스마트 팩토리 플랫폼과 최종발표
농작물의 상태(잎사귀 건강) 분석 솔루션

이 준 혁 (2023250419)
산 업 인 공 지 능 학 과

Dept. of Industrial AI

주제 선정

본인의 직/간접적 직무 분야 중 하나인 임베디드/IoT/스마트팜 분야로 선정
해당 주제 중 머신러닝을 이용한 작물의 병변 확인을 주제로 선정

데이터 선정: 작물의 병변에 관련된 정보

Coffee bean leafs

검색을 통해 적합 데이터 선정(Kaggle)

1600 여개의 잎에 나타난 병변에 관련된 자료 습득 (다운로드 하여 검토 후 불러오기)

진행순서: 사전작업(마운트, 캐글패키지, 다운로드 압축해제 등)

```
from google.colab import drive

# Google Drive를 마운트합니다.
drive.mount('/content/drive')

# .kaggle 폴더 만들기 및 kaggle.json 복사
!mkdir ~/.kaggle/
!cp "/content/drive/MyDrive/kaggle API/kaggle.json" ~/.kaggle/kaggle.js

# kaggle.json 파일 권한 설정
!chmod 600 ~/.kaggle/kaggle.json

# Kaggle 패키지 설치
!python -m pip install -qq kaggle

# 데이터셋 다운로드 및 압축 해제
!kaggle datasets download -d stunningvisionai/coffee-leaf-diseases-yolo
!unzip -qq "/content/coffee-leaf-diseases-yolo.zip"

# YOLOv5 깃허브 클론 및 디렉토리 이동
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
```

진행순서: 경로설정, 서브폴더 생성

```
# 의존성 설치
!pip install -r requirements.txt
!pip install pyyaml

import os
import shutil
from sklearn.model_selection import train_test_split
import yaml

# 데이터셋 경로 설정
dataset_path = '/content/coffee-leaf-diseases-yolo'
original_images_path = dataset_path
original_labels_path = dataset_path

# images, labels 서브 폴더 생성
images_path = os.path.join(dataset_path, 'images')
labels_path = os.path.join(dataset_path, 'labels')
train_images_path = os.path.join(images_path, 'train')
train_labels_path = os.path.join(labels_path, 'train')
val_images_path = os.path.join(images_path, 'val')
val_labels_path = os.path.join(labels_path, 'val')
```

진행순서: 목록 가져오기/데이터셋 분리/정리

```
# 전체 이미지와 라벨 파일 목록 가져오기
image_files = [os.path.join(original_images_path, f) for f in os.listdi
label_files = [f.replace('.jpg', '.txt') for f in image_files]

# 학습 및 검증 데이터셋으로 분리
train_images, val_images, train_labels, val_labels = train_test_split(i

# 분리된 데이터셋을 해당 폴더로 이동
for f in train_images:
    shutil.move(f, train_images_path)

for f in train_labels:
    shutil.move(f, train_labels_path)

for f in val_images:
    shutil.move(f, val_images_path)

for f in val_labels:
```

진행순서: 클래스 명(병변) 설정 및 Yaml 파일 만들기

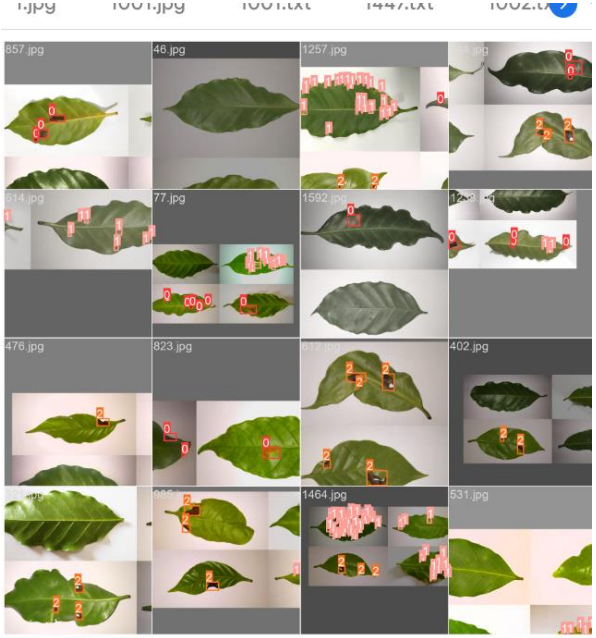
```
# 클래스 이름 설정
class_names = ['0','1', '2','3', '4', '5']

# data.yaml 파일 생성
data_yaml_content = {
    'train': train_images_path,
    'val': val_images_path,
    'nc': len(class_names),
    'names': class_names
}

# yaml 파일 쓰기
with open(os.path.join(dataset_path, 'data.yaml'), 'w') as yaml_file:
    yaml.dump(data_yaml_content, yaml_file)
```

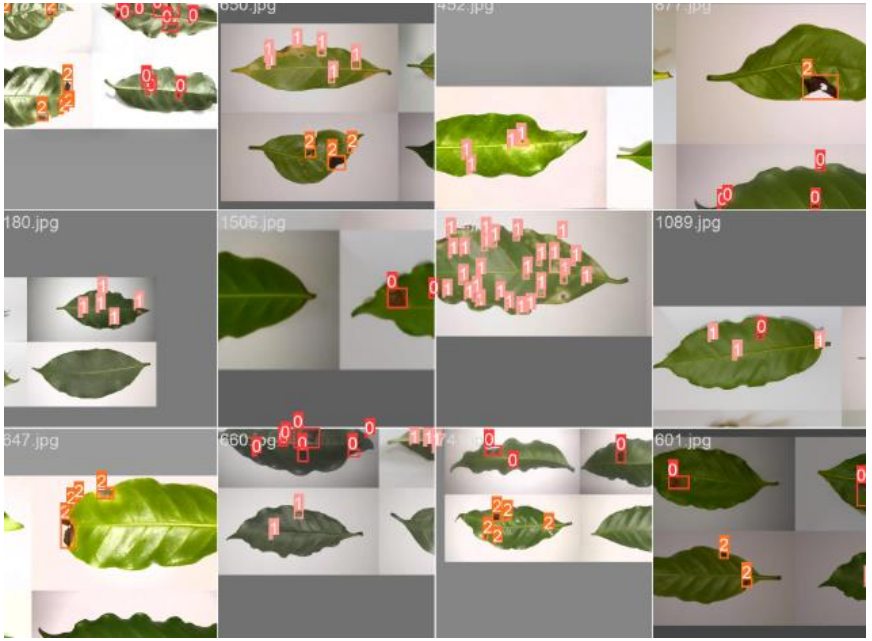
진행순서: 학습 및 검출

```
# 학습 및 검증 세트의 이미지와 라벨 파일 수 확인
num_train_images = len(os.listdir(train_images_path))
num_val_images = len(os.listdir(val_images_path))
num_train_labels = len(os.listdir(train_labels_path))
num_val_labels = len(os.listdir(val_labels_path))

# YOLOv5 모델 학습
!python train.py --img 640 --batch 16 --epochs 30 --data /content/coff

# 학습된 모델로 이미지에서 질병 검출
!python /content/yolov5/detect.py --weights /content/yolov5/runs/train
```

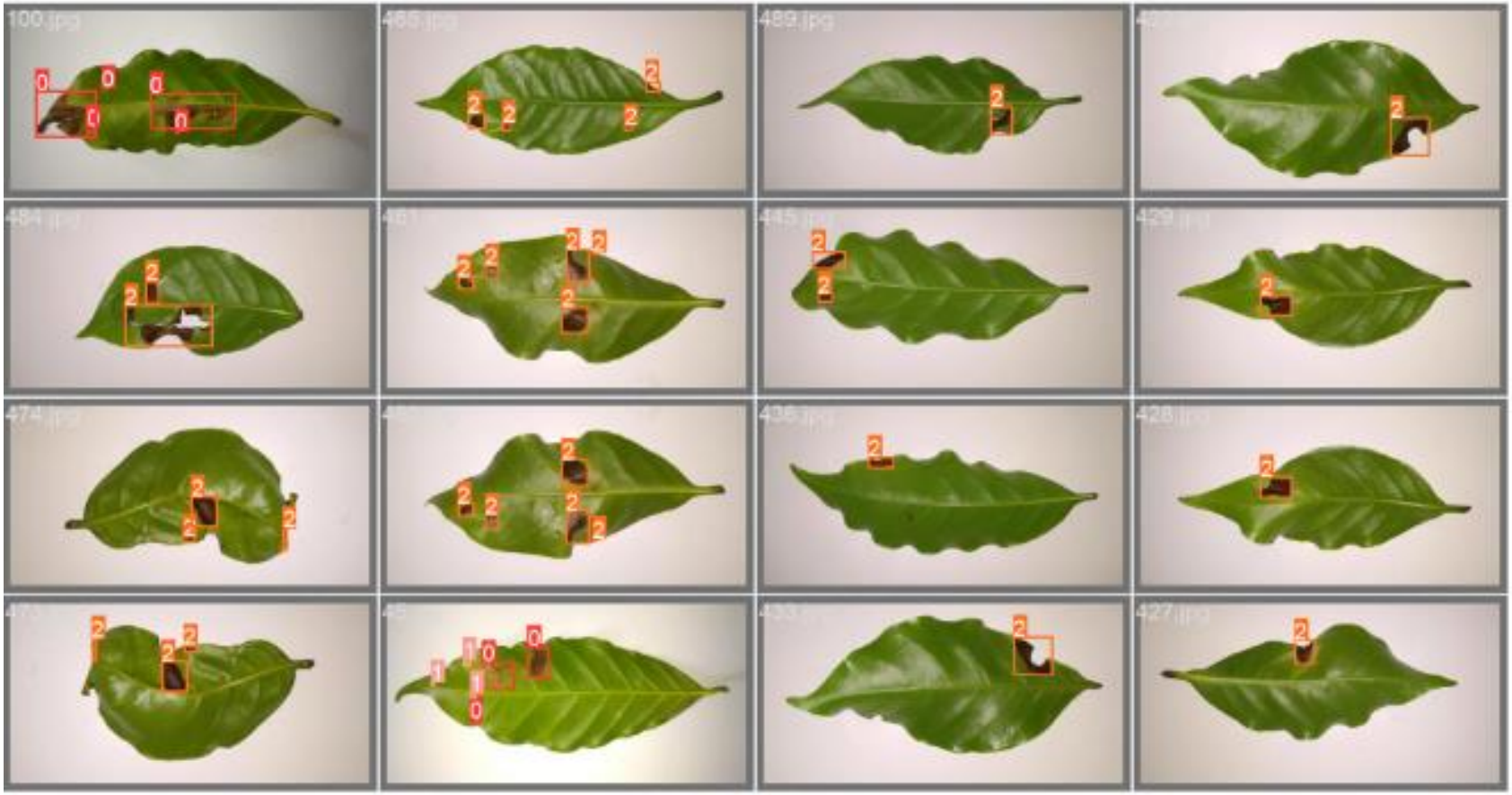
학습결과 (약 17분 소요됨)

```
      29/29      3.47G    0.03093    0.01939  0.0009722         15
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 11
                   all        333       1294      0.944      0.891      0.935      0.692

30 epochs completed in 0.271 hours.
Optimizer stripped from runs/train/exp2/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp2/weights/best.pt, 14.4MB

Validating runs/train/exp2/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 11
                   all        333       1294      0.944      0.891      0.934      0.692
                     0        333        262      0.957      0.958      0.981      0.793
                     1        333        809      0.888      0.747      0.829      0.463
                     2        333        223      0.986      0.967      0.992      0.821
Results saved to runs/train/exp2
```

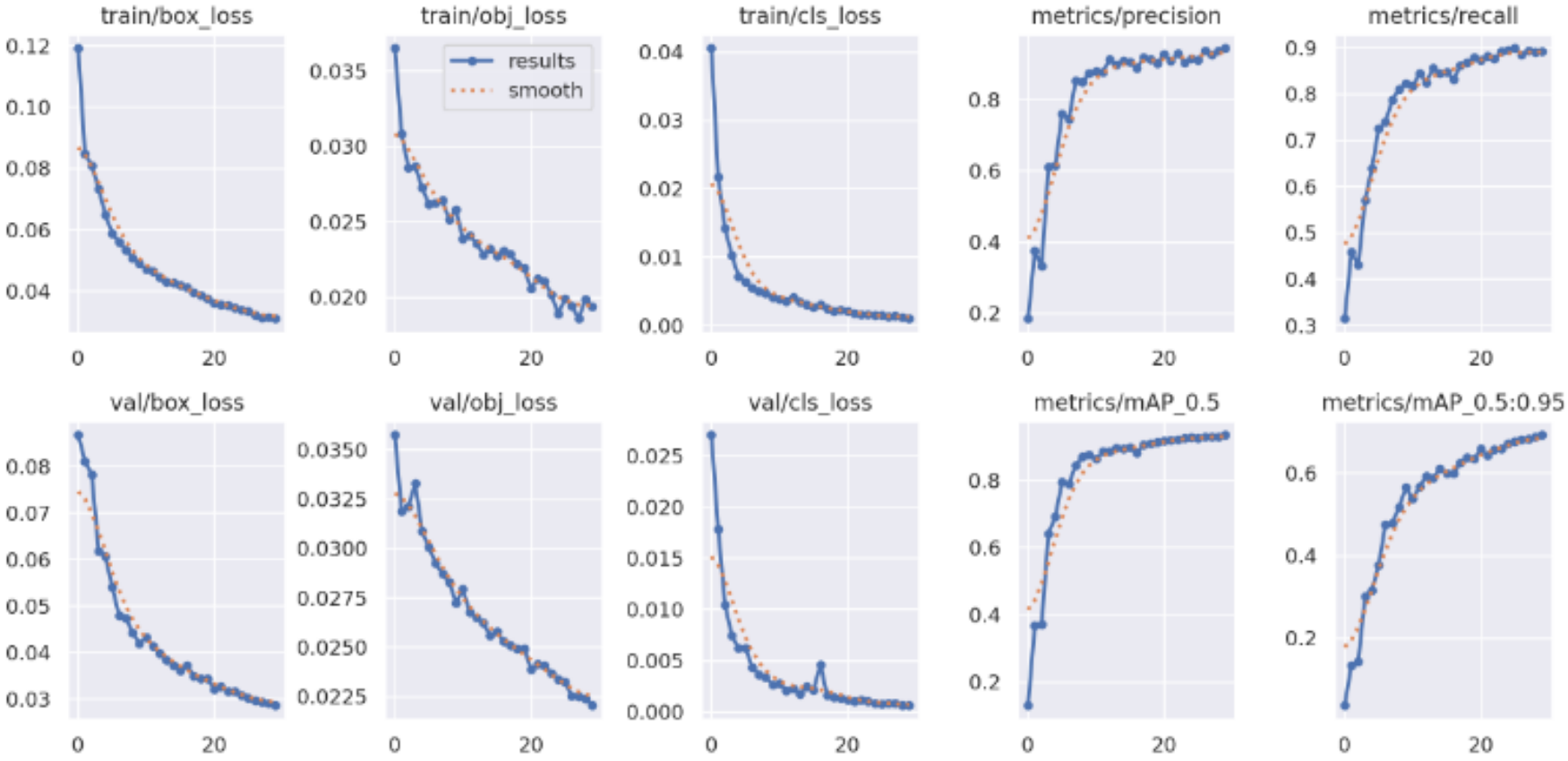




학습결과2



학습결과2


train/box_loss
train/obj_loss
results
smooth
train/cls_loss
metrics/precision
metrics/recall
val/box_loss
val/obj_loss
val/cls_loss
metrics/mAP_0.5
metrics/mAP_0.5:0.95

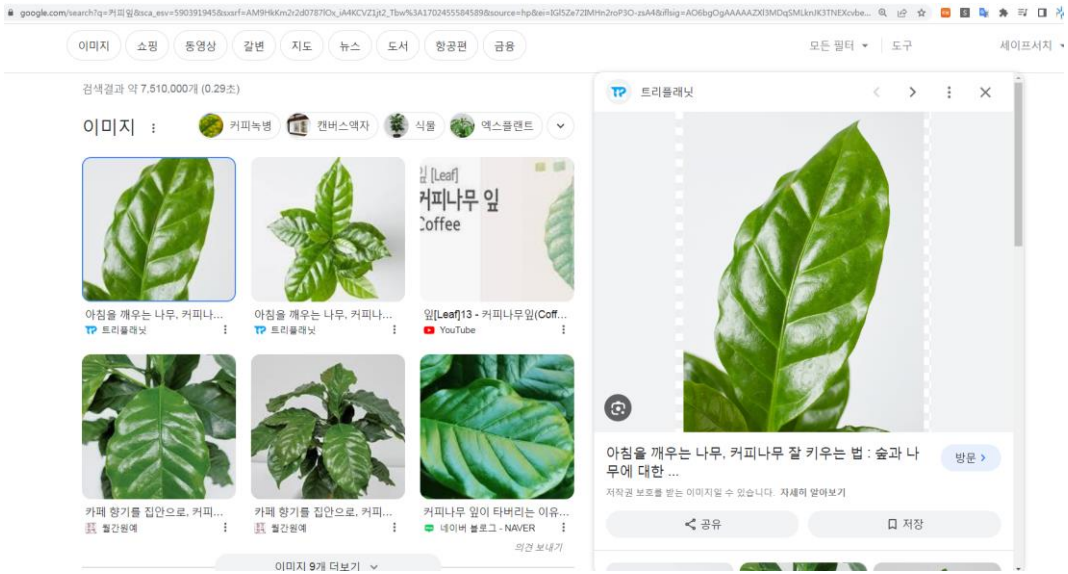
원하는 데이터를 삽입 후 테스트
Coffee bean leafs

```
from google.colab import drive
import torch
from PIL import Image
import matplotlib.pyplot as plt

# Google Drive를 마운트합니다.
drive.mount('/content/drive', force_remount=True)


# 이미지 경로 설정
img_path = '/content/drive/My Drive/커피잎병.jpeg'  # Google D
#img_path = '/content/drive/My Drive/건강잎.jpeg'  # Google Dr
# YOLOv5 detect.py 스크립트 실행
!python /content/yolov5/detect.py --weights /content/yolov5/ru
```

2개의 사진을 구글 드라이브에 업로드 및 결과 확인

Dept.of Industrial AI

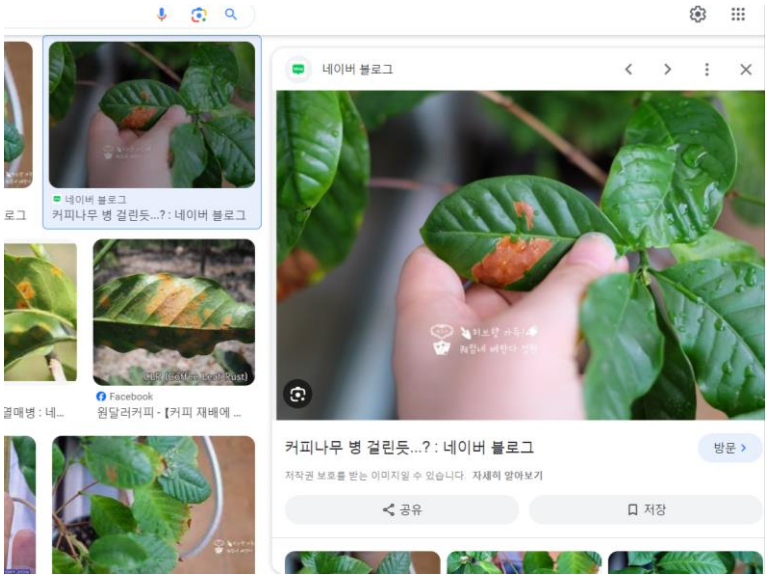
병변이 없는 잎
Coffee bean leafs

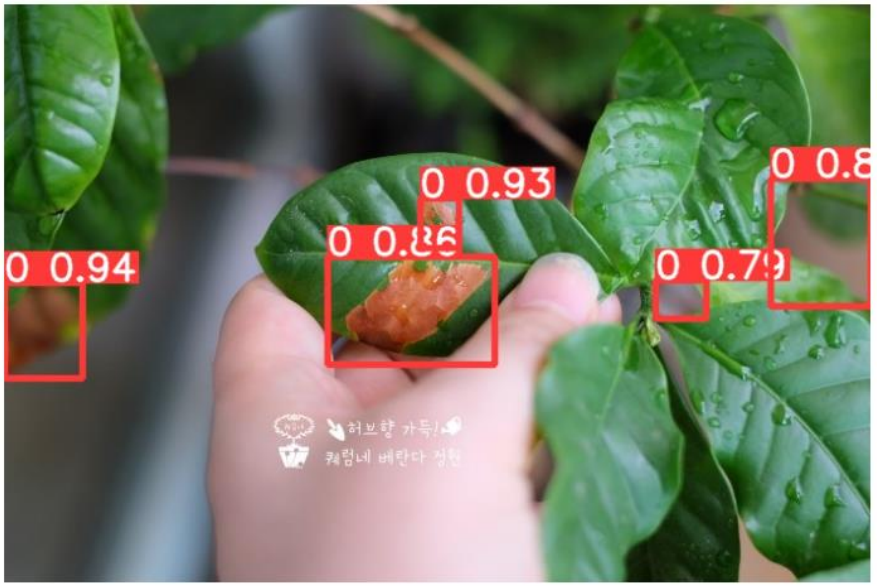


```
Mounted at /content/drive
detect: weights=['/content/yolov5/runs/train/exp2/weights/best.pt'
YOLOv5 🚀 v7.0-249-gf400bba Python-3.10.12 torch-2.1.0+cu118 CUDA:0

Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GF
image 1/1 /content/drive/My Drive/건강잎.jpeg: 640x480 (no detecti
Speed: 0.5ms pre-process, 43.8ms inference, 0.4ms NMS per image at
Results saved to runs/detect/exp11
```

클라스 미검출

병변이 있는 잎
Coffee bean leafs






```
Mounted at /content/drive
detect: weights=['/content/yolov5/runs/train/exp2/weights/best.pt'], source=/content/drive
YOLOv5 🚀 v7.0-249-gf400bba Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
image 1/1 /content/drive/My Drive/커피잎병.jpeg: 448x640 5 0s, 44.2ms
Speed: 0.5ms pre-process, 44.2ms inference, 1.5ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp10
```

추후 계획/ 기타Coffee bean leafs

- 추후에는 원하는 작물별 데이터를 다양화(깻잎, 무순 등등) 하여 테스트 수행 예정
- 추후에는 임베디드 장치에 GPU탑재하여 실시간 영상 감지로 수행 예정
- 느낀 점: API 연결/ 이미지 불러오기 등의 작업을 함에 있어서 생각보다 경로 관련 문제점이 많이 발생하였음.
- 해당 프로젝트를 수행하며 전체적인 머신러닝의 흐름/내용을 파악하며 세부적으로 공부할 수 있는 토대를 이해할 수 있었음.

감사합니다