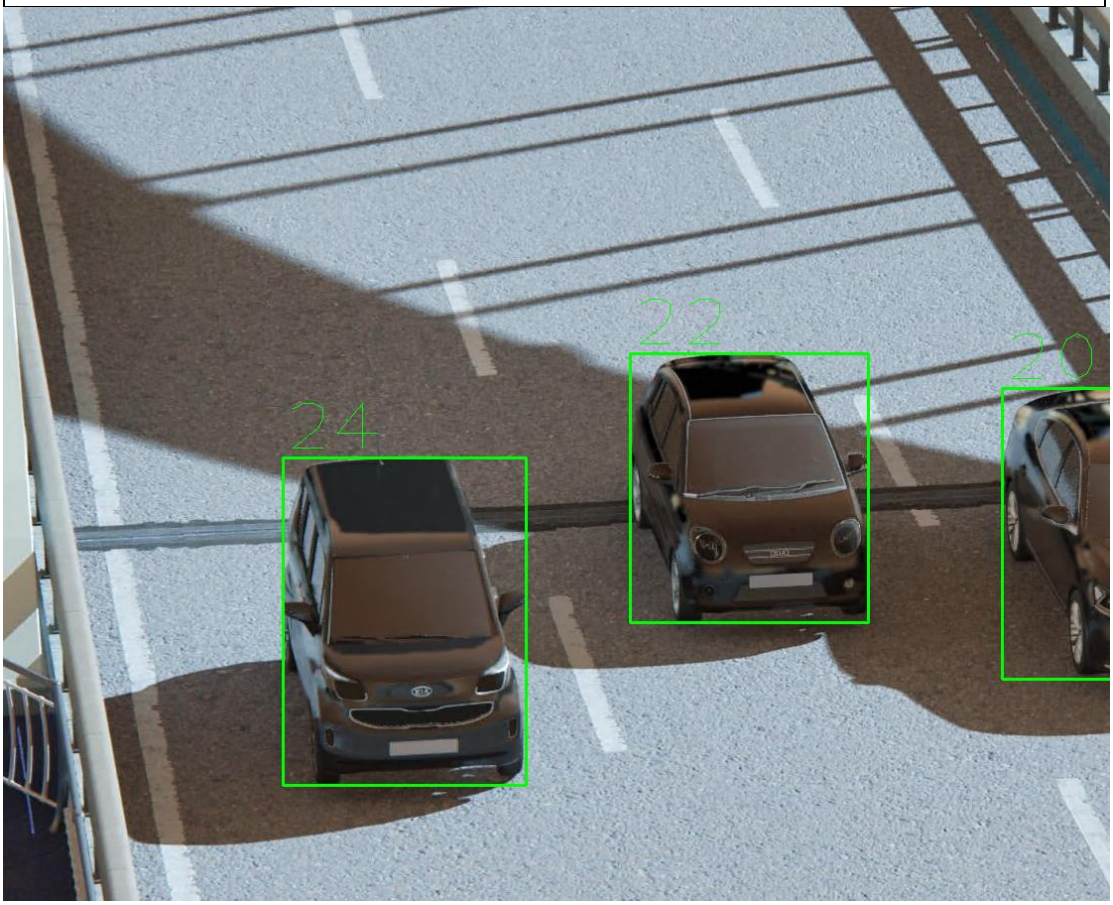
- png 파일 : Test와 train의 폴더 아래 도로에서 운전 중인 자동차 사진들, train의 경우 label txt파일이 같이 들어있다.
- Class txt 파일 : 자동차 클래스 파일

02. 이미지, bbox 시각화

이미지 하나와 txt 파일에 들어가있는 bbox정보를 이용한 시각화 코드를 짜보기

```
import   cv2
import matplotlib.pyplot as plt

def draw_boxes_on_image(image_file, annotation_file):
    # image load
    image = cv2.imread(image_file)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # txt 파일에서 정보 가져오기
    # txt read
    with open(annotation_file,'r', encoding='utf-8')as f :
        lines = f.readlines()   # 줄 단위로 끊어 저장
    for line in lines:
        #print(line)
        values = list(map(float, line.strip().split(' '))) # txt 파일에 공백
간격이어서 얘를 기준으로 나눠 리스트로 값으로 저장
        class_id = int(values[0])
        x_min, y_min = int(round(values[1])), int(round(values[2]))
        x_max, y_max = int(round(max(values[3], values[5], values[7]))),\
                    int(round(max(values[4], values[6], values[8])))

        cv2.rectangle(image, (x_min,y_min), (x_max, y_max), (0,255,0), 2)
        cv2.putText(image, str(class_id), (x_min, y_min -5),
cv2.FONT_HERSHEY_PLAIN, 5,
                    (0,255,0))

        cv2.imshow('test', image)
        if cv2.waitKey(0) & 0xFF == ord('q'):
            exit()

    '''

syn_nnnn.txt
한줄에 클래스, (x1, y1) (x2, y2) (x3, y3) (x4, y4) 이렇게 나타나있다

    '''


if __name__ == "__main__" :
    # folder path
```

```
image_file = "./dataset/train/syn_00025.png"
annotation_file = "./dataset/train/syn_00025.txt"

draw_boxes_on_image(image_file, annotation_file)
```



03. CustomDataSet

```
import torch
import cv2
import glob
import os
import numpy as np
from torch.utils.data import Dataset

# input in batch index를 model에 넘겨줄 수 있도록 하는 함수
# 배치(batch)는 딥러닝 모델에서 한 번에 처리되는 데이터의 묶음
def collate_fn(batch):
    """
    DataLoader에서 호출되는 collate_fn 함수입니다. 이 함수는 개별 데이터 샘플들의 리스트인 batch를 처리하여
    최종적으로 모델에 입력할 배치를 생성합니다.

    """
```

```
        images, targets_boxes, targets_labels = tuple(zip(*batch))
        #  tuple(zip(*batch))는 batch 리스트 안의 데이터 샘플들을 이미지, 바운딩 박스, 레이블로 나누어주는 작업

        # image list image -> torch.stack use one tensor = 0
        images = torch.stack(images, 0)
        targets= [] # [] >> targets_boxes, targets_labels

        # targets_boxes
        for i in range(len(targets_boxes)) :

            target = {
                "boxes" : targets_boxes[i],
                "labels" : targets_labels[i]
            }
            targets.append(target)

        return images, targets

#---------------------------------------------------

class CustomDataSet(Dataset):
    def __init__(self, root, train=True, transforms = None):
        self.root = root
        self.train = train
        self.transforms = transforms
        self.image_path = sorted(glob.glob(os.path.join(root, "*.png")))

        if train : # 이번 예제에서 train은 이미지와 텍스트 파일 들어있는 폴더이므로 train일 경우에만 박스정보도 가져온다
            self.boxes = sorted(glob.glob(os.path.join(root, "*.txt")))

    def parse_boxes(self, box_path):
        with open(box_path, 'r', encoding='utf-8')as f: # read box info from txt file
            lines = f.readlines()

        boxes = []
        labels = []

        for line in lines:
            values = list(map(float, line.strip().split(' ')))# 공백기준으로 txt 라인별로 정보 값 리스트에 저장
            # [2.0, 683.0, 125.0, 902.0, 125.0, 902.0, 365.0, 683.0, 365.0]
            class_id = int(values[0])
            x_min, y_min = int(round(values[1])), int(round(values[2]))
            x_max = int(round(max(values[3], values[5], values[7])))
            y_max = int(round(max(values[4], values[6], values[8])))

            boxes.append([x_min, y_min, x_max, y_max])
            labels.append(class_id)

        return torch.tensor(boxes, dtype=torch.float32), \
            torch.tensor(labels, dtype=torch.int64)                    # 박스, 라벨 텐서형태로 리턴
```

```
        # print(boxes, labels)

    def __getitem__(self, item):
        """
        데이터셋으로부터 특정 인덱스의 데이터 샘플을 가져오는 메서드입니다.

        Args:
            item (int): 가져올 데이터 샘플의 인덱스입니다.

        Returns:
            tuple: 특정 인덱스에 해당하는 데이터 샘플을 반환합니다.
                훈련 모드(train=True)인 경우에는 이미지, 바운딩 박스 정보, 레이블 정보를 반환합니다.
                검증/테스트 모드(train=False)인 경우에는 파일 이름, 이미지, 이미지의 너비와 높이 정보를 반환합니다.
        """
        # item 인덱스에 해당하는 이미지 파일의 경로를 가져옵니다.
        img_path = self.image_path[item]

        # 이미지 파일을 OpenCV를 이용하여 읽어온 후, RGB 색상 채널 순서로 변경하고, 0에서 1 사이의 값을 가지도록 정규화합니다.
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB).astype(np.float32)
        img /= 255.0

        # 이미지의 높이와 너비 정보를 가져옵니다.
        height, width = img.shape[0], img.shape[1]

        if self.train:
            # 훈련 모드(train=True)인 경우, item 인덱스에 해당하는 바운딩 박스 파일의 경로를 가져옵니다.
            box_path = self.boxes[item]

            # 바운딩 박스와 레이블 정보를 파싱하여 가져옵니다. 레이블 정보에는 1을 더하여 background를 0으로 지정합니다.
            boxes, labels = self.parse_boxes(box_path)
            labels += 1

            if self.transforms is not None:
                # transforms가 지정된 경우, 이미지와 바운딩 박스, 레이블 정보를 해당 변환에 따라서 변형시킵니다.
                transformed = self.transforms(image=img, bboxes=boxes, labels=labels)
                img, boxes = transformed['image'], transformed['bboxes']
                labels = transformed['labels']

            # 변형된 이미지, 바운딩 박스, 레이블 정보를 반환합니다.
            return img, torch.tensor(boxes, dtype=torch.float32),
```

```
            torch.tensor(labels, dtype=torch.int64)

        else:
            # 검증/테스트 모드(train=False)인 경우, 이미지를 해당 변환에
따라서 변형시킵니다.
            if self.transforms is not None:
                transformed = self.transforms(image=img)
                img = transformed['image']

            # 파일 이름, 이미지, 이미지의 너비와 높이 정보를 반환합니다.
            file_name = img_path.split('/')[-1]
            return file_name, img, width, height

    def __len__(self):
        return len(self.image_path)

if __name__ == "__main__":
    train_dataset = CustomDataSet("./dataset/train/", train=True,
transforms=None)

    for i in train_dataset:
        print(i)
# 테스트시 사용
```

04. Train 코드에서 사용할 config.py 파일을 미리 만들어둔다(config.py 파일은 모델 학습을 위해 필요한 설정(configurations)들을 저장하는 파이썬 스크립트 파일)

```
config = {
    'NUM_CLASS' : 34,
    'IMG_SIZE' : 512,
    'EPOCHS' : 10,
    'LR' : 0.0025,
    'BATCH_SIZE' : 16,
    'SEED' : 9999
}
```

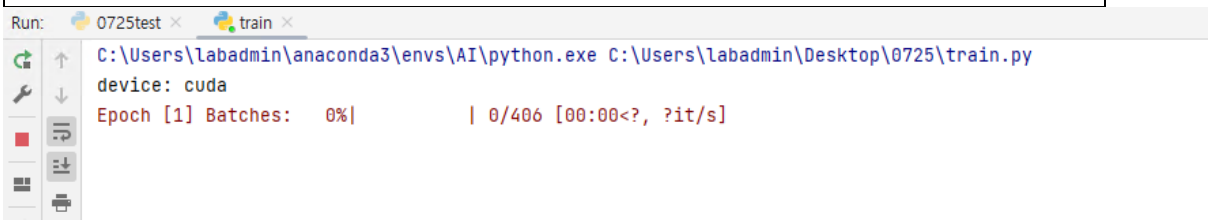
05. Train.py 코드

```
import warnings
warnings.filterwarnings(action='ignore')

import random
import numpy as np
import os
import torch
import torchvision
from torch.utils.data import DataLoader
import albumentations as A
from albumentations.pytorch.transforms import ToTensorV2
from tqdm.auto import tqdm
from CustomDataSet import CustomDataSet, collate_fn
```

```
from config import config

def main():
    device = torch.device('cuda') if torch.cuda.is_available() else
torch.device('cpu')
    print(f"device: {device}")

    # Fixed Random-seed
    def seed_everything(seed):
        random.seed(seed)
        os.environ['PYTHONHASHSEED'] = str(seed)
        np.random.seed(seed)
        torch.manual_seed(seed)
        torch.cuda.manual_seed(seed)
        torch.backends.cudnn.deterministic = True
        torch.backends.cudnn.benchmark = True

    seed_everything(config['SEED']) # Seed fix

# ----------------------------------------------------------------
    # aug
    def get_train_transforms():
        return A.Compose([
            A.Resize(config['IMG_SIZE'], config['IMG_SIZE']),
            ToTensorV2()
        ], bbox_params=A.BboxParams(format='pascal_voc',
label_fields=['labels']))

    def get_test_transforms():
        return A.Compose([
            A.Resize(config['IMG_SIZE'], config['IMG_SIZE']),
            ToTensorV2()
        ])
#----------------------------------------------------------------
    # dataset dataloader
    train_dataset = CustomDataSet("./dataset/train/", train=True,
transforms=get_train_transforms())
    test_dataset = CustomDataSet("./dataset/test/", train=False,
transforms=get_test_transforms())

    #for i in test_dataset:
    #    print(i) # 각 데이터 샘플 출력

    train_loader = DataLoader(train_dataset, batch_size=config['BATCH_SIZE'],
shuffle=True, collate_fn=collate_fn)
    test_loader = DataLoader(test_dataset, batch_size=config['BATCH_SIZE'],
shuffle=False)

#----------------------------------------------------------------
    def build_model(num_classes = config['NUM_CLASS'] + 1) : # +1은
background때문에
        model =
torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
        in_features = model.roi_heads.box_predictor.cls_score.in_features
        model.roi_heads.box_predictor =
torchvision.models.detection.faster_rcnn.FastRCNNPredictor(in_features,
num_classes)
```

```
            return model

        model = build_model()
        model.to(device)
        #print(model)

    #------------------------------------------------------------
        def train(model, train_loader, optimizer, scheduler, device,
    resume_checkpoint = None) :
            model.to(device)
            best_loss = 999999

            start_epoch = 1

            if resume_checkpoint is not None : #  사용시 경로  넣기
                checkpoint = torch.load(resume_checkpoint)
                model.load_state_dict(checkpoint['model_state_dict'])
                optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
                scheduler.load_state_dict(checkpoint['scheduler_state_dict'])
                best_loss = checkpoint['best_loss']
                state_epoch = checkpoint['epoch'] + 1
                print(f"Resuming Training from epoch {start_epoch}")

            for epoch in (range(start_epoch, config['EPOCHS']+1)):

                model.train()
                train_loss = []
                num_batches = len(train_loader)
                for batch_idx, (images, targets) in enumerate(
                            tqdm(train_loader, total=num_batches, desc=f"Epoch
    [{epoch}] Batches", leave=True, mininterval=0)):

                    images = [img.to(device) for img in images]
                    targets = [{k:v.to(device) for k, v in t.items()} for t in targets]

                    optimizer.zero_grad()

                    loss_dict = model(images, targets)
                    losses = sum(loss for loss in loss_dict.values())

                    losses.backward()
                    optimizer.step()

                    train_loss.append(losses.item())
                tr_loss = np.mean(train_loss)
                tqdm.write(f"Epoch [{epoch}] Train loss : {tr_loss:.5f}")

                if scheduler is not None:
                    scheduler.step()

                if best_loss > tr_loss :
                    best_loss = tr_loss
                    best_model = model.static_dict()
                    torch.save(best_model, './best.pt')

                # save checkpoint
                checkpoint = {
```

```
                'epoch' : epoch,
                'model_state_dict' : model.state_dict(),
                'optimizer_state_dict' : optimizer.state_dict(),
                'scheduler_state_dict' : scheduler.state_dict(),
                'best_loss' : best_loss
            }
            torch.save(checkpoint, './checkpoint.pt')


    optimizer = torch.optim.AdamW(model.parameters(), lr=config['LR'],
weight_decay=1e-2)
    scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=5)

    train(model, train_loader, optimizer, scheduler, device,
resume_checkpoint=None)




if __name__ == "__main__":
    main()
```



모델이 돌기 시작합니다.