오픈소스SW 프로젝트 최종보고서

조선대학교 CHOSUN UNIVERSITY

과목명	오픈소스SW 프로젝트
담당교수	나인섭 교수님
학과	컴퓨터공학과
학년	4학년
학번	20195106
이름	김근영
제출일	2023.01.25

목차

조선대학교 CHOSUN UNIVERSITY

1. 프로젝트 주제

▷ 시각 장애인을 위한 음료수 캔 이미지 인식 기반 알림 서비스

현재 음료수 캔의 점자

이 주제를 선택한 계기로는 시각 장애인 유튜버의 음료수 캔 구분 영상을 보고, 시각 장애인을 위한 음료수 캔 종류를 구분해주고 알려주는 서비스를 만들면 좋겠다고 생각하여 선택하게 되었습니다.

위의 사진을 보시다시피 현재 음료수 캔의 표시되어있는 점자로 시각 장애인들이 얻을 수 있는 정보는 탄산과 음료, 2가지의 정보밖에 없습니다. 따라서 저는 음료수 캔의 이미지를 인식하여 종류를 구분하고, 이것을 바탕으로 음성으로 알려주는 서비스를 목표로 프로젝트를 진행하였습니다.

2. 기존 소스코드 설명

▷ 기존(참고) 소스 주소 및 라이센스

- 참고 사이트 : Git Hub
- 기존 소스 주소 : https://github.com/taeminHan/What-sYourMerchandise
- 기존 소스의 라이센스 : 라이센스 미표기
- 기존 소스 설명 : 시각 장애인을 위한 이미지 인식 기반 음성 상품 정보 출력 프로젝트

▷ 기존소스의 프로그램 흐름(자세한 분석은 깃허브에 올려놓았습니다.)

- 기존 소스는 이미지 데이터로 모델을 학습시켜 이미지를 분류해주는 코드
- 데이터의 경로 및 클래스 지정

```
caltech_dir = 'C:/Users/taemin/PycharmProjects/What-sYourMerchandise/DataSet/Train'
categories = ["Cocacola", "sevenstar", "sprite"]
nb_classes = len(categories)
```

- 이미지 전처리 후 배열로 저장

```
for idx, cat in enumerate(categories):

    # one-hot 돌리기.
    label = [0 for i in range(nb_classes)]
    label[idx] = 1

    image_dir = caltech_dir + "/" + cat
    files = glob.glob(image_dir + "/*.jpg")
    print(cat, " 파일 길이 : ", len(files))
    for i, f in enumerate(files):
        img = Image.open(f)
        img = img.convert("RGB")
        img = img.resize((image_w, image_h))
        data = np.asarray(img)

        X.append(data)
        y.append(label)

        if i % 700 == 0:
            print(cat, " : ", f)

X = np.array(X)
y = np.array(y)
```

- 데이터 셋을 훈련 데이터와 테스트 데이터로 분할 후 npy 파일로 저장
- 저장한 npy 파일을 불러오기

```
X_train, X_test, y_train, y_test = train_test_split(X, y)
xy = (X_train, X_test, y_train, y_test)
np.save("C:/Users/taemin/PycharmProjects/What-sYourMerchandise/multi_image_data.npy", xy)

print("ok", len(y))
```

```
X_train, X_test, y_train, y_test = np.load('C:/Users/taemin/PycharmProjects/What-sYourMerchandise/multi_image_data.npy')
print(X_train.shape)
print(X_train.shape[0])
```

- 순차적으로 모델 쌓기
- 합성곱 신경망 구성

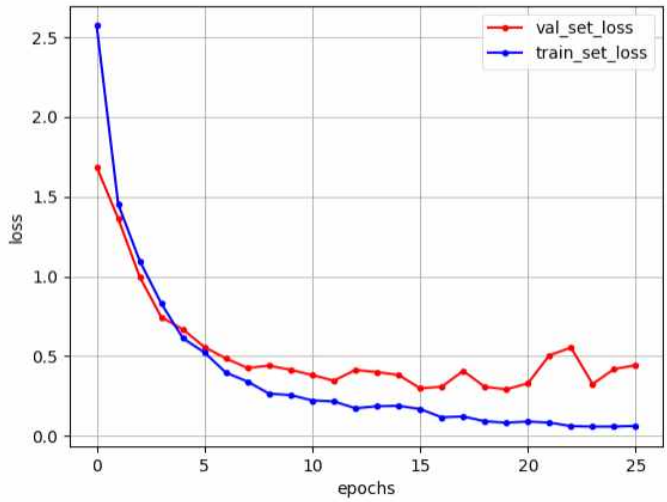
```
#일반화
X_train = X_train.astype(float) / 255
X_test = X_test.astype(float) / 255

model = Sequential()
model.add(Conv2D(32, (3,3), padding="same", input_shape=X_train.shape[1:], activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model_dir = './model'
```

- 모델을 저장할 폴더가 없으면 만들기
- 모델 경로 지정 및 콜백 함수 설정

```
if not os.path.exists(model_dir):
    os.mkdir(model_dir)

model_path = model_dir + '/multi_img_classification.model'
checkpoint = ModelCheckpoint(filepath=model_path , monitor='val_loss', verbose=1, save_best_only=True)
early_stopping = EarlyStopping(monitor='val_loss', patience=6)

model.summary()
```

- 모델 실행 및 오차 저장
- 오차 그래프로 시각화

```
history = model.fit(X_train, y_train, batch_size=32, epochs=50, validation_data=(X_test, y_test), callbacks=[checkpoint, early_stopping])

print("정확도 : %.4f" % (model.evaluate(X_test, y_test)[1]))

y_vloss = history.history['val_loss']
y_loss = history.history['loss']

x_len = np.arange(len(y_loss))

plt.plot(x_len, y_vloss, marker='.', c='red', label='val_set_loss')
plt.plot(x_len, y_loss, marker='.', c='blue', label='train_set_oss')
plt.legend()
plt.xlabel('epochs')
plt.ylabel('loss')
plt.grid()
plt.show()
```



· 테스트 데이터 예측 후 출력

```
prediction = model.predict(X)
np.set_printoptions(formatter={'float': lambda x: "{0:0.3f}".format(x)})
cnt = 0


for i in prediction:
    pre_ans = i.argmax()  # 예측 레이블
    print(i)
    print(pre_ans)
    pre_ans_str = ''
    if pre_ans == 0: pre_ans_str = "코카콜라"
    elif pre_ans == 1: pre_ans_str = "칠성사이다"
    elif pre_ans == 2: pre_ans_str = "스프라이트"
    else: pre_ans_str = "게"
    if i[0] >= 0.8 : print("해당 "+filenames[cnt].split("\\")[1]+"이미지는 "+pre_ans_str+"로 추정됩니다.")
    if i[1] >= 0.8: print("해당 "+filenames[cnt].split("\\")[1]+"이미지는 "+pre_ans_str+"으로 추정됩니다.")
    if i[2] >= 0.8: print("해당 "+filenames[cnt].split("\\")[1]+"이미지는 "+pre_ans_str+"로 추정됩니다.")
    cnt += 1
```

· 프로그램 결과 : 정확도는 대략 0.85 이상, 이미지 분류도 잘됨

```
정확도 : 0.8777
3/3 [==============================] - 0s 19ms/step
[1.000 0.000 0.000 0.000 0.000 0.000]
0
해당 coca_T1.jpg이미지는 코카콜라로 추정됩니다.
[1.000 0.000 0.000 0.000 0.000 0.000]
0
해당 coca_T10.jpg이미지는 코카콜라로 추정됩니다.
[1.000 0.000 0.000 0.000 0.000 0.000]
0
해당 coca_T11.jpg이미지는 코카콜라로 추정됩니다.
[1.000 0.000 0.000 0.000 0.000 0.000]
0
```

3. 나의 소스코드 설명

▷ 깃허브 주소 및 라이센스

· 내 소스의 주소 : https://github.com/rmsdud0363/2023_OSSP

· 기존 소스의 라이센스 : MIT License

· 나의 소스 설명 : 시각 장애인을 위한 이미지 인식 기반 음성 상품 정보 출력 프로젝트 소스에 실시간 이미지 인식 코드 추가

▷ 기존 소스와 차별점

· 크롤링 코드 추가

· 구글 이미지를 가져오는 크롤링 코드를 작성하여 이미지 데이터 구축

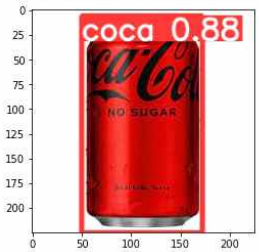
· 이미지를 저장하는 코드를 완벽하게 구성하지 못하여 원하는 폴더에 저장하지 못함, 일일이 코드와 크롬 드라이버를 저장하고 싶은 폴더로 옮겨 크롤링을 진행한 부분이 아쉽다.

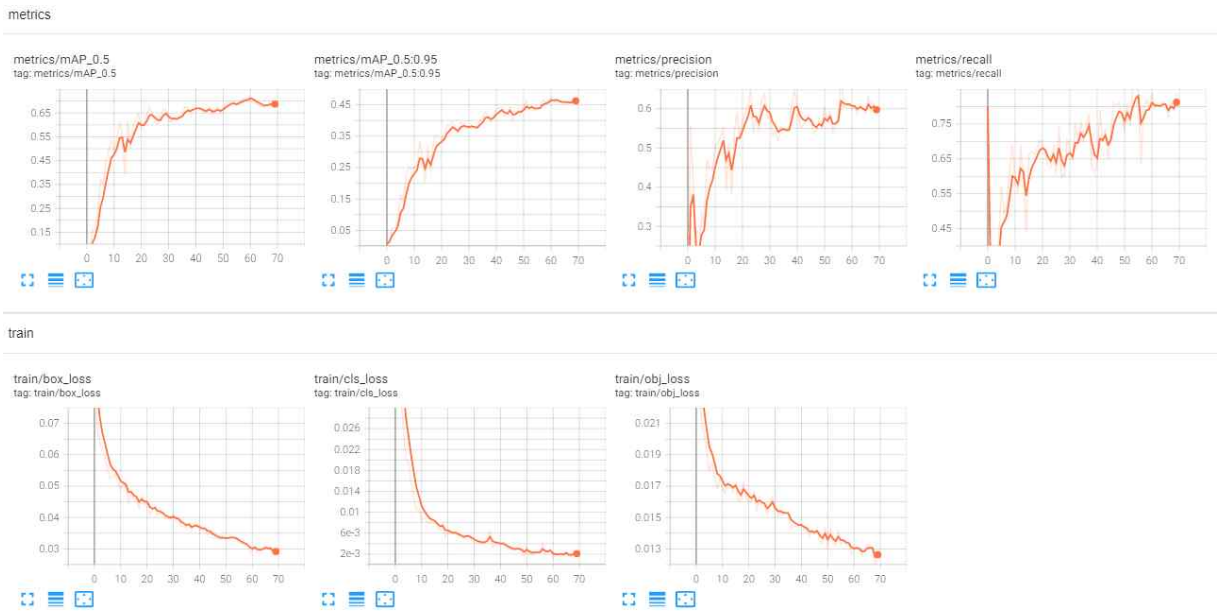
```
# 2023_OSSP_Google_Crawling

from selenium.webdriver.chrome.service import Service
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import urllib.request
import urllib.parse
import time

# Step 2. 사용자에게 검색 관련 정보들을 입력 받습니다.
print("=" *100)
print(" 2023_OSSP_Google_Crawling ")
print("=" *100)
query_txt = input('1. 검색어를 입력하시오 : ')
file_name = input('1. 파일명을 설정하시오 : ')

s = Service(r"C:\Users\rmsdu\OneDrive\문서\GitHub\2023_OSSP\2023_OSSP_Data\Train\chromedriver.exe")
driver = webdriver.Chrome(service=s)  # 구글 웹드라이버를 사용
driver.implicitly_wait(3)

url = "https://www.google.co.kr/imghp?hl=ko&ogbl" # 구글 이미지 페이지 주소
driver.get(url)  # 구글 이미지 페이지로 들어간다.
time.sleep(3)

element = driver.find_element(By.NAME, 'q')
element.send_keys(query_txt)
element.send_keys(Keys.RETURN)

SCROLL_PAUSE_TIME = 1
```

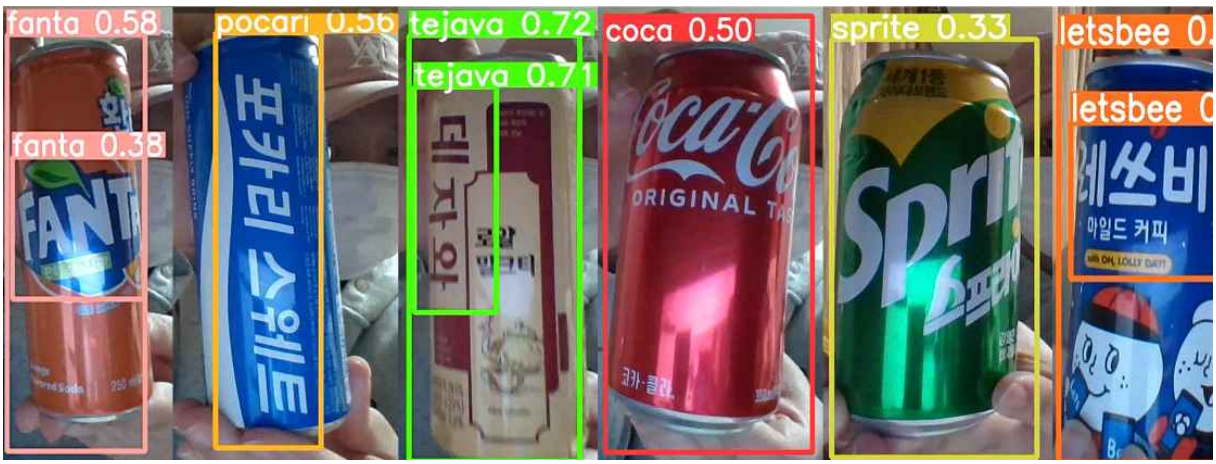
- Yolo V5 코드 추가
- 클래스 별로 이미지 라벨링을 진행
- 코랩으로 이미지 라벨링을 바탕으로 모델 학습 후 실시간 인식 적용

```
# 출력 결과물 : 이미지 & 음성파일
from IPython.display import Image, Audio
import os
import matplotlib.pyplot as plt
import matplotlib.image as img
import time

val_img_path =  '/content/MyDrive/MyDrive/2023_OSSP_YoloV5/Labeling_IMG/coca10.jpg' # 예측하고 싶은 이미지의 경로 입력

# !python detect.py --weights /content/yolov5/MyDrive/MyDrive/Colab/project01/yolov5_2s/best.pt --img 416 --conf 0.5 --source "{val_img_path}"
!python detect.py --weights /content/MyDrive/MyDrive/2023_OSSP_YoloV5/yolov5/runs/train/drink_yolov5s_results/weights/best.pt --img 416 --conf 0.5 --sou
                  # 생성된 가중치를 불러온다.                                    # 이미지 크기   # confidence    # 소스
# Image(os.path.join('/content/yolov5/runs/detect/exp',os.path.basename(val_img_path)))

result = img.imread(os.path.join('/content/MyDrive/MyDrive/2023_OSSP_YoloV5/yolov5/runs/detect/exp',os.path.basename(val_img_path)))
plt.imshow(result)
plt.show()
time.sleep(5)
```

```
detect: weights=['/content/MyDrive/MyDrive/2023_OSSP_YoloV5/yolov5/runs/train/drink_yolov5s_results/weights/best.pt'], source=/content/MyDrive/MyDrive/20
23_OSSP_YoloV5/Labeling_IMG/coca10.jpg, data=data/coco128.yaml, imgsz=[416, 416], conf_thres=0.5, iou_thres=0.45, max_det=1000, device=, view_img=False, s
ave_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=run
s/detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False, vid_stride=1
YOLOv5 🚀 v7.0-71-gc442a2e Python-3.8.10 torch-1.13.1+cu116 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
YOLOv5s summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
image 1/1 /content/MyDrive/MyDrive/2023_OSSP_YoloV5/Labeling_IMG/coca10.jpg: 416x416 1 coca, 8.6ms
Speed: 0.4ms pre-process, 8.6ms inference, 1.7ms NMS per image at shape (1, 3, 416, 416)
Results saved to runs/detect/exp5
```

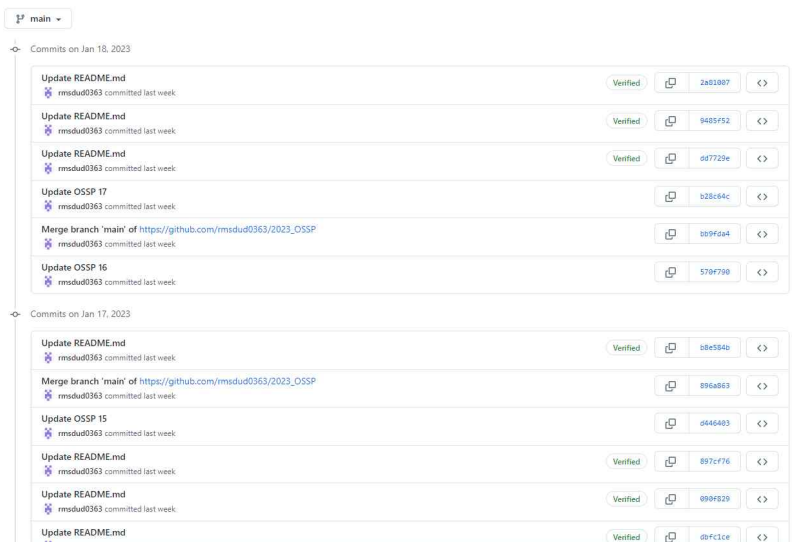






▷ 깃허브 로그 정보

· 1월 11일부터 하루를 제외하고는 성실하게 프로젝트를 진행하였습니다.



4. 프로젝트 느낀점

▷ 프로젝트하면서 좋았던 점

· 새로운 기술들을 공부하면서 진행할 수 있어서 좋았습니다.

· 오류 코드를 수정하고 프로젝트를 완성했을 때 뿌듯하였습니다.

· 깃허브 활용법을 배울 수 있어서 좋았습니다.

▷ 프로젝트하면서 아쉬웠던 점

· 선택한 소스코드가 오래전에 작성되었어서 분석의 어려움이 있었다.

· 소스코드의 오류가 있어 코드를 수정하는데 어려움이 많이 있었다.

· 프로젝트 시간 분할을 못하여 마지막에 음성 출력 코드를 추가하지 못한 점이 아쉬웠다.