DLIP Extra Project Checking prescription filled for pharmacists

Date: 2023/06/27

Author: 22000288 Youhyeon Park

Demo Video:https://youtu.be/2NAq6mm8Y98

Github : https://github.com/youhyeon2709/DLIP_Project_Checking-prescription-filled-for-pharmacists_2023.git

Introduction

Article 26, Paragraph 1 of the Pharmacy Law (약사법 제26조 제1항)

> **Article 26 (Alteration or Modification of Prescription)**
> ① A pharmacist or oriental medicine pharmacist shall not alter or modify a prescription and dispense it without the consent of the prescribing physician, dentist, oriental medical practitioner, or veterinarian.

There is a legislation regarding pharmacists in South Korea, as mentioned above.

In May 2020, a pharmacist was prosecuted for violating the pharmacy law on suspicion of altering and dispensing a prescription generated by a doctor at a pharmacy located in Seoul. The incident involved Pharmacist B who dispensed a 7-day supply of medication to Patient A according to the prescription. However, the dispensed medication included an additional prescription item, namely Pristic Sustained-Release Tablets 50mg, which was not indicated on the prescription. Due to the presence of multiple pharmacists working simultaneously, an error occurred in the dispensing process.

Mistakes often occur during the medication dispensing process, especially during busy hours when patients gather at the pharmacy. These errors not only pose risks to patient health but also put diligent pharmacists in challenging situations due to simple human errors. In order to improve the treatment of pharmacists and ensure that patients receive the correct medication, there is a plan to develop a program that checks prescriptions and dispensed medication.

One of the most frequently prescribed types of medications at "Boryeong Pharmacy" located in Daegu City was used for this purpose. In a recorded video of the dispensing process captured by a camera, a single dosage unit was defined as one section, and when it was correctly dispensed, it was marked in blue, while incorrect dispensing was marked in red.

Customizing the data for detecting medicines required labeling the medications, which was achieved by utilizing Roboflow. A training set of 48 images was created by segmenting the objects. To detect the objects, YOLOv8 in Visual Studio Code and Python via Anaconda virtual environment were employed.

Goal of this project

- Accuracy : 90%
- Make students have good eating habit through the app.

1. Requirement

Hardware

- NVIDIA DGX Station A100

Environment constraint

- Camera height : 41[cm] from Desk

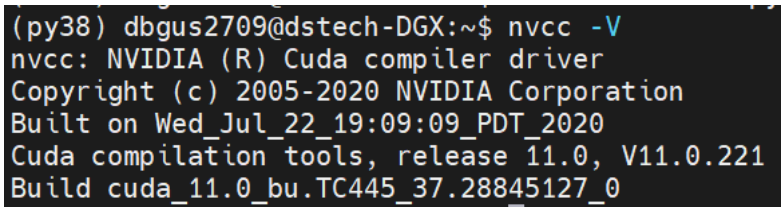
Software Installation

- Visual Studio Code
- Anaconda
- YOLO v8 l model
- Python 3.8.16
- Pytorch 1.10.2
- CUDA 11.0

Anaconda settings

Check what CUDA version has installed.

```
# Check your CUDA version
nvcc -V
```



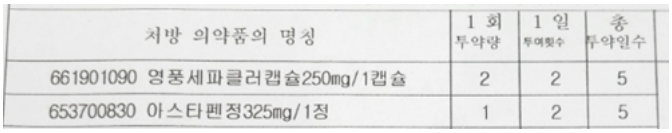
Check my CUDA version is 11.0.

DATASET

- Prescription

  We selected a prescription composed of the most frequently prescribed medications from "Boryeong Pharmacy" located in Daegu City. The prescription includes the following medications:

  1. Cefaclor Capsules (referred to as 'Cefaclor_Cap' below): It should be taken at a dosage of 2 capsules per administration, twice a day (morning and evening), for a duration of 5 days.
  2. Astaphen Tablets (referred to as 'Astaphen_Tab' below): It should be taken at a dosage of 1 tablet per administration, twice a day (morning and evening), for a duration of 5 days.

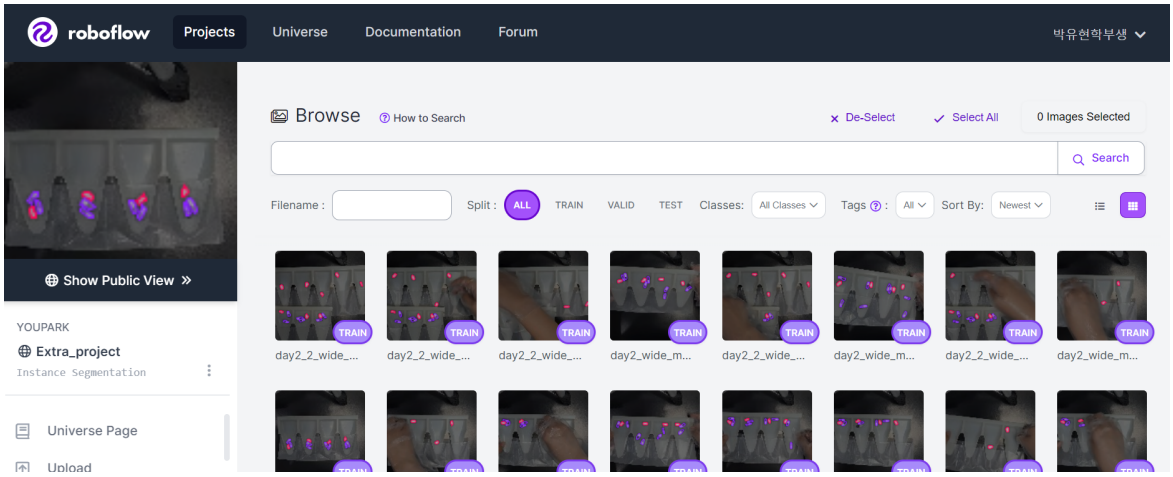


처방 의약품의 명칭	1 회 투약량	1 일 투여횟수	총 투약일수
661901090 영풍세파클러캡슐250mg/1캡슐	2	2	5
653700830 아스타펜정325mg/1정	1	2	5

- **Dataset download URL** : https://app.roboflow.com/ds/q4l5mq1nNo?key=Gui83v3mOH

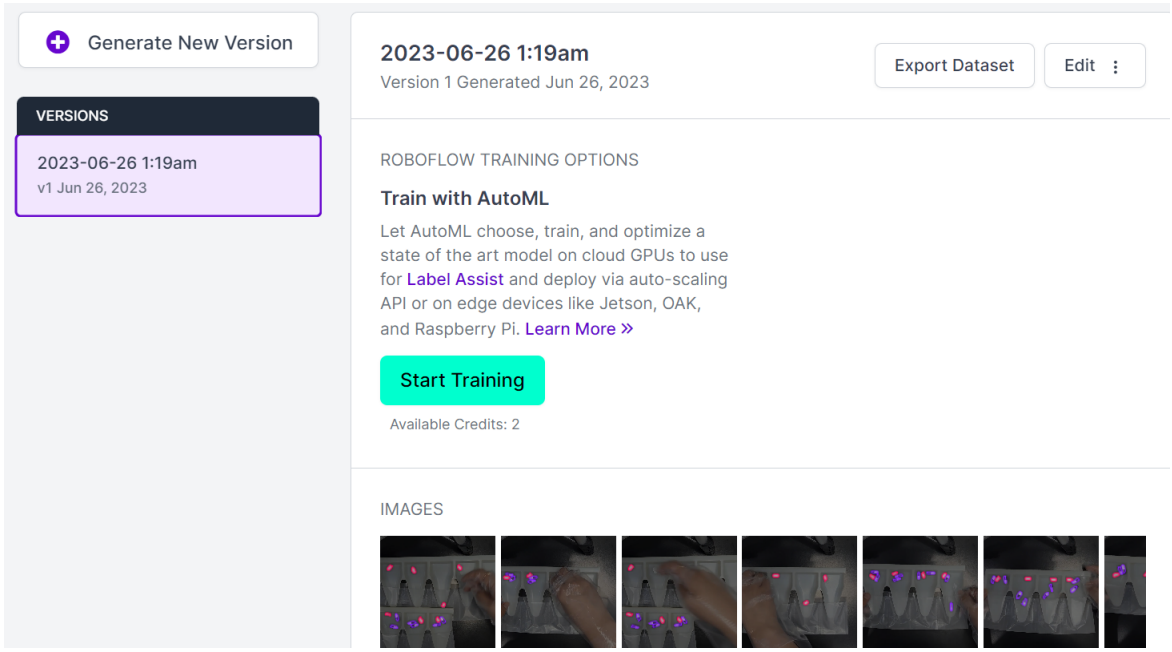
Labeling

- Roboflow



Roboflow is a website that allows for the generation of custom data. We created instance segments for each medication. Additionally, this program enables image augmentation through flipping, rotation, and other techniques, thereby generating multiple training images. We collected 16 images for labeling, but with augmentation, we obtained a total of 48 training data sets. Although we collected 16 images, each image contains multiple instances of the same type of medication. As a result, the actual number of labeled medications is approximately 4-5 times the number of images.

1. Upload images.
2. Start annotating and draw instance segmentaion.
3. Add them to dataset.
4. Generate train data.
    - You can also split training, validation, test set.
    - You can use preprocessing of data if you need.
    - Augment images.
    - Generate data.



If train set is downloaded well, you can get data.yaml file that includes the number of classes and their names, and path of the images. You need to modify the path to the same path with yolov8 file exists.

```
train: ../datasets/calories/images/train/
val: ../datasets/calories/images/validation/

nc: 2
names: ['Astaphen_Tab', 'Cefaclor_Cap']
```

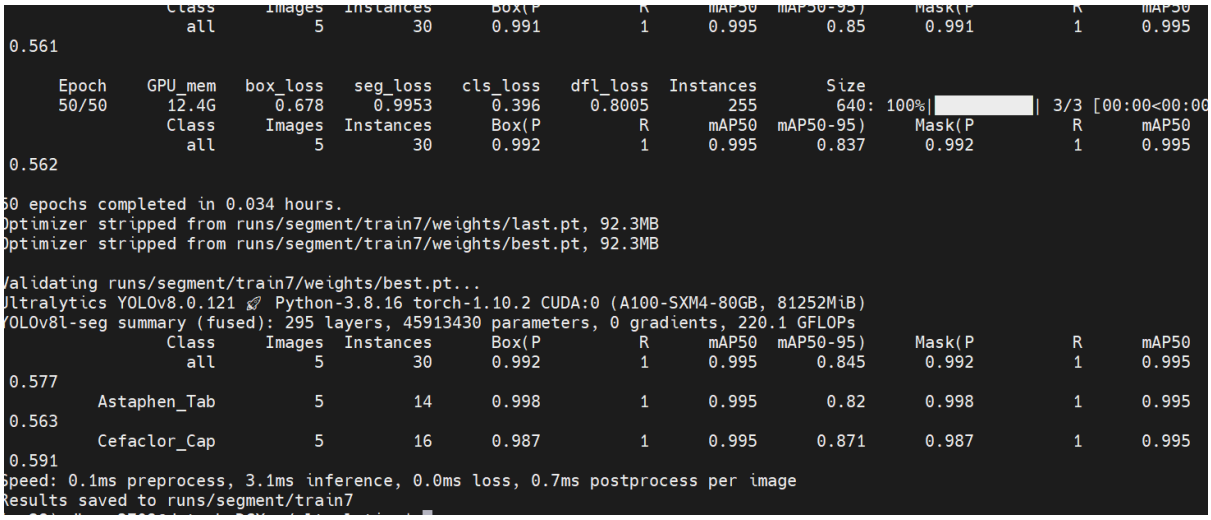
If data are ready to be trained like this, training these data with yolov8 is next step.

Training data

In this project, we utilized the YOLOv8 framework, specifically the YOLOv8l-seg model. Depending on the specifications of the computer, program speed, and object detection accuracy, we could choose between different sizes: small (s), medium (m), large (l), or extra-large (x). Due to the limited amount of data available, we opted for the 'l' model, which offers higher accuracy. We trained the YOLOv8 model using the code provided in the 'train.py' script, resulting in the acquisition of the 'best.pt' file.

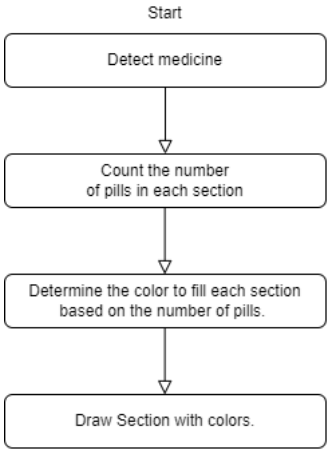
```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8l-seg.yaml')  # build a new model from YAML
model = YOLO('yolov8l-seg.pt')  # load a pretrained model (recommended for training)
model = YOLO('yolov8l-seg.yaml').load('yolov8l.pt')  # build from YAML and transfer weights

# Train the model
model.train(data='medicine.yaml', epochs=100, imgsz=640)
```

To train the model, you need to specify the images, data, epochs, and image size. In our case, we used the 'medicine.yaml' file as a configuration file and trained the images using 100 epochs and an image size of 640.



2. Algorithm

Flow Chart

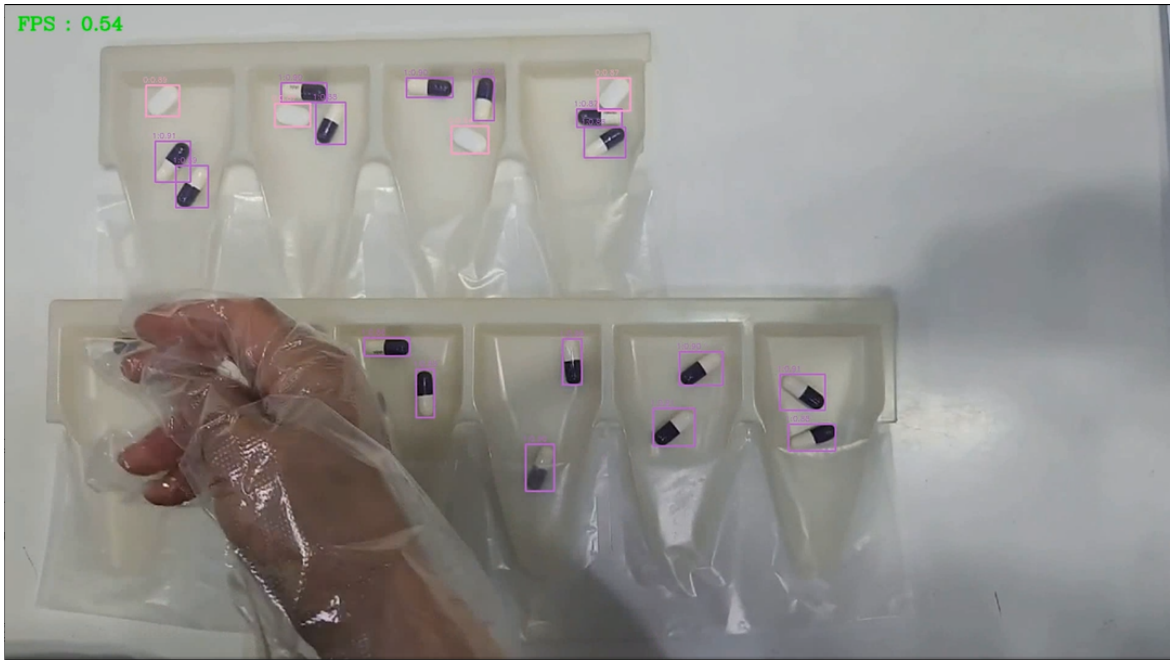


1. Detet medicine with Yolov8.
2. Count the number of pills in each section.
3. Determine the color to fill each section based on the number of pills.
4. Draw section withe the colors.

Detet medicine with Yolov8

One uses the best.pt file trained with yolov8 to detect pills.

- Result Image



- Code

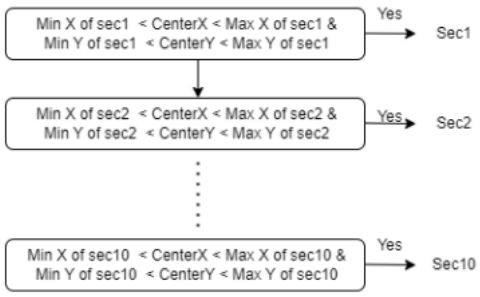
```
results = model(frame, imgsz=640)
        result = results[0]
        len_result = len(result)

        if len_result != 0: # if object are detectec

                for idx in range(len_result):

                    detection = result[idx]
                    box = detection.boxes.cpu().numpy()[0]
```

```
                    conf = box.conf[0]

                    if conf<0.5 : continue
                    cls = int(box.cls[0])

                    xywh    = box.xywh[0].astype(int)
                    centerX = xywh[0]
                    centerY = xywh[1]
                    area    = xywh[2] * xywh[3]


                    color = colors[cls]

                    counts_med(centerX,centerY)


                    r = box.xyxy[0].astype(int) # box

                    class_id = result.names[cls]
                    conf = round(conf.item(), 2)

                    cv2.rectangle(frame, r[:2], r[2:], color, 2)
                    cv2.putText(frame, f'{cls_name[cls]}:{conf:.2f}', (r[0]-5, r[1] - 5),cv2.FONT_HERSHEY_SIMPLEX, 0.4, color, 1)
```

Count the number of pills in each section

One finds the section based on where the center x, y coordinates of the detection box are. The number of pills in each section is counted up.

- flow chart



- Code

```
def counts_med(centerX,centerY):

    if (recxyxy[0][0] < centerX< recxyxy[0][2]) and (recxyxy[0][1] < centerY < recxyxy[0][3]):       #  Min X of sec1  < CenterX <
        counts['sec1'][cls] +=1

    elif (recxyxy[1][0] < centerX< recxyxy[1][2]) and (recxyxy[1][1] < centerY < recxyxy[1][3]):
        counts['sec2'][cls] +=1

    elif (recxyxy[2][0] < centerX < recxyxy[2][2]) and (recxyxy[2][1] < centerY < recxyxy[2][3]):
        counts['sec3'][cls] +=1

    elif (recxyxy[3][0] < centerX < recxyxy[3][2]) and (recxyxy[3][1] < centerY < recxyxy[3][3]):
        counts['sec4'][cls] +=1

    elif recxyxy[4][0] < centerX < recxyxy[4][2] and recxyxy[4][1] < centerY < recxyxy[4][3]:
        counts['sec5'][cls] +=1

    elif recxyxy[5][0] < centerX < recxyxy[5][2] and recxyxy[5][1] < centerY < recxyxy[5][3]:
        counts['sec6'][cls] +=1

    elif recxyxy[6][0] < centerX < recxyxy[6][2] and recxyxy[6][1] < centerY < recxyxy[6][3]:
        counts['sec7'][cls] +=1

    elif recxyxy[7][0] < centerX < recxyxy[7][2] and recxyxy[7][1] < centerY < recxyxy[7][3]:
```

```
        counts['sec8'][cls] +=1

    elif recxyxy[8][0] < centerX < recxyxy[8][2] and recxyxy[8][1] < centerY < recxyxy[8][3]:
        counts['sec9'][cls] +=1

    elif recxyxy[9][0] < centerX < recxyxy[9][2] and recxyxy[9][1] < centerY < recxyxy[9][3]:
        counts['sec10'][cls] +=1
```

Determine the color to fill each section based on the number of pills.

If the section is empty, it will be colored white, and if it contains the right number of pills, it will be colored green. Otherwise, it is colored red.

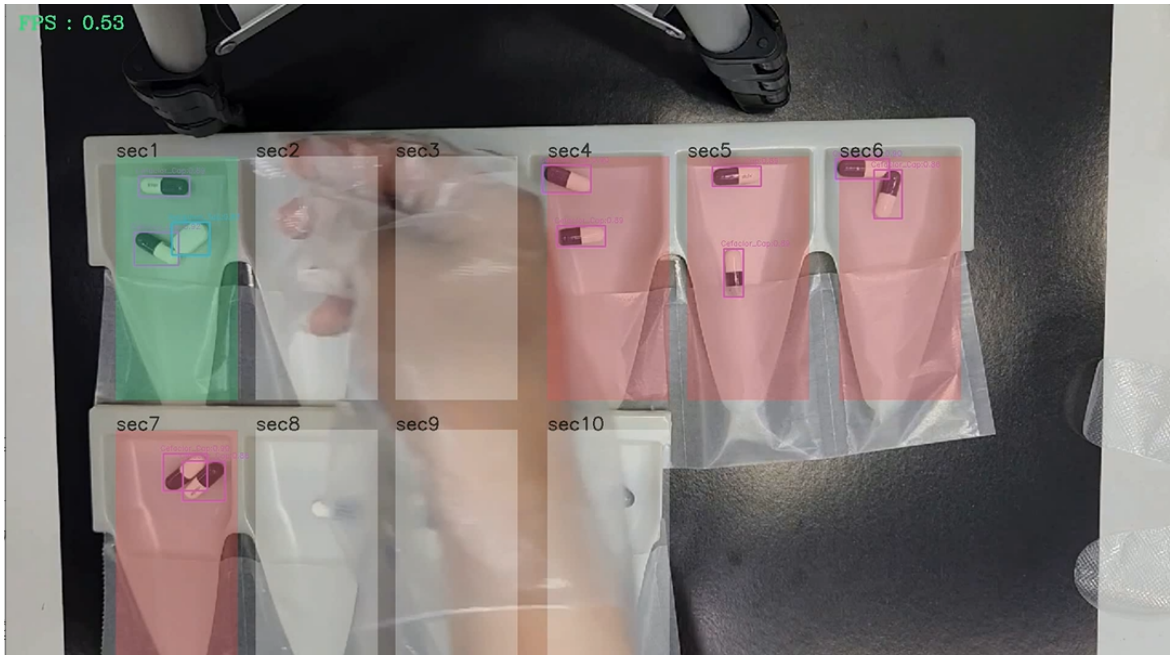
- Code

```
def section_color():

    for i in range(len(counts)):
        if counts[f'sec{i+1}'][:2]==[0,0]:      # if section is empty
            counts[f'sec{i+1}'][2] = white

        elif counts[f'sec{i+1}'][:2]==[1,2]:    # if Medication dispensed correctly.
            counts[f'sec{i+1}'][2] = green

        else :                                  # wrong
            counts[f'sec{i+1}'][2] = red
```

Draw section withe the colors.

The color of each section, which you specified in the step above, indicates whether the pill is being dispensed correctly. If a section is colored blue, it indicates that it is dispensing correctly, red indicates that the pill is dispensing incorrectly, and white indicates that the section is empty.

- Result Image



- Code

```
def drawsec(image,recxyxy):

    overlay = image.copy()

    for i in range(10):
        cv2.putText(image, recxyxy[i][4], recxyxy[i][:2],cv2.FONT_ITALIC, 1, black, 2)
        cv2.rectangle(overlay, recxyxy[i][:2],recxyxy[i][2:4], counts[recxyxy[i][4]][2], -1)        #draw rectangle

    image = cv2.addWeighted(overlay, alpha, image,  1 - alpha, 0)                                   # Composite the rectangle to the o
    return image
```

3. Evaluation and Analysis

Calculate Accuracy

- Definition

  Accuracy means the ratio of the number of samples predicted by my algorithm to be correct out of the total number of samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} :$$

- Method

  Count the number of true and false Detected objects in the last frame of the Day 1 and Day 2 videos.

- Result

Video	True	False
Da1	29	1
Day2	30	0

  Accurancy = 98.3%

Analysis

Of the objects detected in the last frame of the 'Day1' and 'Day2' videos, 59 are true and 1 is false, for an accuracy of 98.3%. This exceeds our initial goal of 90%, making this project a success.

When detecting objects, our first dataset attempt was challenged by light reflections and the difference between the size of the dataset pills and the size of the pills in the frame. This was solved by stabilizing the camera with a tripod and placing a piece of black paper under the pill to reduce light reflection.

4. Conclusion

Through this project, we could reach the goal what we set as 90% in accuracy.

We used roboflow to segment the pill data to create a dataset. We used yolov8 to train the dataset and recognize objects. We conducted a test using prescriptions and manufacturing process videos from 'Boryeong Pharmacy' and recognized most of the pills well. We used the parameter of the bounding box of the recognized pill to count the number of pills. While watching the video, we set the coordinates of the sections of one pill bag as one section. If the center of the bounding box was within the section coordinates, the number of pills in the section was counted up for each type. If the pills were dispensed well, they were colored blue or red to help the pharmacists in dispensing them.

Appendix

Code

```
import torch
import cv2
from matplotlib import pyplot as plt
import time
import pandas as pd
import numpy as np
from ultralytics import YOLO
import yaml

'''
load model
'''

weight_path = 'best.pt'
model = YOLO(weight_path)

'''
class name
'''

with open('data.yaml') as f:
    yaml_data = yaml.load(f, Loader=yaml.FullLoader)

    # Access the 'names' key and loop through the items
    cls_name = yaml_data.get('names')
    if cls_name:
        for name in cls_name:
            print(name)
    else:
        print("'names' key not found in the YAML file")

'''
video open
'''
input_file = 'day2_wide.mp4'
cap = cv2.VideoCapture(input_file)
cap.set(cv2.CAP_PROP_FPS, 60.0)
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M','J','P','G'))
cap.set(cv2.CAP_PROP_AUTO_EXPOSURE, 0.75)

width   = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height  = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps     = int(cap.get(cv2.CAP_PROP_FPS))
user_font    = cv2.FONT_HERSHEY_COMPLEX


'''
video storage
'''
fourcc = cv2.VideoWriter_fourcc(*'XVID')
output_filename = f'output_{input_file}'
frame_size = (width, height)
out = cv2.VideoWriter(output_filename, fourcc, fps, frame_size)

'''
load prescription
'''
prescription = 'prescription.xlsx'
df_prescription = pd.read_excel(prescription)                          # 엑셀 파일을 데이터프레임으로 읽기

'''
Define color
'''
colors = {0: (232, 168, 63),  # blue for class 0
          1: (211,85,186),  # Purple for class 1
          }
green = (127, 232, 63)
red = (84,63,232)
black = (0,0,0)
sky = (208,224,64)
white=(255,255,255)


'''
section coordination
'''
recxyxy = { 0: (180,250,380,650,'sec1'),
            1:(410,250,610,650,'sec2'),
            2:(640,250,840,650,'sec3'),
            3:(890,250,1090,650,'sec4'),
            4:(1120,250,1320,650,'sec5'),
            5:(1370,250,1570,650,'sec6'),
            6:(180,700,380,1080,'sec7'),
            7:(410,700,610,1080,'sec8'),
            8:(640,700,840,1080,'sec9'),
```

```
            9:(890,700,1090,1080,'sec10'),
}

alpha = 0.3

counts={
        'sec1':[0,0,white],
        'sec2':[0,0,white],
        'sec3':[0,0,white],
        'sec4':[0,0,white],
        'sec5':[0,0,white],
        'sec6':[0,0,white],
        'sec7':[0,0,white],
        'sec8':[0,0,white],
        'sec9':[0,0,white],
        'sec10':[0,0,white],
}

'''
section 그리기
'''

def drawsec(image,recxyxy):

    overlay = image.copy()

    for i in range(10):
        cv2.putText(image, recxyxy[i][4], recxyxy[i][:2],cv2.FONT_ITALIC, 1, black, 2)
        cv2.rectangle(overlay, recxyxy[i][:2],recxyxy[i][2:4], counts[recxyxy[i][4]][2], -1)        #draw rectangle

    image = cv2.addWeighted(overlay, alpha, image,  1 - alpha, 0)                                   # Composite the rectangle to the o
    return image

'''
Count medicine
'''

def counts_med(centerX,centerY):

    if (recxyxy[0][0] < centerX< recxyxy[0][2]) and (recxyxy[0][1] < centerY < recxyxy[0][3]):       #  Min X of sec1  < CenterX < Max
        counts['sec1'][cls] +=1

    elif (recxyxy[1][0] < centerX< recxyxy[1][2]) and (recxyxy[1][1] < centerY < recxyxy[1][3]):
        counts['sec2'][cls] +=1

    elif (recxyxy[2][0] < centerX < recxyxy[2][2]) and (recxyxy[2][1] < centerY < recxyxy[2][3]):
        counts['sec3'][cls] +=1

    elif (recxyxy[3][0] < centerX < recxyxy[3][2]) and (recxyxy[3][1] < centerY < recxyxy[3][3]):
        counts['sec4'][cls] +=1

    elif recxyxy[4][0] < centerX < recxyxy[4][2] and recxyxy[4][1] < centerY < recxyxy[4][3]:
        counts['sec5'][cls] +=1

    elif recxyxy[5][0] < centerX < recxyxy[5][2] and recxyxy[5][1] < centerY < recxyxy[5][3]:
        counts['sec6'][cls] +=1

    elif recxyxy[6][0] < centerX < recxyxy[6][2] and recxyxy[6][1] < centerY < recxyxy[6][3]:
        counts['sec7'][cls] +=1

    elif recxyxy[7][0] < centerX < recxyxy[7][2] and recxyxy[7][1] < centerY < recxyxy[7][3]:
        counts['sec8'][cls] +=1

    elif recxyxy[8][0] < centerX < recxyxy[8][2] and recxyxy[8][1] < centerY < recxyxy[8][3]:
        counts['sec9'][cls] +=1

    elif recxyxy[9][0] < centerX < recxyxy[9][2] and recxyxy[9][1] < centerY < recxyxy[9][3]:
        counts['sec10'][cls] +=1

'''
Define Section's color
'''
def section_color():

    for i in range(len(counts)):
        if counts[f'sec{i+1}'][:2]==[0,0]:      # if section is empty
            counts[f'sec{i+1}'][2] = white

        elif counts[f'sec{i+1}'][:2]==[1,2]:    # if Medication dispensed correctly.
            counts[f'sec{i+1}'][2] = green

        else :                                  # wrong
            counts[f'sec{i+1}'][2] = red

# Loop through the video frames
while cap.isOpened():
```

```
    start_time = time.time()
    prev_time = start_time

    ret, frame = cap.read()

    for key in counts:
        counts[key][:2] = [0, 0]                            #initialize counts dictionary
        counts[key][2] = 'white'

    if ret == True: # Run YOLOv8 inference on the frame

        results = model(frame, imgsz=640)
        result = results[0]
        len_result = len(result)

        if len_result != 0:                                 # object are decteced

                for idx in range(len_result):

                    detection = result[idx]
                    box = detection.boxes.cpu().numpy()[0]
                    conf = box.conf[0]

                    if conf<0.5 : continue
                    cls = int(box.cls[0])

                    xywh    = box.xywh[0].astype(int)
                    centerX = xywh[0]
                    centerY = xywh[1]
                    area    = xywh[2] * xywh[3]


                    color = colors[cls]

                    counts_med(centerX,centerY)               #count pills in section

                    r = box.xyxy[0].astype(int) # box

                    class_id = result.names[cls]
                    conf = round(conf.item(), 2)

                    cv2.rectangle(frame, r[:2], r[2:], color, 2)
                    cv2.putText(frame, f'{cls_name[cls]}:{conf:.2f}', (r[0]-5, r[1] - 5),cv2.FONT_HERSHEY_SIMPLEX, 0.4, color, 1)

        section_color()                                             #determine section's color
        frame = drawsec(frame,recxyxy)                              #draw section

        '''
        calculate FPS
        '''

        diff_time = time.time() - prev_time

        if diff_time > 0:
            fps = 1 / diff_time

        cv2.putText(frame, f'FPS : {fps:.2f}', (20, 40), user_font, 1, green, 2)

        out.write(frame)

        resized_image = cv2.resize(frame, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_LINEAR)                    # 이미지 크기 조정
        cv2.imshow("mask", resized_image)

        key = cv2.waitKey(1) & 0xFF

        if key == ord('q')   :   break

    else:
        print("Vidieo is empty")
        break

cap.release()
cv2.destroyAllWindows()
```

Demo video (Full version)

[Full video]DLIP_Project_Checking prescription filled for pharmacists_2023

https://youtu.be/Lc-z0f1Qp6I