

Deep Learning Based Vehicle Recognition Sys.

Kumoh National Institute of Technology

System Software Lab.

20200573 서준혁

SSL SEMINAR

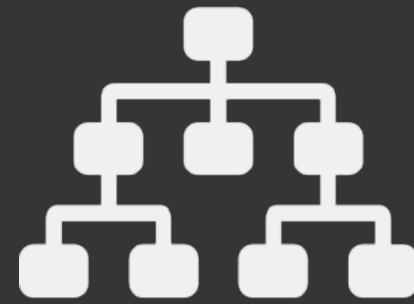
PART
01



DATASET 개요

차량 외관 데이터셋 소개

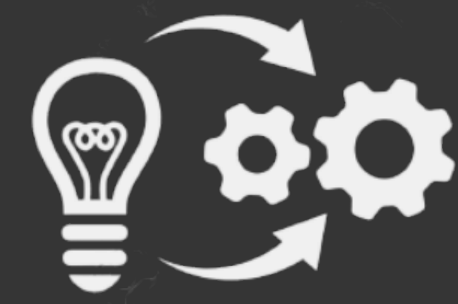
PART
02



DATASET 전처리

데이터셋 세부 내용 확인 및 전처리

PART
03



기타 사항

향후 계획 및 고민

PART
01

Dataset

차량 외관 데이터셋 개요

차량 외관 영상 데이터셋

데이터셋 준비를 위한 방법을 찾던 중, 국내 AI 정보 플랫폼 AI-HUB에서 차량 외관 영상을 제공하는 것을 발견했다.



구축년도는 2021년 말, 최종 갱신년월은 2022년 8월로 상당히 최근 데이터임을 확인했으며

다운로드 수가 현저히 적어 아직 널리 사용된 데이터셋은 아님을 확인했다.

용량은 총 79.73GB이며 차량 100종에 대한 트림, 색상별

영상 데이터 약 110만장과 JSON 타입의 라벨링 데이터로 이루어져있다.

차량 외관 영상 데이터셋

데이터는 차량 외관 데이터 중에서도, 차량 전체 BOUNDING BOX와 14개 부품의 BOUNDING BOX를 제공한다.



그 외에도 날씨, 촬영 장소, 일자, 촬영 FPS등 다양한 정보가 포함되어있다.

본 주제에서는 차량 전체 외관 BOX 데이터를 제외한 데이터는 필요하지 않기 때문에 나머지 데이터는 우선은 제외시키기로 한다.

1. 데이터 소개

- 차량 외관(차종, 연식, 색상, 트림)과 14개 파트(프론트범퍼, 리어범퍼, 타이어, A필러, C필러, 사이드미러, 앞도어, 뒷도어, 라디에이터그릴, 헤드램프, 리어램프, 보닛, 트렁크, 루프)를 식별할 수 있는 AI 학습용 데이터셋. 차량 외관과 색상 그리고 번호판 인식만 가능했던 기존 데이터셋을 보완하여 다양한 차종과 트림, 색상, 파트 등 세세한 속성을 정의한 데이터

2. 대표도면

구분	원천데이터	라벨링데이터
유형		
형식	jpg	json

차량 외관 영상 데이터셋

전체 용량은 80GB로, 다운로드에 약 1시간 30분정도 소요된다.

YOLOV5 모델이 있는 ROOT 디렉터리에 다운로드를 시작했으며

다운로드가 되는 동안 샘플 데이터를 이용해 PARSE를 미리 만들자.

18% - AI 학습용 데이터 다운로드



18% (14GB/80GB)

1 hour 8 min 25 sec (131Mbps)

Downloading



Pause

C:\Users\ssam2\Desktop\yolov5

INNORIX

PART
02

Pre-Processing

데이터셋 전처리 구현

샘플 데이터

앞서 본 것처럼, 다운받은 데이터셋에는 우리가 필요한 차량 전체 외관 바운딩 박스를 제외하고

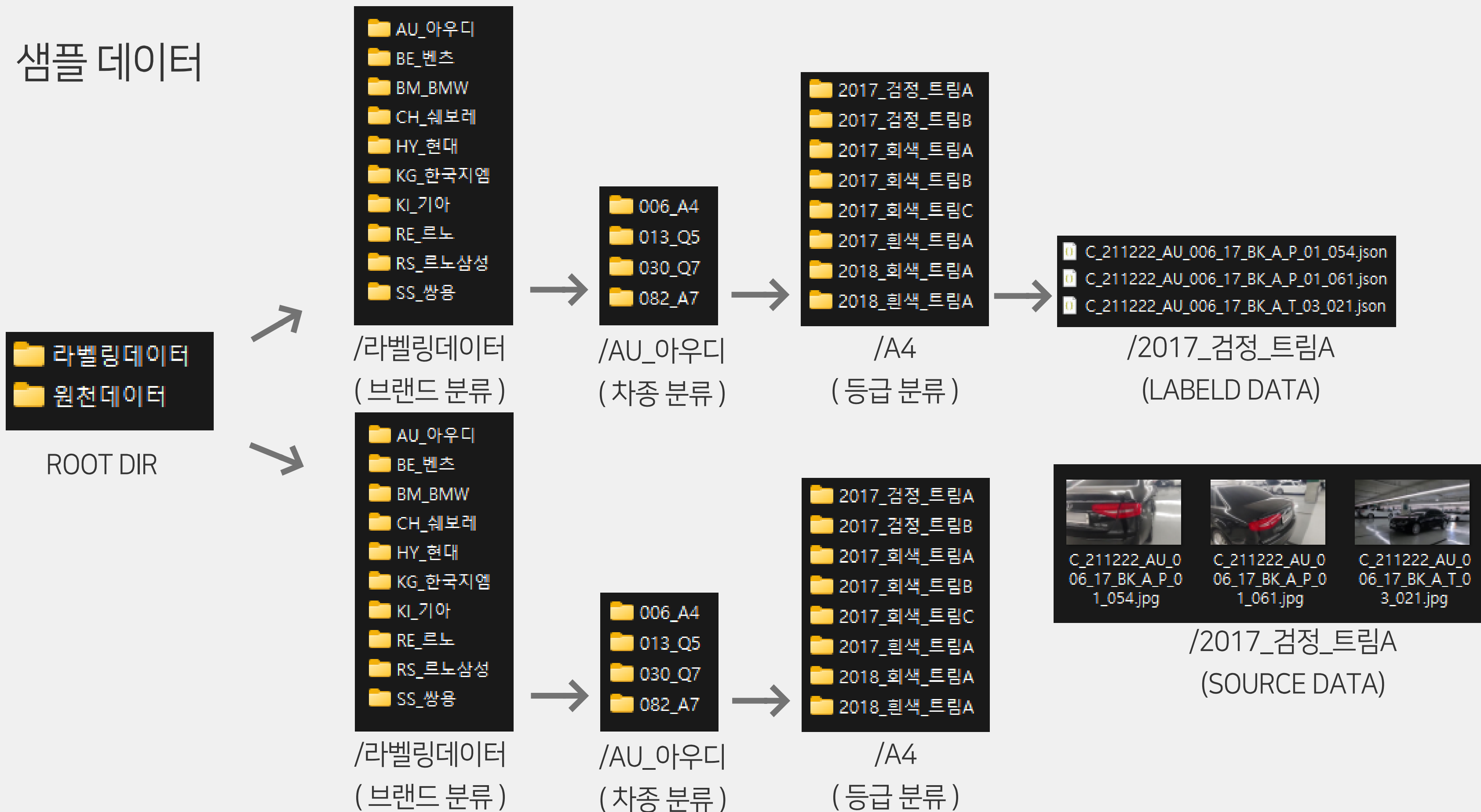
다른 필요없는 정보들이 꽤 많다.

80GB의 파일들을 직접 하나하나 수정하기에는 시간이 꽤나 오래 걸리기 때문에

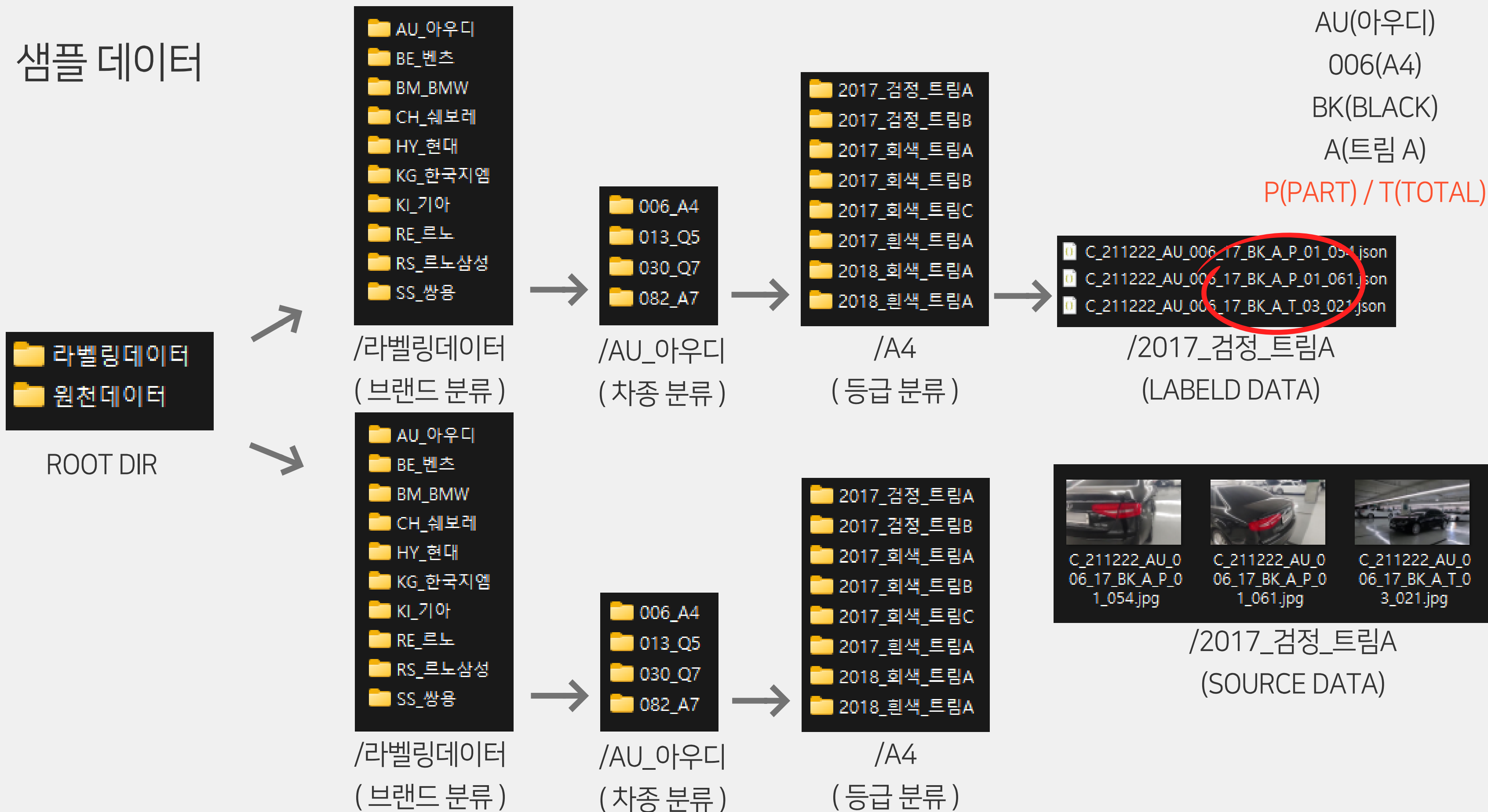
전체 데이터셋 가공에 앞서 500MB 정도의 가벼운 샘플 데이터를 가지고

데이터셋 자동 전처리 프로그램을 미리 만들어놓아보자.

샘플 데이터



샘플 데이터



```
{
  "rawDataInfo": {
    "rawDataID": "C_211111_SS_097_21_RE_B_T_03",
    "copyrighter": "(주)미디어그룹사람과숲",
    "resolution": "1920*1080",
    "date": "2021-11-11",
    "StartTime": "17:00:00",
    "EndTime": "17:15:00",
    "length": 109,
    "Local": "시화",
    "season": "Autumn",
    "weather": "Cloudy",
    "precip": 0,
    "temp": 7,
    "fps": 30,
    "fStop": "F/13",
    "exposureTime": "1/60",
    "ISO": 200,
    "LargeCategoryId": "소형차",
    "MediumCategoryId": "쌍용",
    "SmallCategoryId": "티볼리",
    "yearId": 2021,
    "colorId": "빨강",
    "trimId": "B",
    "fileExtension": "mp4"
  },
  "sourceDataInfo": {
    "sourceDataID": "C_211111_SS_097_21_RE_B_T_03_013",
    "fileExtension": "jpg"
  },
  "learningDataInfo": {
    "path": null,
    "LearningDataId": "C_211111_SS_097_21_RE_B_T_03_013",
    "fileExtension": "json",

```

```
"classId": "P00.차량전체",
"annotation": "bbox",
"coords": {
  "tl": {
    "x": 667.8818213868292,
    "y": 350.6399999999996
  },
  "tr": {
    "x": 1176.6287348704532,
    "y": 350.6399999999996
  },
  "bl": {
    "x": 667.8818213868292,
    "y": 836.2599999999995
  },
  "br": {
    "x": 1176.6287348704532,
    "y": 836.2599999999995
  }
},
"left": 667.8818213868292,
"top": 350.6399999999996,
"width": 508.74691348362404,
"height": 485.62,
"angle": 0

```

샘플 데이터셋의 JSON 파일 내부이다.

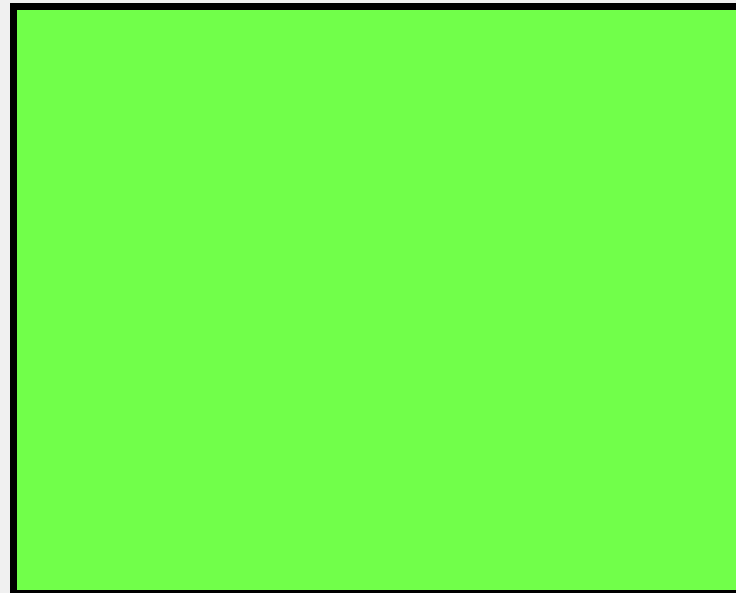
저작권자, 날씨, 노출시간, 차종, 브랜드 등의 다양한 정보를 포함하고 있다.

또한, T 타입의 파일이기 때문에 차량 전체 좌표를 포함하고 있는 것을 확인할 수 있다.

본 주제에서는 동그라미친 부분의 데이터만 필요하기에 해당 부분만 뽑아낼 수 있도록 파서를 구현했다.

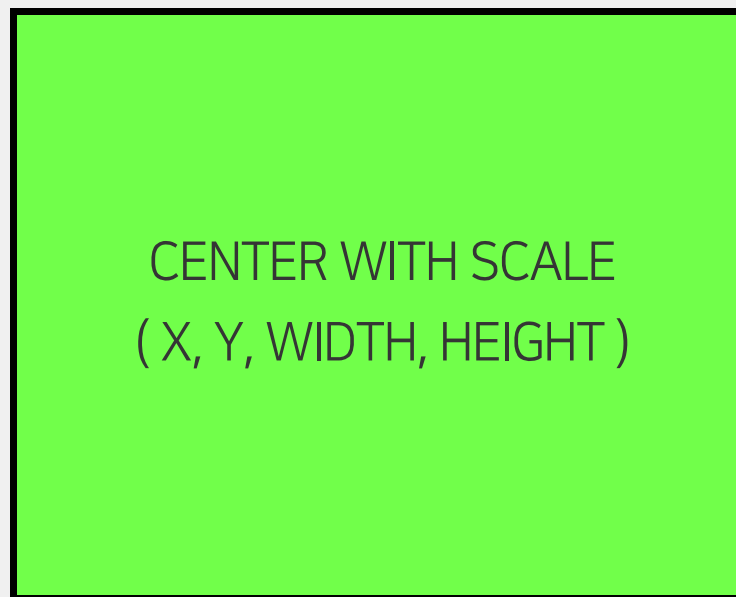
이름	수정된 날짜	유형	파일	편집	보기
<input checked="" type="checkbox"/> 1.txt	2022-09-02 오전 3:55	텍스트 문서	0 0.508236 0.565245 0.811047 0.608527		
2.txt	2022-09-02 오전 3:55	텍스트 문서			
3.txt	2022-09-02 오전 3:55	텍스트 문서			
4.txt	2022-09-02 오전 3:55	텍스트 문서			
5.txt	2022-09-02 오전 3:55	텍스트 문서			
6.txt	2022-09-02 오전 3:55	텍스트 문서			
7.txt	2022-09-02 오전 3:55	텍스트 문서			
8.txt	2022-09-02 오전 3:55	텍스트 문서			
9.txt	2022-09-02 오전 3:55	텍스트 문서			
10.txt	2022-09-02 오전 3:55	텍스트 문서			
11.txt	2022-09-02 오전 3:55	텍스트 문서			
12.txt	2022-09-02 오전 3:55	텍스트 문서			
13.txt	2022-09-02 오전 3:55	텍스트 문서			
14.txt	2022-09-02 오전 3:55	텍스트 문서			
15.txt	2022-09-02 오전 3:55	텍스트 문서			
16.txt	2022-09-02 오전 3:55	텍스트 문서			
17.txt	2022-09-02 오전 3:55	텍스트 문서			
18.txt	2022-09-02 오전 3:55	텍스트 문서			
19.txt	2022-09-02 오전 3:55	텍스트 문서			
20.txt	2022-09-02 오전 3:55	텍스트 문서			
21.txt	2022-09-02 오전 3:55	텍스트 문서			

LEFT TOP
(XMIN, YMIN)



일반 BOUNDING BOX

RIGHT BOTTOM
(XMAX, YMAX)



CENTER WITH SCALE
(X, Y, WIDTH, HEIGHT)

YOLO FORMAT

```
# Convert Data to Yolo Format - Fin
def toYOL0(size, box):
    dw = 1./size[0]
    dh = 1./size[1]

    x = ( box[0] + box[1] )/2.0
    y = ( box[2] + box[3] )/2.0
    w = box[1] - box[0]
    h = box[3] - box[2]

    x = x * dw
    w = w * dw
    y = y * dh
    h = h * dh

    return (x, y, w, h)
```

왼쪽 상단 좌표와 오른쪽 하단 좌표를 사용해
표현하는 BOUNDING BOX 스타일에서

중앙 좌표값과 너비, 높이를 사용해 표현하는
YOLO 스타일로 좌표를 변환하는 함수이다.

```
# Choose JSON Data and Return Text
# Not Finished
def J2Text(item, json_data):
    scale_X = 1920
    scale_Y = 1080
    tl_X = item["coords"]["tl"]["x"] # MinX
    tl_Y = item["coords"]["tl"]["y"] # MinY
    br_X = item["coords"]["br"]["x"] # MaxX
    br_Y = item["coords"]["br"]["y"] # MaxY

    box = (tl_X, br_X, tl_Y, br_Y)
    size = (scale_X, scale_Y)

    x, y, width, height = toYOLO(size, box)

    name = json_data["rawDataInfo"]["SmallCategoryId"]
    model = getClass(name)

    labels = str(str(model[2]) + " " + str(x) + " " + str(y) + " " + str(width) + " " + str(height))

    return labels
```

```
['BMW', '3SERIES', 0]
0 0.47421874999999997 0.49583333333333335 0.6171875 0.4305555555555556
```

데이터셋의 영상은 스케일(크기)이
모두 1920*1080이었기 때문에 픽스하고,

JSON 파일안에 있는 TOP LEFT의 X, Y값과
BOTTOM RIGHT의 X, Y 값을 변환 함수로 전달한다.

그대로 리턴받아 파일 출력에 사용할 수 있도록
YOLO STYLE에 맞춰 리턴하도록 구현한 결과,

차량 모델의 분류 클래스 인덱스를 포함한
YOLO 라벨값들이 잘 출력되는 것을 확인할 수 있었다.

* GETCLASS 함수는 추후 설명

```
# Initialize Destination Directory to Save Pre-Processed Data
# Finished
def destInit(dest):
    if (not os.path.isdir(dest)):
        mode = 0o777
        os.mkdir(destDir, mode)

        destPath = os.path.join(destDir, "train")
        imagePath = os.path.join(destPath, "images")
        labelPath = os.path.join(destPath, "labels")

        os.mkdir(destPath, mode)
        os.mkdir(imagePath, mode)
        os.mkdir(labelPath, mode)
```

정리한 데이터를 담은 목적지 디렉터리를 초기화한다.

TRAIN을 폴더를 만들고, 해당 경로 안에 IMAGES와 LABELS 폴더를 만들도록 작성한다.

디렉터리가 생성된 것이 확인되면 하위 디렉터리를 만들도록 MODE를 설정해줬다.

내 PC > 바탕 화면 > JSONParser > dataset > train

이름

수정한 날짜

images

2022-09-19 오후 4:25

labels

2022-09-19 오후 4:25

해당 코드를 실행한 결과이다.

하위 폴더들이 경로에 맞게 잘 생성된 것을 확인할 수 있다.

```
10 # Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
11 path: ../datasets/coco # dataset root dir
12 train: train2017.txt # train images (relative to 'path') 118287 images
13 val: val2017.txt # val images (relative to 'path') 5000 images
14 test: test-dev2017.txt # 20288 of 40670 images, submit to https://competitions.codalab.org/competitions/20794
15
16 # Classes
17 names:
18 0: person
19 1: bicycle
20 2: car
21 3: motorcycle
22 4: airplane
23 5: bus
24 6: train
25 7: truck
26 8: boat
27 9: traffic light
28 10: fire hydrant
29 11: stop sign
30 12: parking meter
31 13: bench
32 14: bird
33 15: cat
34 16: dog
35 17: horse
36 18: sheep
37 19: cow
38 20: elephant
39 21: bear
40 22: zebra
```

저번 시간에 잠깐 봤듯이, 여러 종류의 객체를 인식하고 싶다면

클래스의 이름과 함께 해당 클래스를 뜻하는 인덱스를 지정해주어야한다.


```
def getClass(name):
    models = {
        "3시리즈" : ["BMW", "3SERIES"],
        "5시리즈" : ["BMW", "5SERIES"],
        "7시리즈" : ["BMW", "7SERIES"],
        "X3" : ["BMW", "X3"],
        "X5" : ["BMW", "X5"],

        "알티마" : ["NISSAN", "ALTIMA"],

        "디스커버리" : ["LANDROVER", "DISCOVERY"],
        "레이노버" : ["LANDROVER", "RANGEROVER"],

        "ES" : ["LEXUS", "ES"],

        "캡처" : ["RENAULT", "CAPTURE"],
        "클리오" : ["RENAULT", "CLIO"],

        "Countryman" : ["MINI", "COUNTRYMAN"],
        "클럽맨" : ["MINI", "CLUBMAN"],

        "A-Class" : ["BENZ", "ACLASS"],
        "C-Class" : ["BENZ", "CCLASS"],
        "CLA-Class" : ["BENZ", "CLACCLASS"],
        "CLS-Class" : ["BENZ", "CLSCLASS"],
        "E-Class" : ["BENZ", "ECLASS"],
        "GLA-Class" : ["BENZ", "GLACCLASS"],
        "GLC-Class" : ["BENZ", "GLCCCLASS"],
        "GLE-Class" : ["BENZ", "GLECLASS"],
        "S-Class" : ["BENZ", "SCLASS"],

        "S90" : ["VOLVO", "S90"],
        "XC60" : ["VOLVO", "XC60"],

        "A4" : ["AUDI", "A4"],
        "A6" : ["AUDI", "A6"],
        "Q3" : ["AUDI", "Q3"],
        "Q7" : ["AUDI", "Q7"],
        "A7" : ["AUDI", "A7"],

        "랑글러" : ["JEEP", "Rangler"],

        "칼리" : ["TOYOTA", "CAMLEE"],
        "프리우스" : ["TOYOTA", "PRIUS"],

        "익스플로러" : ["FORD", "EXPLORER"],

        "아티온" : ["VOLKSWAGEN", "ARTHEON"],
        "골프" : ["VOLKSWAGEN", "GOLF"],
        "파사트" : ["VOLKSWAGEN", "PASSAT"],
        "티구안" : ["VOLKSWAGEN", "TIGUAN"],

        "CR-V" : ["HONDA", "CR-V"],
        "어코드" : ["HONDA", "ACCORD"],

        "K3" : ["KIA", "K3"],
        "K5" : ["KIA", "K5"],
        "K7" : ["KIA", "K7"],
        "K9" : ["KIA", "K9"],
        "니로" : ["KIA", "NIRO"],
        "레이" : ["KIA", "RAY"],
        "모닝" : ["KIA", "MORNING"],
        "모하비" : ["KIA", "MOHAVE"],
        "봉고3" : ["KIA", "BONGO3"],
        "셀토스" : ["KIA", "SELTO5"],
        "스토닉" : ["KIA", "STONIC"],
        "스텔러" : ["KIA", "STINGER"],
        "스포티지" : ["KIA", "SPORTAGE"],
        "쏘렌토" : ["KIA", "SORENTO"],
        "쏘울" : ["KIA", "SOUL"],
        "카니발" : ["KIA", "CARNIVAL"],
```

```
index = 0
for model in models:
    if (model == name):
        result = models[model]
        break
    index += 1

result.append(index)

return result
```

해당 함수로 차량의 이름이 전달되면, 각 브랜드의 리스트를 돌면서 어떤 브랜드의 차량인지 클래스와 인덱스를 구하도록 작성했다.

```
"LargeCategoryId": "소형차",
"MediumCategoryId": "상용",
"SmallCategoryId": "티볼리",
```

데이터셋에서 브랜드명과 모델명이 한글로 구성되어있는데, 코드에 한글이 포함될시에 추후 오류가 발생할 여지가 있어 입력받은 한글 모델명을 토대로 영문 브랜드명과 영문 모델명, 그리고 클래스 인덱스를 세서 리턴하도록 작성했다.

* SSANGYONG TIVOLI 56

핵심 함수인 READJSON이다.

매개변수로 최상위 경로를 입력받아 GLOB 라이브러리를 이용해

하위 디렉터리의 모든 파일들을 재귀적으로 읽을 수 있도록했다.

파일 확장명이 .JSON 또는 .JPG일 경우, 앞서 보여준 함수들에

전달해 가공하여 각각 라벨과 이미지 데이터로 저장 디렉터리에

저장하도록 구현했다.

```
# Read JSON Type Files Recursively
# Not Finished
def readJson(dir):
    labelCount = 0
    imageCount = 0

    for filename in glob.iglob(dir + "**/*", recursive=True):
        if ("_T_" in filename and filename.endswith(".json")):
            f = open(filename, 'rt', encoding='UTF8')
            json_data = json.load(f)

            for item in json_data["learningDataInfo"]["objects"]:
                if item["classId"] == "P00.차량전체":
                    data = J2Text(item, json_data)
                    title = filename
                    writeTxt(data, title)
                    labelCount += 1

            elif ("_T_" in filename and filename.endswith(".jpg")):
                image_read = np.fromfile(filename, np.uint8)
                image = cv2.imdecode(image_read, cv2.IMREAD_COLOR)
                title = filename
                writeImg(image, title)
                imageCount += 1

    print("Processed Image :", imageCount)
    print("Processed Labels :", labelCount)
```

실제로 라벨 데이터와 영상 데이터를 저장하는 함수들이다.

각 함수 모두 공통적으로 현재 디렉리를 CURPATH 변수에 잠깐 저장하고

OS 모듈을 이용해 일시적으로 디렉터리를 옮긴 뒤,

각각 TXT 파일과 JPG 파일을 저장하고 돌아오는 방식으로 구현했다.

이미지 파일의 경우 일반적인 방식으로는 불러올 수 없기 때문에

OPENCV와 NUMPY를 이용하여 읽고 쓰는 과정을 거쳤다.

```
# Write Labels in dataset/train/labels Directory
def writeTxt(data, title):
    curPath = os.getcwd()

    os.chdir("./dataset/train/labels")

    title = title.split("\\")[6]

    f = open(title, "w")
    f.write(data)
    #print("Processing", title)

    os.chdir(curPath)

# Copy Images in dataset/train/images Directory
def writeImg(img, title):
    curPath = os.getcwd()

    os.chdir("./dataset/train/images")

    title = title.split("\\")[6]

    cv2.imwrite(title, img)
    #print("Processing", title)

    os.chdir(curPath)
```

해당 프로그램을 구동한 결과이다.

DATASET > TRAIN > IMAGES 폴더와
DATASET > TRAIN > LABELS 폴더에

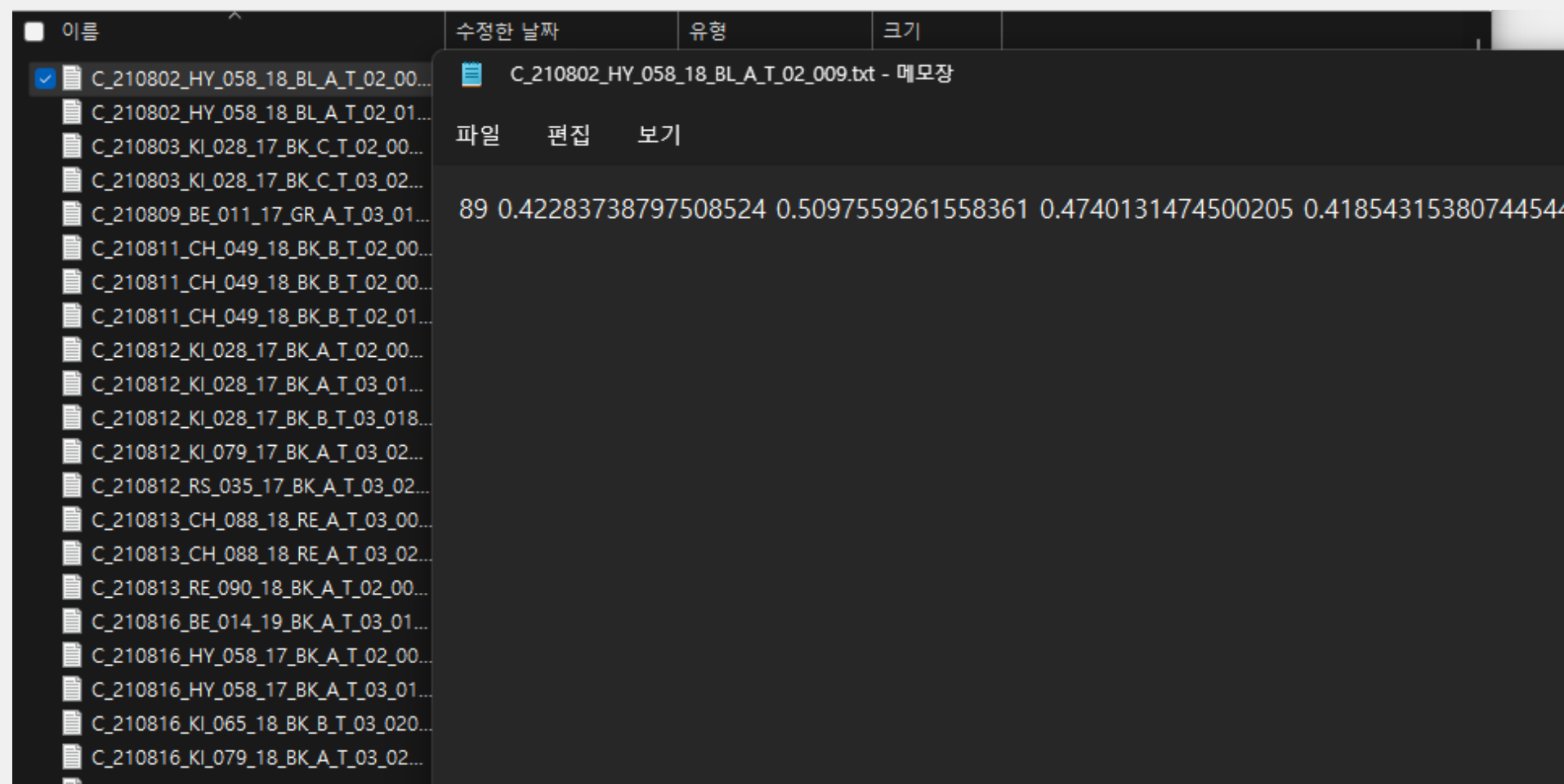
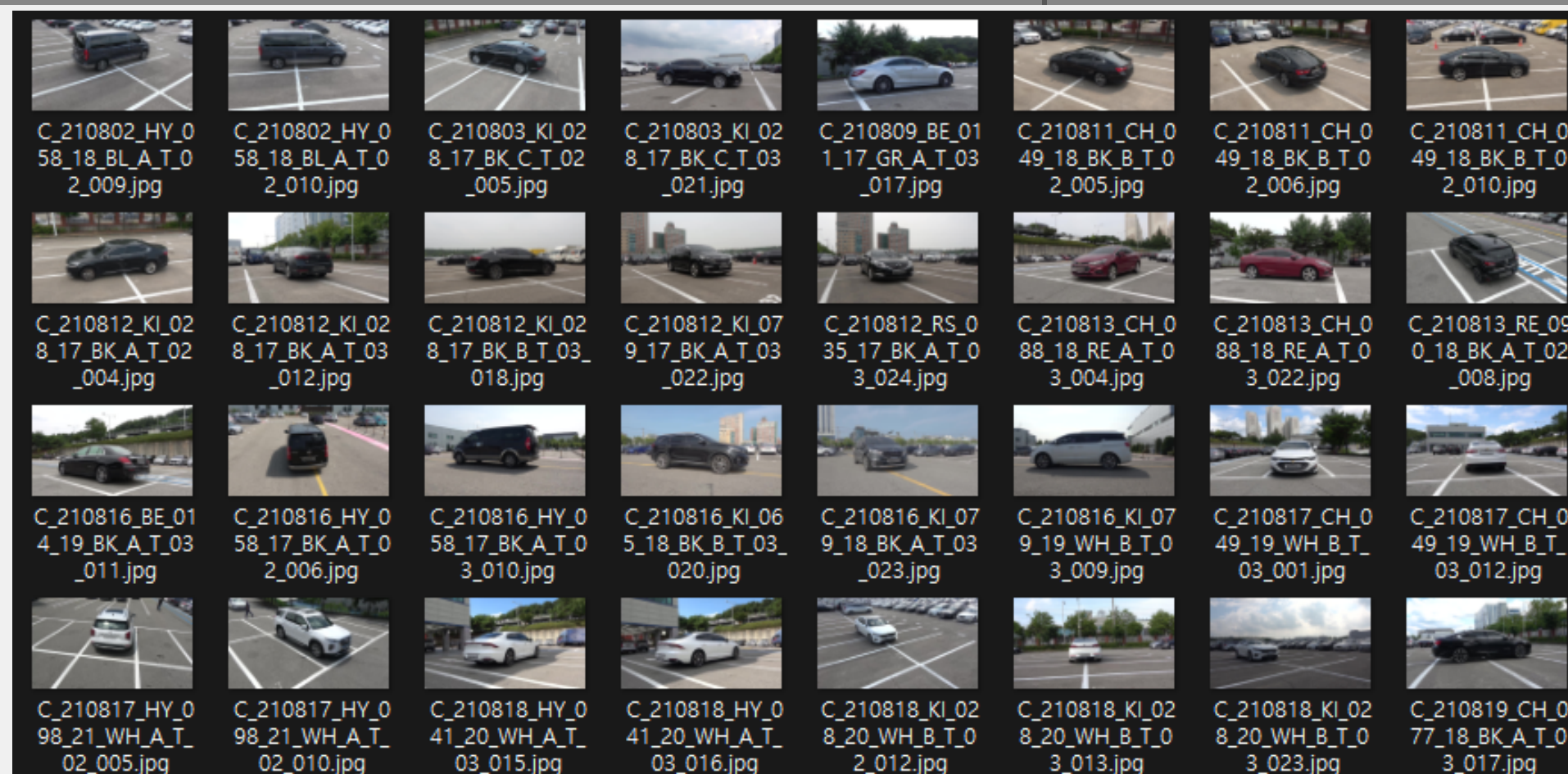
다음과 같이 차량 전체 외관 영상과 라벨 데이터가

잘 처리되어 저장되는 것을 확인할 수 있었다.

Processed Image : 647
Processed Labels : 644

처리한 영상 파일과 라벨을 카운트한 결과이다.

약간의 오차가 생기는 걸로 보아 원인을 찾을 필요가 있어 보인다.



느낀 점

전체 데이터셋을 다운받는 동안 노트북이 잠기면 다운로드가

아예 취소되버려서 곤란하기도 했지만, 80GB의 데이터를 모두 준비했다.

앞서 말했던 문제점이 왜 발생했는지 해결한 뒤에,

실 데이터셋에 적용시켜 처리가 잘 되는지 빠르게 확인해보고

어서 모델 학습에 적용시켜 볼 수 있으면 좋겠다.



091.차량 외관 영상 데이터

종류:	파일 폴더
위치:	C:\Users\ssam2\Desktop\yolov5
크기:	79.6GB (85,575,134,559 바이트)
디스크 할당 크기:	79.6GB (85,575,163,904 바이트)
내용:	파일 16, 폴더 7

Deep Learning Based Vehicle Recognition Sys.

Contact

발표자 | 서준혁

TEL. 010-9802-5053

E-mail ssam2s@naver.com