DLIP Project Calculating Calories for HGU Student2023

Date: 2023/06/20

Author: 22000055 Nayeon Kim, 22000288 Youhyeon Park

Demo Video:DLIP_Project_CalculateCalories_2023

Github : https://github.com/youhyeon2709/DLIP_Project_Calculating_Calories_2023.git

Introduction

Through the final project, we want to implement a diet detection and an algorithm for calculating calories and nutrients necessary for the diet control application. This program can be helpful for students who want to care for their health and control diet by checking what and how much they ate a day. Nutritional information includes three major and essential ingredients; carbohydrates, protein, and fat. In addition, by showing the recommended daily calories and the recommended daily amounts of carbohydrates, proteins, and fats, user can identify which nutrients are lacking and how much you should take. In this program, we chose three main cafeterias in Handong and they are '든든한동', '명성', and 'Yesod'.

Because we need some customized data such as specific menus in Handong, we used roboflow in order to label food images. We segmented original food images and augmented them by flipping and rotating so that we could get about 1,000 training set. For object detection, we used YOLOv5 and Python via Anaconda virtual environment in Visual Studio Code.

Goal of this project

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

1. Requirement

Hardware

- NVIDIA GeForce GTX 1050

Software Installation

- Visual Studio Code
- Anaconda
- YOLO v5 l model
- Python 3.9.13
- Pytorch 2.0.1
- CUDA 11.8

Anaconda settings

Check what CUDA version has installed.

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

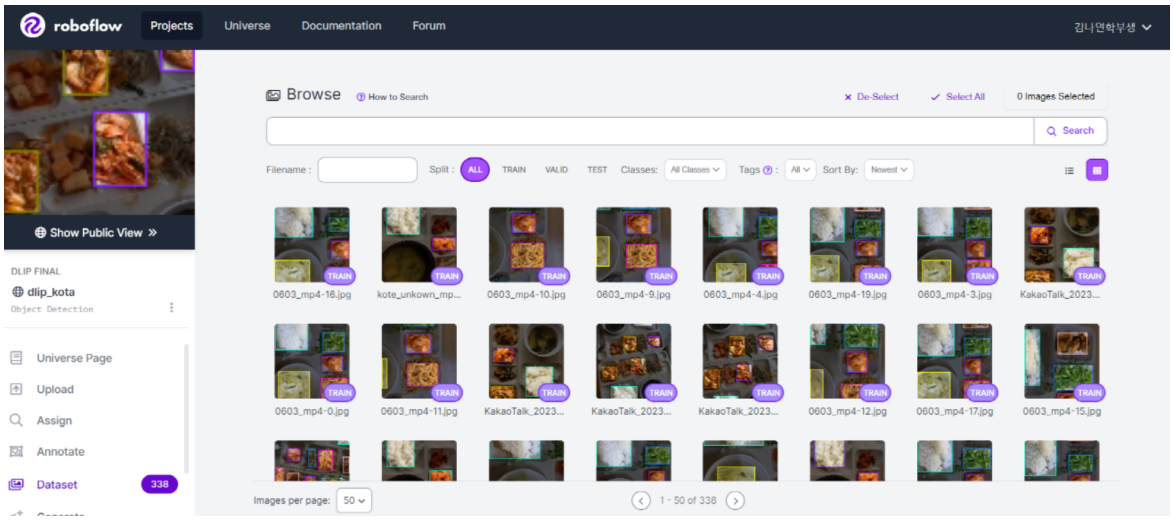
```
(base) PS C:₩Users₩navy8> nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:41:10_Pacific_Daylight_Time_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

Check my CUDA version is 11.8.

DATASET

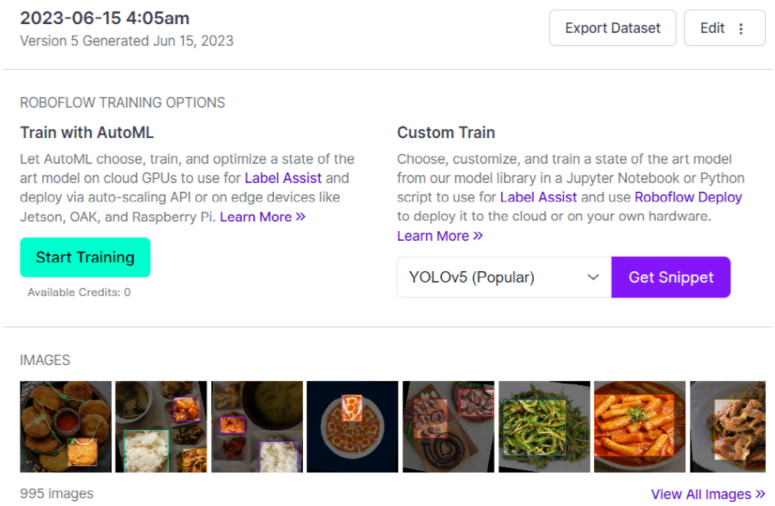
Labeling

- Roboflow



Roboflow is web site that can generate custom data. We made bounding boxes for each menus. In addition, using this program, we can augment images by flipping and rotating and so on so that a number of train images can be generated. For labeling, we collected 338 images, however, after augmenting, we could get total 998 train data set.

1. Upload images.
2. Start annotating and draw bounding boxes.
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 yolov5 file exists. Also, use the split_data.py file to divide the data into train and validation. Then, image files and labeling text file contain each information of detected classes will be saved respectively.

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

nc: 13
names: ['baechudoenjang-gug', 'cabbage', 'chicken mayo rice', 'dolnamul chojang', 'kimchi', 'kongnamulgug', 'meat sauce spaghetti',
'ssalbab', 'ssamjang', 'stir-fried pork', 'sundae', 'tteokbokki', 'vegetable croquettes']
```

If data are ready to be trained like this, training these data with yolov5 is next step.

Training data

In this project, we used yolov5l model, but depending on the computer specifications and GPUs, you can use the smaller size models s, m or the bigger one, x model too. We tried the x model too, but it took a long time, so we finally decided to use the l model. When you decide the model you will use, you need to start to train data now. Here is the code of training below.

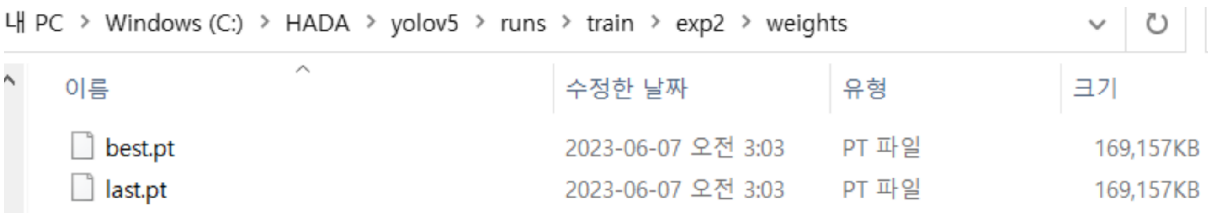
```
python train.py --img 640 --batch 4 --epochs 50 --data calories.yaml --weights yolov5l.pt
```

You need to select the image, batch, epochs size for training and in our case, we set the size of image to 640, 4 for batch, and 10 epochs. Also, you need to double check if the right path of yaml data file has written in the code. Selecting appropriate batch size is also important too because if too large size is selected, there will be an error about this. We tried to apply various epochs like 10, 50, and 100, and we chose 50 epochs for our train. Due to the long running time of epoch 100 and low accuracy of epoch 10, we selected 50 as epochs. So, numerous trials and errors are unavoidable in this perspective. The code below is about training only 1 epoch using yolov5l model.

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
        0/0      3.35G    0.05822    0.02677     0.0472          4        640: 100%|██████████| 131/131 [02:38<00:00,  1.21s/it]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████| 8/8 [00:09<00:00,  1.20s/it]
                   all         58         58    0.00336      0.962     0.0899     0.0333

1 epochs completed in 0.048 hours.
Optimizer stripped from runs\train\exp9\weights\last.pt, 92.9MB
Optimizer stripped from runs\train\exp9\weights\best.pt, 92.9MB

Validating runs\train\exp9\weights\best.pt...
Fusing layers...
Model summary: 267 layers, 46172898 parameters, 0 gradients, 107.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████| 8/8 [00:09<00:00,  1.24s/it]
                   all         58         58    0.00336      0.962     0.0895     0.0325
    baechudoenjang-gug         58          3    0.00285          1     0.0319     0.0131
               cabbage         58          3    0.00234          1     0.0103    0.00709
     chicken mayo rice         58          5     0.0043          1     0.0539     0.0199
      dolnamul chojang         58          4    0.00219          1      0.139     0.0409
                kimchi         58          5    0.00358          1     0.0494     0.0206
          kongnamulgug         58          7     0.0048          1      0.323     0.0933
  meat sauce spaghetti         58          9    0.00677          1     0.0888     0.0335
               ssalbab         58          4    0.00408          1    0.00664     0.0023
              ssamjang         58          4    0.00355          1     0.0222    0.00681
       stir-fried pork         58          5    0.00355          1      0.163     0.0647
                sundae         58          4    0.00253          1     0.0815     0.0147
            tteokbokki         58          3     0.0023          1      0.029    0.00661
  vegetable croquettes         58          2   0.000806        0.5      0.165     0.0991
Results saved to runs\train\exp9
```

Testing data

After train the data, in yolov5/runs/train folder, best.pt and last.pt files will be saved to the folder named exp. The best.pt file should be used for test because using last.pt file can cause overfitting. This is because last.pt file is saved when all traiining is done.

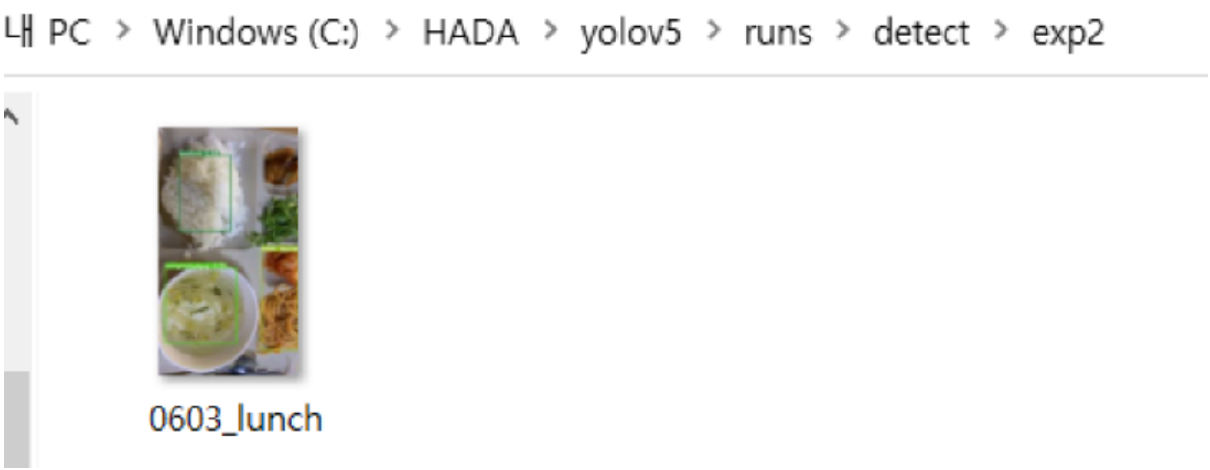


Run this test code in terminal.

```
python detect.py --weights runs/train/exp2/weights/best.pt --img 640 --conf 0.2 --source data/images/0603_lunch.mp4
```

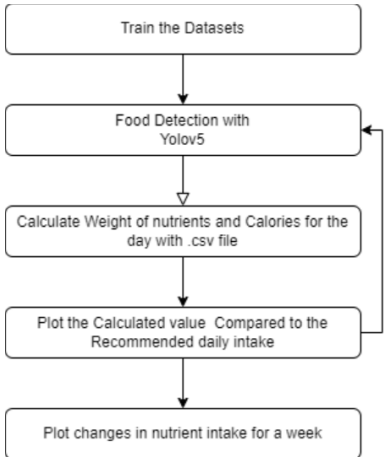
If you run this code in vs code, your result file will be saved in yolov5/runs/detect folder named with exp2.

```
video 1/1 (468/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 meat sauce spaghetti, 2 ssalbabs, 117.7ms
video 1/1 (469/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 meat sauce spaghetti, 2 ssalbabs, 125.7ms
video 1/1 (470/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 meat sauce spaghetti, 2 ssalbabs, 121.7ms
video 1/1 (471/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 121.6ms
video 1/1 (472/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 117.7ms
video 1/1 (473/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 114.7ms
video 1/1 (474/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 118.6ms
video 1/1 (475/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 118.7ms
video 1/1 (476/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 118.7ms
video 1/1 (477/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 122.7ms
video 1/1 (478/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 122.7ms
video 1/1 (479/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 115.7ms
video 1/1 (480/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 118.7ms
video 1/1 (481/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 115.7ms
video 1/1 (482/482) C:\HADA\yolov5\data\images\0603_lunch.mp4: 640x384 1 kongnamulgug, 1 ssalbab, 119.7ms
Speed: 1.0ms pre-process, 119.0ms inference, 2.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\exp13
```



2. Algorithm

Flow Chart



1. Detet Food with Yolov5.
2. Calculate the composition of calories and nutrients.
3. Plot the calories and nutrients of food consumed during the day.
4. Plot the calories and nutrients of food consumed during the accumulated data for a week.

Detect Foods with Yolov5.

Detect objects with food photo data trained using yolov5. If cv2 is used, it is input as BGR, so BGR must be changed to RGB for detection.

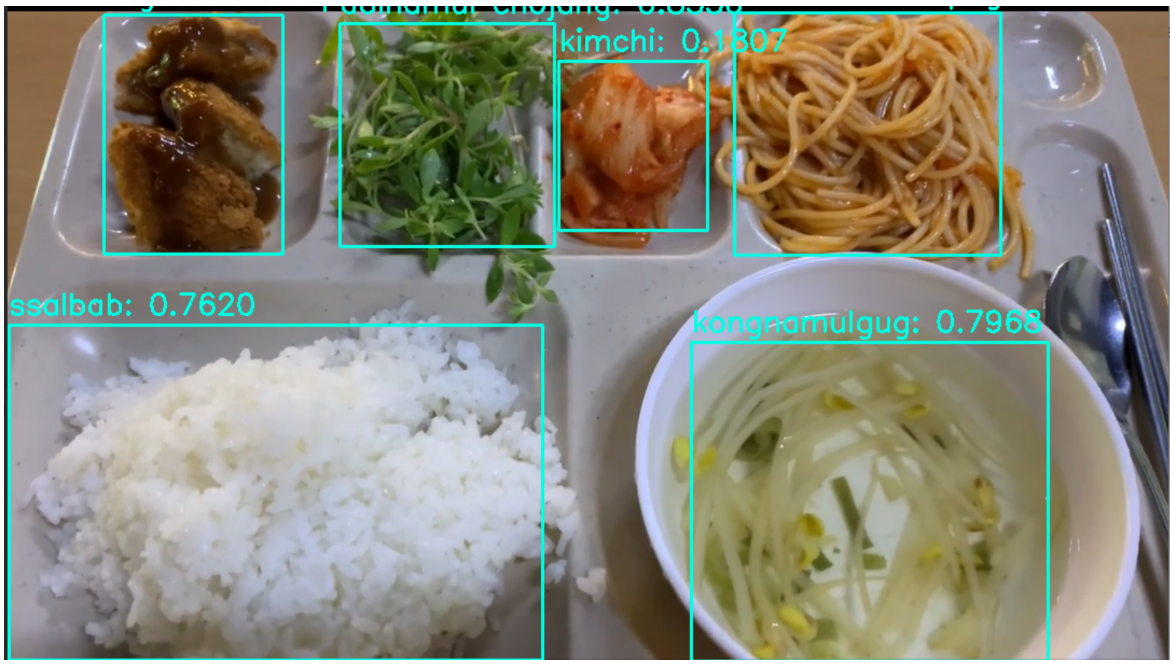
```
def detect_objects(image):

  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

  model.conf = 0.1
  results = model(image)

   # process result
  process_results(results, image)

  # image show
  image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
  resized_image = cv2.resize(image, None, fx=1, fy=1, interpolation=cv2.INTER_LINEAR)
  cv2.imwrite("Detection_image.png", resized_image)
```

```
  return 'Detection_image.png'
```

Calculate calories and nutrients of the food.

The information on calories, carbohydrates, protein, and fat of the detected foods is loaded from a .csv file and the values of the variables total_caloreis, total_protein, total_carbohydrates, and total_fat are updated.

- Result Image



- Code

```
def process_results(results, image):

    global total_caloreis
    global total_carbohydrates
    global total_protein
    global total_fat


    # Get Object Information from Results
    objects = results.pandas().xyxy[0]
    # Output object information
    print(objects)

    # Draw object information on images
    for _, obj in objects.iterrows():
        label = obj['name']
        confidence = obj['confidence']
        x_min = int(obj['xmin'])
        y_min = int(obj['ymin'])
        x_max = int(obj['xmax'])
        y_max = int(obj['ymax'])

        #Calculate Calories and Nutrients of Detected Objects
        if label in df['food name'].values:
            food_data = df[df['food name'] == label].iloc[0]
            caloreis = food_data['energy(kcal)']
            carbohydrates = food_data['carbohydrates(g)']
            protein = food_data['protein(g)']
            fat = food_data['fat(g)']

            total_caloreis += caloreis
            total_carbohydrates += carbohydrates
            total_protein += protein
            total_fat += fat
```

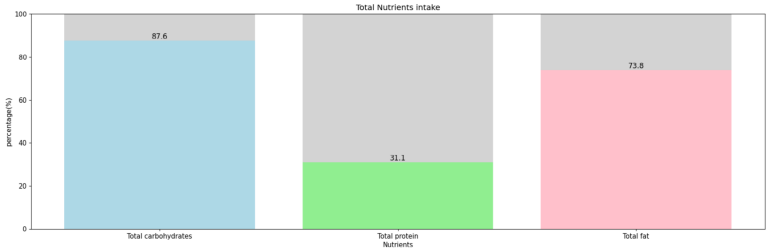
```
        cv2.rectangle(image, (x_min, y_min), (x_max, y_max), lime_green, 2)
        cv2.putText(image, f'{label}: {confidence:.4f}', (x_min, y_min - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, lime_green, 2)
```

Plot the calories and nutrients of food consumed during the day.

1. Plot the calories during the day.

Based on the daily recommended intake of 130 g, 50 g, and 25 g of carbohydrates, protein, and fat, the percentage of nutrients consumed is calculated and plotted in a bar graph.

- Result Image



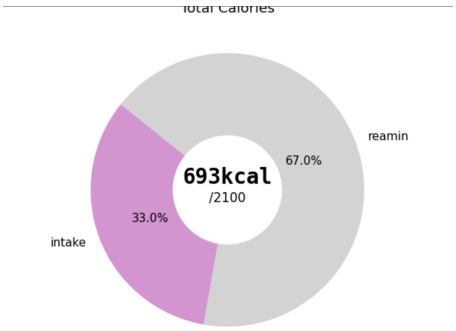
- Code

```
def total_nutrients(total_carbohydrates, total_protein, total_fat):

  values=[]
  ratios=[]
  remain=[]

  '''
  Plot Total nutrient
  '''
  values = [total_carbohydrates, total_protein, total_fat]
  recommended_values = [130, 55, 25]
  nutrients = ['Total carbohydrates', 'Total protein', 'Total fat']

 #Calculate the percentage of nutrients consumed based on the recommended calorie intake
  ratios = [(v / r)*100 for v, r in zip(values, recommended_values)]
  remain = [100-per for per in ratios]
  color=['lightblue','lightgreen','pink']
  plt.clf()
  bar = plt.bar(nutrients,ratios,color=color)
  plt.bar(nutrients,remain,bottom = ratios, color='lightgray')

# Convert graph to image and output
  fig_nutrients = plt.gcf()
  plt.title('Total Nutrients intake')
  plt.xlabel('Nutrients')
  plt.ylabel('percentage(%)')


# Insert values into the graph
  for rect in bar:
      height = rect.get_height()
      plt.text(rect.get_x() + rect.get_width()/2.0, height, '%.1f' % height, ha='center', va='bottom', size = 12)

  plt.ylim(0, 150)
  plt.tight_layout()
  plt.savefig("Total nutrient.png")
  plt.close()

  return 'Total nutrient.png'
```

2.Plot the nutrients during the day.

Based on the recommended daily calorie intake of 2100 kcal, the percentage of calories consumed is displayed on a circle graph.

- Result Image



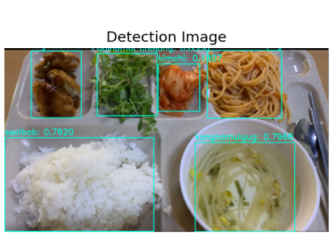
- Code

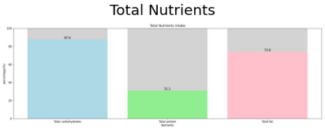
```
def total_calory_oneday(total):

  #Calculate total calories consumed per day

  oneday_caloreis=[]
  oneday_cal = total
  remain_total_cal = 2100 - oneday_cal
  oneday_caloreis= [oneday_cal,remain_total_cal]
  labels = ['intake', 'reamin']
  pie_colors = ['#d395d0', 'lightgray']
  wedgeprops={'width': 0.7, 'edgecolor': 'w', 'linewidth': 5}

  plt.clf()
  fig_calories = plt.gcf()

  # Setting wedgeprops to empty the middle part
  wedgeprops = {'width': 0.6}
  plt.pie(oneday_caloreis, labels=labels, autopct='%.1f%%', startangle=260, counterclock=False, colors=pie_colors, wedgeprops=wedgepro
  plt.text(0, 0, f'{total_caloreis:.0f}kcal\n',fontname='monospace', fontsize=20, fontweight='bold',color='black', ha='center', va='ce
  plt.text(0, 0, f'\n/2100', fontsize=12, color='black', ha='center', va='center')
  plt.tight_layout()
  plt.title('Total Calories')
  plt.savefig("Total Calories.png")
  plt.close()
  return 'Total Calories.png'
```

3.Display Results

This function shows the recognized image, the result of plotting calories and nutrients at once.

- Result Image





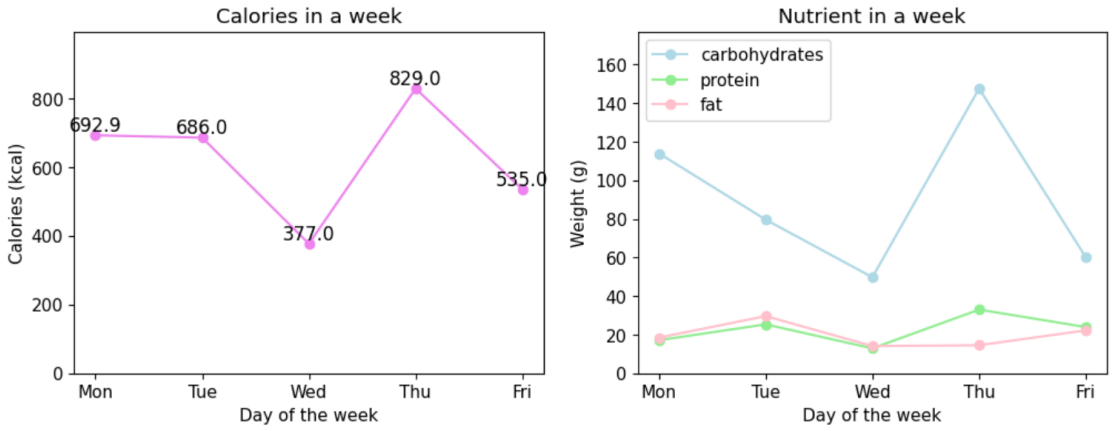
- Code

```
def plot_oneday(day, png_resized_image, png_total_nutrients, png_total_calory_oneday):

  plt.clf()
  # Create a plot
  fig_plot_oneday, axs = plt.subplots(1, 3, figsize=(18, 6))

  # Display each PNG file on the plot
  #Display detection image
  img_resized = Image.open(png_resized_image)
  axs[0].imshow(img_resized)
  axs[0].axis('off')
  axs[0].set_title("Detection Image")
  #Display nutrient
  img_nutrients = Image.open(png_total_nutrients)
  axs[1].imshow(img_nutrients)
  axs[1].axis('off')
  axs[1].set_title("Total Nutrients")
  #Display total calories
  img_calory = Image.open(png_total_calory_oneday)
  axs[2].imshow(img_calory)
  axs[2].axis('off')
  axs[2].set_title("Total Calories One Day")

  image = fig_plot_oneday
  display(image)

  outpath = "plot_oneday_" + day + ".png"
  plt.savefig(outpath)
```

Plot the calories and nutrients of food consumed during the accumulated data for a week.

Plot the total daily intake of calories, carbohydrates, protein, and fat for one week. Through this graph, you can easily figure out the changes in nutrients and calories consumed throughout the day.

- Result Image



- Code

```
def plot_week(week_nutrient):

  plt.clf()
  # 1x2 subplot
  fig_plot_week, axs = plt.subplots(1, 2, figsize=(12, 4))
  '''
  Calories in a week
  '''

  x = week_nutrient['date']
  y = week_nutrient['calories']
```

```
max_value = week_nutrient['calories'].max()
axs[0].plot(x, y, 'o-', color='violet')
axs[0].set_ylim(0, max_value * 1.2)
for i in range(len(x)):
    height = y[i]
    axs[0].text(x[i], height + 0.25, '%.1f' % height, ha='center', va='bottom', size=12)
axs[0].set_title('Calories in a week')
axs[0].set_xlabel('Day of the week')
axs[0].set_ylabel('Calories (kcal)')
'''
Nutrient in a week
'''
y_nut = week_nutrient[['carbohydrates', 'protein', 'fat']]
axs[1].plot(x, y_nut['carbohydrates'], 'o-', color=color[0],label='carbohydrates')
axs[1].plot(x, y_nut['protein'], 'o-', color=color[1],label='protein')
axs[1].plot(x, y_nut['fat'], 'o-', color=color[2],label='fat')
axs[1].legend()
max_value_nut = y_nut['carbohydrates'].max()
axs[1].set_ylim(0, max_value_nut * 1.2)

axs[1].set_title('Nutrient in a week')
axs[1].set_xlabel('Day of the week')
axs[1].set_ylabel('Weight (g)')

display(fig_plot_week)
plt.close()
```

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

  Onee uses 4-5 seconds of video for each menu. Each frame is indexed and the foods detected in the frame are output as a .txt file. Accaracy is calculated based on the number of frames in the .txt file, setting frames that correctly detected objects as True, and frames that incorrectly recognized objects as False.

  - Video used

    https://youtube.com/shorts/OVogcwZex3Q?feature=share

- Result

	False(frame)	True(frame)	Accuracy
Chickenmayo Rice	5	190	97.4%
Sundae & Tteockboki	16	120	99.3%
Pizza (Peperoni & Cheeze)	12	146	92.4%
3 Jun, 든든한동	0	146	100%
Average			97.3 %

Analysis

In the 'Chickenmayo Rice' video, the object was correctly recognized in 190 frames out of a total of 195 frames, and 5 frames were incorrectly recognized, resulting in 97.4% accuracy. In the same way as above, 'Sundae & Tteockboki' video has 99.3% accuracy, 'Pizza (peperoni&Cheeze' video) has 92.4% accuracy, and '3 Jun, strong movement' video has 100% accuracy. The average accuracy is 97.3%, which is much higher than the original target of 90% accuracy.

4. Conclusion

Through this project, we could reach the goal what we set as 90% in accuracy. By using roboflow, we customized food data with segmenting at first. In addition, by using yolov5, we trained these custom data and detected menus successfully. We selected three cafeterias '든든한동', '명성', and 'Yesod' for the test, and all of the menus were detected well. After finishing object detection, we made a program that shows up some menu's calories and their nutritional imformation. It shows the calories of food that users ate and the recommended quantities of calories, carbohydrates, proteins, and fat simultaneously. So the user can know what and how much they ate and what nutritions are needed more per a day. For the evaluation, we calculated accuracy based on the number of frames in the .txt file we exported from the code while setting frames that correctly detected objects as True, and frames that incorrectly recognized objects as False. The calculated accuracy was 97.3% and in the long-term perspective, if students in Handong use this program for a long time, they can make a good eating habit. It is the positive effect we expect for this program.

Appendix

Code for App program.

```
import pandas as pd
import numpy as np
import torch
import cv2
from google.colab.patches import cv2_imshow
from IPython.display import display
import sys

import os
from google.colab import drive
drive.mount('/content/drive')
root_path = '/content/drive/MyDrive/DLIP'  #change dir to your project folder
os.chdir(root_path)  #change dir
os.getcwd()

!pip install ultralytics

import cv2
from matplotlib import pyplot as plt
import time
from pathlib import Path
from PIL import Image

# Load YOLOv5 model
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = torch.hub.load('ultralytics/yolov5', 'custom', path='best.pt')

# Resetting the frame count
frame_count = 0
```

```
# Initialize the start time
start_time = 0

# DLIP_calories.csv file
df = pd.read_csv('DLIP_calories.csv')

#Initialize color
sky_blue = (255,191,0)
lime_green = (0,255,227)
color=['lightblue','lightgreen','pink']

total_calories =0
total_carbohydrates = 0
total_protein = 0
total_fat = 0

def detect_objects(image):
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
# setting confidenc = 0.1
  model.conf = 0.1
  results = model(image) # detect object


  process_results(results, image)

  # image show
  image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
  resized_image = cv2.resize(image, None, fx=1, fy=1, interpolation=cv2.INTER_LINEAR)
  cv2.imwrite("Detection_image.png", resized_image)

  return 'Detection_image.png'

def process_results(results, image):

    global total_caloreis
    global total_carbohydrates
    global total_protein
    global total_fat


    # Get Object Information from Results
    objects = results.pandas().xyxy[0]
    # Output object information
    print(objects)

    # Draw object information on images
    for _, obj in objects.iterrows():
        label = obj['name']
        confidence = obj['confidence']
        x_min = int(obj['xmin'])
        y_min = int(obj['ymin'])
        x_max = int(obj['xmax'])
        y_max = int(obj['ymax'])

        #Calculate Calories and Nutrients of Detected Objects
        if label in df['food name'].values:
            food_data = df[df['food name'] == label].iloc[0]
            caloreis = food_data['energy(kcal)']
            carbohydrates = food_data['carbohydrates(g)']
            protein = food_data['protein(g)']
            fat = food_data['fat(g)']

            total_caloreis += caloreis
            total_carbohydrates += carbohydrates
            total_protein += protein
            total_fat += fat

        cv2.rectangle(image, (x_min, y_min), (x_max, y_max), lime_green, 2)
        cv2.putText(image, f'{label}: {confidence:.4f}', (x_min, y_min - 10),cv2.FONT_HERSHEY_SIMPLEX, 0.9, lime_green, 2)

    print(f'process result:{total_caloreis}')

def total_nutrients(total_carbohydrates, total_protein, total_fat):

  values=[]
  ratios=[]
  remain=[]

  '''
  Plot Total nutrient
  '''
  values = [total_carbohydrates, total_protein, total_fat]
  recommended_values = [130, 55, 25]
  nutrients = ['Total carbohydrates', 'Total protein', 'Total fat']

 #Calculate the percentage of nutrients consumed based on the recommended calorie intake
  ratios = [(v / r)*100 for v, r in zip(values, recommended_values)]
```

```
  remain = [100-per for per in ratios]
  color=['lightblue','lightgreen','pink']
  plt.clf()
  bar = plt.bar(nutrients,ratios,color=color)
  plt.bar(nutrients,remain,bottom = ratios, color='lightgray')

# Convert graph to image and output
  fig_nutrients = plt.gcf()
  plt.title('Total Nutrients intake')
  plt.xlabel('Nutrients')
  plt.ylabel('percentage(%)')


# Insert values into the graph
  for rect in bar:
      height = rect.get_height()
      plt.text(rect.get_x() + rect.get_width()/2.0, height, '%.1f' % height, ha='center', va='bottom', size = 12)

  plt.ylim(0, 100)
  plt.tight_layout()
  plt.savefig("Total nutrient.png")
  plt.close()

  return 'Total nutrient.png'

def total_calory_oneday(total):

  #Calculate total calories consumed per day
  print(f'total:{total}')

  oneday_caloreis=[]
  oneday_cal = total
  remain_total_cal = 2100 - oneday_cal
  oneday_caloreis= [oneday_cal,remain_total_cal]
  labels = ['intake', 'reamin']
  pie_colors = ['#d395d0', 'lightgray']
  wedgeprops={'width': 0.7, 'edgecolor': 'w', 'linewidth': 5}


  plt.clf()
  fig_calories = plt.gcf()

  # Setting wedgeprops to empty the middle part
  wedgeprops = {'width': 0.6}
  plt.pie(oneday_caloreis, labels=labels, autopct='%.1f%%', startangle=260, counterclock=False, colors=pie_colors, wedgeprops=wedgepro
  plt.text(0, 0, f'{total_caloreis:.0f}kcal\n',fontname='monospace', fontsize=20, fontweight='bold',color='black', ha='center', va='ce
  plt.text(0, 0, f'\n/2100', fontsize=12, color='black', ha='center', va='center')
  plt.tight_layout()
  plt.title('Total Calories')
  plt.savefig("Total Calories.png")
  plt.close()
  return 'Total Calories.png'

def merge_images(image_paths):

    # Open the first image
    images = [Image.open(path) for path in image_paths]
    widths, heights = zip(*(i.size for i in images))

    # Calculate the size of the new image
    new_width = sum(widths)
    new_height = max(heights)

    # Create a new image
    new_image = Image.new('RGB', (new_width, new_height))

    # Paste the images onto the new image
    x_offset = 0
    for image in images:
        new_image.paste(image, (x_offset, 0))
        x_offset += image.width

    # Save the resulting image
    new_image.save('merge.jpg')

    return 'merge.jpg'

def open_file(input_path):

  global total_caloreis
  global total_carbohydrates
  global total_protein
  global total_fat

  total_caloreis = 0
  total_carbohydrates = 0
  total_protein = 0
```

```
  total_fat = 0

  if len(input_path) > 1:
    input_path = merge_images(input_path)
  else :
    input_path = input_path[0]

  # open video file
  if input_path.endswith('.mp4') or input_path.endswith('.avi'):

    vid = cv2.VideoCapture(input_path)

    #height, width, channels = frame.shape
    height = int(vid.get(cv2.CAP_PROP_FRAME_HEIGHT))
    width  = int(vid.get(cv2.CAP_PROP_FRAME_WIDTH))

    while True:
      ret, frame = vid.read()
      if not ret:
        print('video in not open')
        break

      resized_frame = detect_objects(frame)
      cv2_imshow(resized_frame)
      if vid.get(cv2.CAP_PROP_POS_FRAMES) == vid.get(cv2.CAP_PROP_FRAME_COUNT):
        break

      if cv2.waitKey(1) == ord('q'):
        break

      vid.release()

  else:
    img = cv2.imread(input_path) # open image file

    if img is None:
      print("The image is empty. Exiting debugging.")
      sys.exit()


    png_resized_image = detect_objects(img) # detect food

    png_total_nutrients = total_nutrients(total_carbohydrates, total_protein, total_fat) # plot onday nutrient

    png_total_calory_oneday = total_calory_oneday(total_caloreis) # plot onday caloeris

    return png_resized_image, png_total_nutrients, png_total_calory_oneday

def week_dfappend(week_nutrient, date, calories, carbohydrates, protein,fat):

  new_data = {
    'date': [date],
    'calories': [calories],
    'carbohydrates': [carbohydrates],
    'protein': [protein],
    'fat': [fat]
  }
  new_row = pd.DataFrame(new_data)
  week_nutrient = week_nutrient.append(new_row, ignore_index=True) # add the day nutrient information
  print(week_nutrient)

  return week_nutrient

def plot_oneday(day, png_resized_image, png_total_nutrients, png_total_calory_oneday):

  plt.clf()
  # Create a plot
  fig_plot_oneday, axs = plt.subplots(1, 3, figsize=(18, 6))

  # Display each PNG file on the plot
  #Display detection image
  img_resized = Image.open(png_resized_image)
  axs[0].imshow(img_resized)
  axs[0].axis('off')
  axs[0].set_title("Detection Image")
  #Display nutrient
  img_nutrients = Image.open(png_total_nutrients)
  axs[1].imshow(img_nutrients)
  axs[1].axis('off')
  axs[1].set_title("Total Nutrients")
  #Display total calories
  img_calory = Image.open(png_total_calory_oneday)
  axs[2].imshow(img_calory)
  axs[2].axis('off')
  axs[2].set_title("Total Calories One Day")

  image = fig_plot_oneday
```

```
  display(image)

  outpath = "plot_oneday_" + day + ".png"
  plt.savefig(outpath)

def plot_week(week_nutrient):

  plt.clf()
  # 1x2 subplot
  fig_plot_week, axs = plt.subplots(1, 2, figsize=(12, 4))
  '''
  Calories in a week
  '''

  x = week_nutrient['date']
  y = week_nutrient['calories']
  max_value = week_nutrient['calories'].max()
  axs[0].plot(x, y, 'o-', color='violet')
  axs[0].set_ylim(0, max_value * 1.2)
  for i in range(len(x)):
      height = y[i]
      axs[0].text(x[i], height + 0.25, '%.1f' % height, ha='center', va='bottom', size=12)
  axs[0].set_title('Calories in a week')
  axs[0].set_xlabel('Day of the week')
  axs[0].set_ylabel('Calories (kcal)')
  '''
  Nutrient in a week
  '''
  y_nut = week_nutrient[['carbohydrates', 'protein', 'fat']]
  axs[1].plot(x, y_nut['carbohydrates'], 'o-', color=color[0],label='carbohydrates')
  axs[1].plot(x, y_nut['protein'], 'o-', color=color[1],label='protein')
  axs[1].plot(x, y_nut['fat'], 'o-', color=color[2],label='fat')
  axs[1].legend()
  max_value_nut = y_nut['carbohydrates'].max()
  axs[1].set_ylim(0, max_value_nut * 1.2)

  axs[1].set_title('Nutrient in a week')
  axs[1].set_xlabel('Day of the week')
  axs[1].set_ylabel('Weight (g)')

  display(fig_plot_week)
  plt.close()

import matplotlib.image as mpimg

columns = ['date', 'calories', 'carbohydrates', 'protein', 'fat']
week_nutrient= pd.DataFrame(columns=columns)
# 객체 인식 수행
mon_input_path = ['0603.png']

mon_resized_image, mon_total_nutrients, mon_total_calory_oneday = open_file(mon_input_path)

plot_oneday("Mon",mon_resized_image, mon_total_nutrients, mon_total_calory_oneday)

week_nutrient = week_dfappend(week_nutrient,'Mon',total_caloreis,total_carbohydrates,total_protein,total_fat)

Tue_input_path = ['chickenmayo.jpg']
open_file(Tue_input_path)
Tue_resized_image, Tue_total_nutrients, Tue_total_calory_oneday = open_file(Tue_input_path)

plot_oneday("Tue",Tue_resized_image, Tue_total_nutrients, Tue_total_calory_oneday)

week_nutrient = week_dfappend(week_nutrient,'Tue',total_caloreis,total_carbohydrates,total_protein,total_fat)

Wed_input_path = ['cheeze.jpg','kochi.jpg']
open_file(Wed_input_path)
Wed_resized_image, Wed_total_nutrients, Wed_total_calory_oneday = open_file(Wed_input_path)

plot_oneday("Wed",Wed_resized_image, Wed_total_nutrients, Wed_total_calory_oneday )

week_nutrient = week_dfappend(week_nutrient,'Wed',total_caloreis,total_carbohydrates,total_protein,total_fat)

Thu_input_path = ['tteockbocki.jpg','sundae.jpg']
open_file(Thu_input_path)
Thu_resized_image, Thu_total_nutrients, Thu_total_calory_oneday = open_file(Thu_input_path)

plot_oneday("Thu",Thu_resized_image, Thu_total_nutrients, Thu_total_calory_oneday )

week_nutrient = week_dfappend(week_nutrient,'Thu',total_caloreis,total_carbohydrates,total_protein,total_fat)

import ipywidgets as widgets

from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode
```

```
def take_photo(filename='photo.jpg', quality=0.8):
  #Start writing javascript
  js = Javascript('''
    async function takePhoto(quality) {

      //Create a div (space)
      const div = document.createElement('div');

      //create button
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      //create video
      const video = document.createElement('video');
      //video shape square
      video.style.display = 'block';
      //Calling up the camera (webcam)
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      //Add child space below the div
      document.body.appendChild(div);
      //put video in space
      div.appendChild(video);
      //Connect video and webcam
      video.srcObject = stream;
      //await -> asynchronous processing (related to thread) (async and set)
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      //create canvas
      const canvas = document.createElement('canvas');
      //fit to size
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      //draw image
      canvas.getContext('2d').drawImage(video, 0, 0);
      //turn off video
      stream.getVideoTracks()[0].stop();
      //delete div
      div.remove();
      // return file address
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  #Send the return value of #javascript code execution to Python
  data = eval_js('takePhoto({})'.format(quality))
  # Save data as base64 when saving data from web browser
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename

from IPython.display import Image
try:
  filename = take_photo()
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or if they do not
  # grant the page permission to access it.
  print(str(err))

from PIL import Image
# 객체 인식 수행
Fri_input_path =[format(filename)]
open_file(Fri_input_path)
Fri_resized_image, Fri_total_nutrients, Fri_total_calory_oneday = open_file(Fri_input_path)

plot_oneday("Fri",Fri_resized_image, Fri_total_nutrients, Fri_total_calory_oneday )

week_nutrient = week_dfappend(week_nutrient,'Fri',total_caloreis,total_carbohydrates,total_protein,total_fat)

plot_week(week_nutrient)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Demo video

DLIP_Project_CalculateCalories_2023

https://youtu.be/1rCme4aukLY

Recommeded Daily Intake for 20s Woman.

Calories(kcal)	Calbohydrates(g)	Protein(g)	Fat(g)
2100	130	55	25

Source : 식품의약품안전처

http://www.foodsafetykorea.go.kr/foodcode/01_03.jsp?idx=12131