

2022-1 기계학습과 인공지능

# 음식 이미지 인식 후, 음식의 칼로리 추정

지능시스템학과

김준용, 길다영, 장재훈

# CONTENTS

- 프로젝트 설명
- NUVI lab
- Work flow
- Dataset
- Food detection
- Volume Estimation
- Calorie Estimation
- 향후 계획
- Reference

- 목표 : 음식 이미지 인식 후, 음식의 칼로리 추정 결과

- 기대 효과

- 일일이 음식의 종류와 양을 기입하는 불편함 없이, **사진 한장으로 칼로리 계산** 가능.

## 기존 칼로리 계산 앱들의 한계

- 기존의 앱들은 식단에 나타난 **모든 음식들의 종류와 양을 직접 기입**해야 하는 불편함이 있음.
- 실제로 먹은 양이 정확히 몇 g 인지 예상하기 어렵기 때문에 **양 기입 시 오차가 많이 발생할** 수 있음.

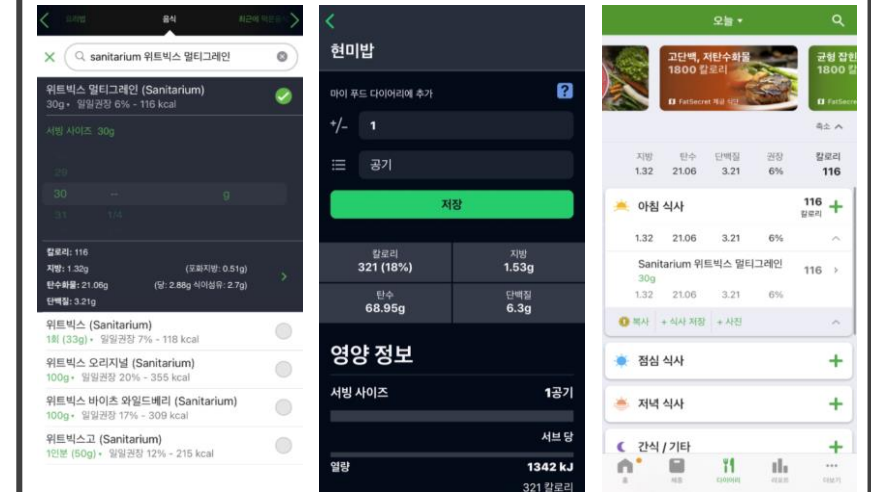
## 이 프로젝트의 장점

- 일일이 음식의 종류와 양을 기입하는 불편함 없이, **사진 한장으로 칼로리 계산** 가능.
- **부피를 계산하여 칼로리를 계산**하기 때문에 보다 정확함.

- 연구중인 “3D object detection” 에 도움이 될 것이라 기대.

- 연구중인 “human motion detection”의 detection 부분에 도움이 될 것이라 기대.

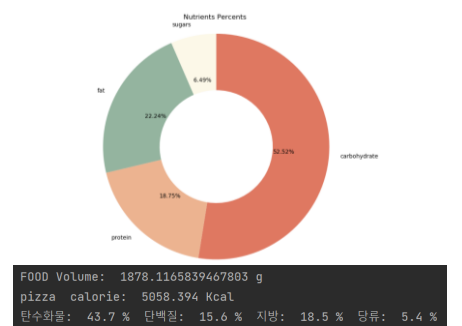
## 기존 칼로리 계산 앱 - FatSecret



Input

Results

## 프로젝트 결과



Input image

Results

## 누비랩의 인공지능(AI) 음식 스캐너 '누비 스캔'



[CES 2022] 음식 사진 촬영하면 AI가 영양 정보를 알려 준다



누비랩, 'AI 푸드 스캐너'로 CES2021 혁신상 수상

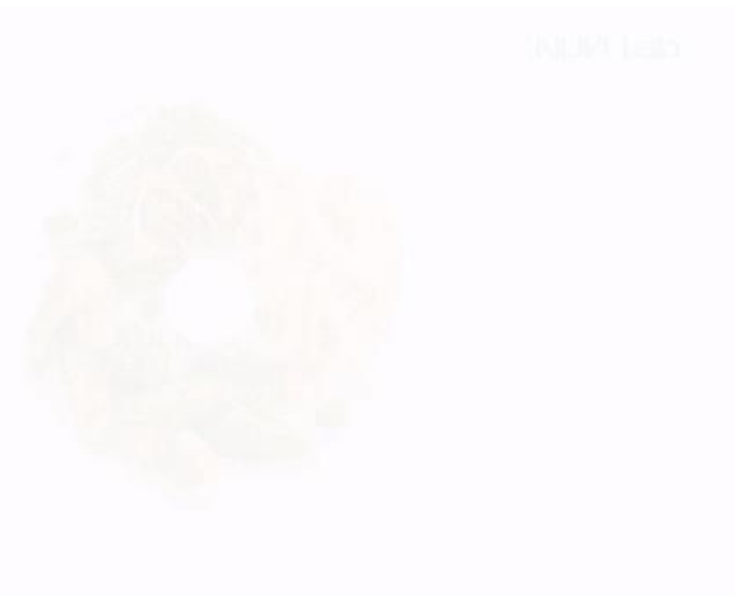


[UP! START] 누비랩, 북유럽 시장 공략...AI로 음식물 쓰레기 절감



AI로 음식물 쓰레기 줄이고 영양관리까지 1초만에 '싹'

- 식당 이용자들의 배식량·섭취량·잔반량 등을 측정해 데이터화하고, 이를 기반으로 개개인의 음식 소비량을 예측해 다음에 준비해야 할 적정 음식량을 계산해주는 방식
- 배식구와 퇴식구에 각각 **AI 기능을 탑재한 스캐너와 센서**를 설치한다.
- 이용자가 스캐너 밑에 식판을 가져다 대면 카메라 센서가 1초 이내에 **95%가 넘는 정확도로** 식판에 담긴 **음식의 영양 성분과 칼로리**를 알려준다.
- 각종 음식 이미지를 학습한 **AI가 여러 음식의 종류를 인식**하고, **센서는 음식의 양을 분석**한다.

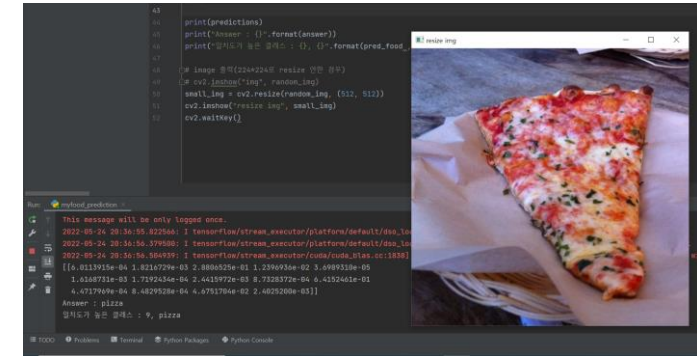


# Work flow

## 1. Input image

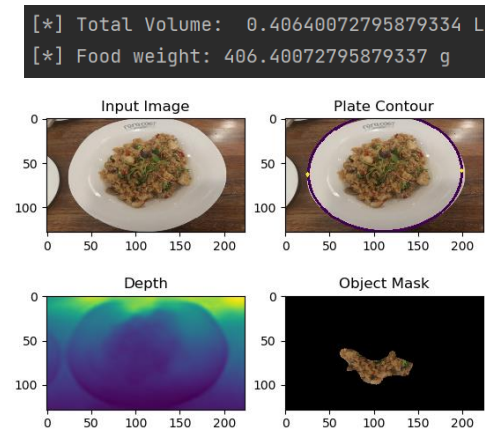


## 2. Food Detection



Convolution layer와 Pooling layer로 구성된 **CNN 모델** 생성 후 fit()를 통해 모델 학습

## 3. Volume Estimation

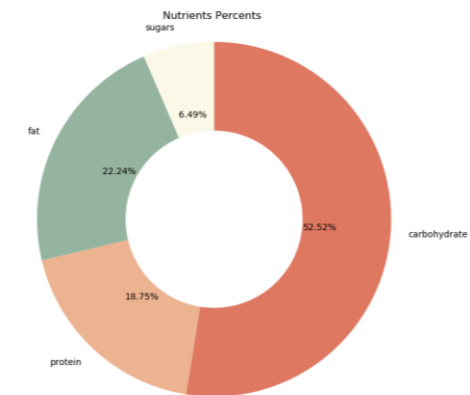


Depth Estimation Network & Segmentation Network & Point Cloud-to-Volume Algorithm  
이용하여 부피 계산

## 4. Calorie calculation

```
dict = {'food': [g, kcal, 탄수화물, 단백질, 지방, 염류]}  
food_dic = {'apple': [100, 57, 14.36, 0.2, 0.03, 11.14], 'bibimbab': [450, 537, 84.76, 30.89, 19.44, 0.22],  
'bulgogi': [300, 486, 14.77, 48.38, 23.86, 0], 'chocolate_cake': [100, 420, 43.6, 5.3, 25, 23],  
'fried_egg': [46, 89, 0.43, 6.24, 6.76, 0.38], 'hamburger': [93, 270, 26.8, 14.6, 11, 0],  
'jjajangmyeon': [450, 785, 129.43, 26.01, 19.98, 6.56], 'kinbap': [200, 318, 57.6, 7.3, 6.5, 0.5],  
'kimchi_stew': [200, 121, 6.47, 7.51, 7.57, 1.77], 'pizza': [150, 404, 43.7, 15.6, 18.5, 5.4],  
'ramen': [369, 515, 87.8, 11, 15.8, 0], 'sandwich': [146, 352, 33.35, 20.69, 15.48, 0],  
'steak': [200, 686, 33.79, 29.3, 46.22, 0], 'sushi': [125, 179, 37.38, 5.37, 0.52, 0]}
```

FOOD Volume: 1878.1165839467803 g  
pizza calorie: 5058.394 Kcal  
탄수화물: 43.7 % 단백질: 15.6 % 지방: 18.5 % 당류: 5.4 %



추정된 부피 기반으로 **칼로리표**를 이용하여 칼로리와 영양 성분 계산



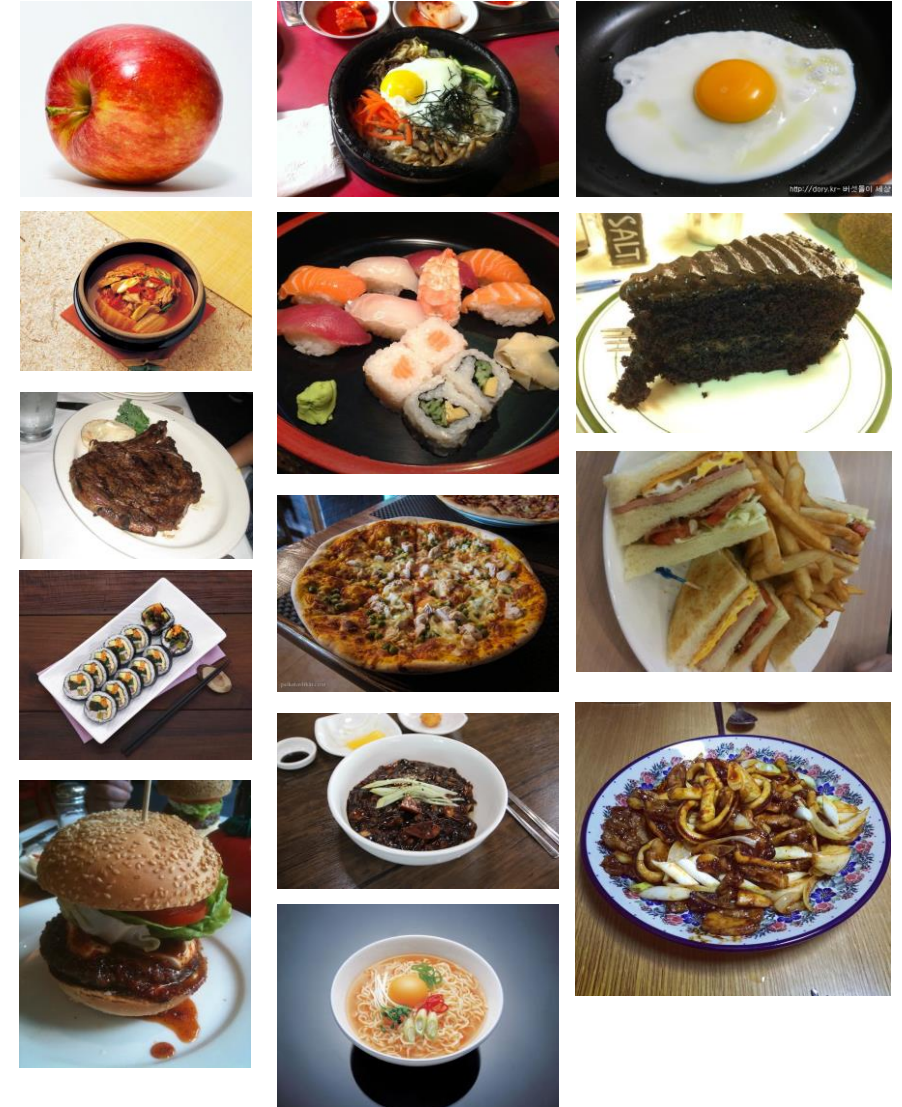
## ■ Data Preprocessing

### Dataset

1. Tensorflow\_datasets에 있는 **음식 101종(종별 1천장)** 중에서 7종류 음식 선별
2. AI hub에 있는 **한국 음식 150종(종별 1천 장)** 중에서 8종류 음식 선별
3. **음식 종류** : 사과, 비빔밥, 불고기, 초콜릿 케이크, 계란 후라이, 햄버거, 짜장면, 김밥, 김치찌개, 피자, 라면 샌드위치, 스테이크, 초밥

### Tensorflow Data split

- Tensorflow **101개의 식품 카테고리**(각 1000장)의 **101,000개**의 데이터
- 공식 홈페이지 코드를 통해 **train : test = 7.5 : 2.5**로 나눈 것을 확인
- tar 압축해제 시 txt를 통해 train, test set이 정리되어 있음



# Food Detection Model

## ■ CNN model 구성

### CNN Model

- Layer 개수와 종류, 과대 적합 방법을 바꾸어 실험하며 성능이 개선된 CNN 모델 생성
- 7개의 **Convolution layer**와 5개의 **Pooling layer**로 구성
- **Dropout**을 통해 과대적합 방지
- **Input\_size**를 224\*224로 변경 (VGG16 구조에서 착안)

### 학습 결과

- batch\_size는 150, epochs는 300으로 설정
- 학습 결과 **Loss: 130.82%, ACC : 82.83%**가 나옴

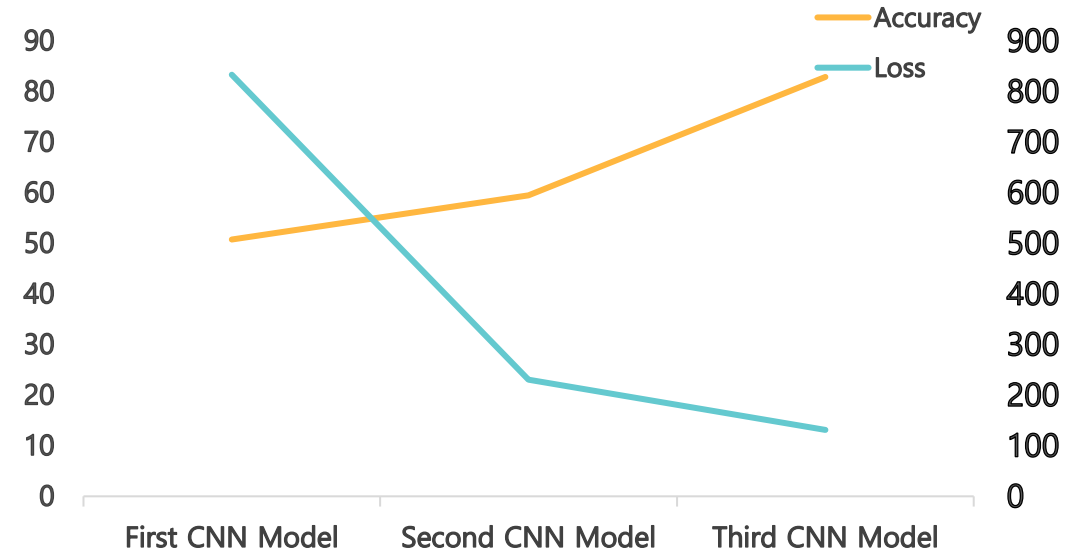
### Problem

- Food Detection에 필요한 환경과 Volume Estimation에 필요한 **환경이 달라 한번에 실행하기 어려움.**
  - Food Detection: tensorflow 2.x 버전
  - Volume Estimation: tensorflow 1.x 버전

<VGG16 Model>

VGGNet은 옥스퍼드 대학의 연구팀 VGG에 의해 개발된 모델로, 2014년 이미지넷 이미지 인식 대회에서 준우승을 한 모델. VGG16은 13개의 Convolution Layer와 3개의 Fully Connected Layer인 16층으로 구성되어 있으며 Convolution kernel\_size는 가장 작은 3 x 3으로 stride는 1로 고정했으며 MaxPooling은 2 x 2에 stride 2로 고정.

### CNN model Loss & Accuracy 변화



# Food Detection Model

## ■ Modified CNN model

### 수정된 CNN model

- 3개의 **Convolution layer**와 **Pooling layer**로 구성
- Keras를 통한 이미지 로드 및 전처리 방법을

image\_dataset\_from\_directory에서

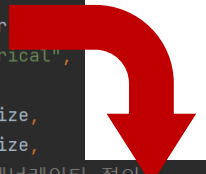
**image.ImageDataGenerator**로 변경

### 수정된 CNN model Result

- batch\_size는 150, epochs는 100으로 설정
- 학습 결과 **Loss: 108.18%, ACC : 73.42%**가 나옴
- 정확도가 약 10% 낮게 나타남
- GPU동작을 못하여 **학습 시간이 tf2에 비해 2-3배** 늘어남

```
# 제너레이터 정의
def generator(train_dir, batch_size, image_size):
    train_set = image_dataset_from_directory(directory=train_dir,
                                             label_mode="categorical",
                                             color_mode="rgb",
                                             batch_size=batch_size,
                                             image_size=image_size,
                                             shuffle=True, # 제너레이터 정의
                                             seed=42,
                                             validation_split=0.2,
                                             interpolation="bilinear",
                                             subset="train")
    val_set = image_dataset_from_directory(directory=train_dir,
                                           label_mode="categorical",
                                           color_mode="rgb",
                                           batch_size=batch_size,
                                           image_size=image_size,
                                           shuffle=False,
                                           validation_split=0.2,
                                           interpolation="bilinear",
                                           subset="validation")
    return train_set, val_set

Tf2 < image_dataset_from_directory >
```



```
train_argmentation = ImageDataGenerator(
    rescale=1./255, # 이미지 0-1 사이의 값으로 변경
    rotation_range=10,
    zoom_range=0.2,
    shear_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    vertical_flip=False
)
val_argmentation = ImageDataGenerator(
    rescale=1./255
)
augment_size = 150
train_generator = train_argmentation.flow_from_directory(
    train_dir,
    target_size=(224,224),
    batch_size=augment_size,
    shuffle=True,
    class_mode='categorical'
)
val_generator = val_argmentation.flow_from_directory(
    val_dir,
    target_size=(224,224),
    batch_size=augment_size,
    shuffle=False,
    class_mode='categorical'
)

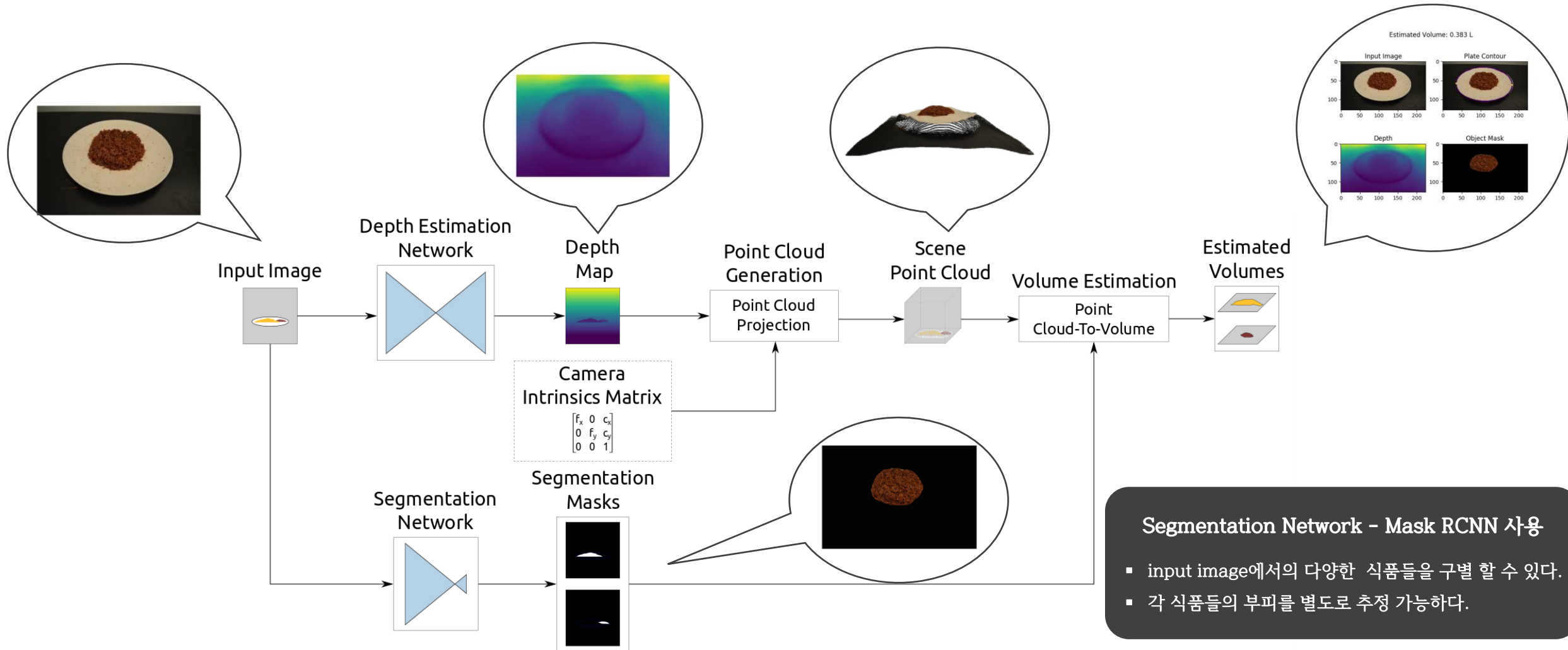
Tf1 < image.ImageDataGenerator >
```



# Volume Estimation

## Volume Estimation work flow

- Depth Estimation Network를 이용하여 **depth map**과 **point cloud**를 생성.
- Segmentation Network와 생성된 Point Cloud를 이용하여 **부피 계산**



# Calorie Estimation

## Calorie Estimation Results

- Food 101 데이터 셋의 음식 종류와 g수에 따른 칼로리, 영양성분을 나타내기 위해 `dict = {'food' : [g, kcal, 탄수화물, 단백질, 지방, 당류]}` 형태의 딕셔너리 생성
- (추정한 `volume / g`) \* `kcal` 로 input image의 최종 칼로리 계산

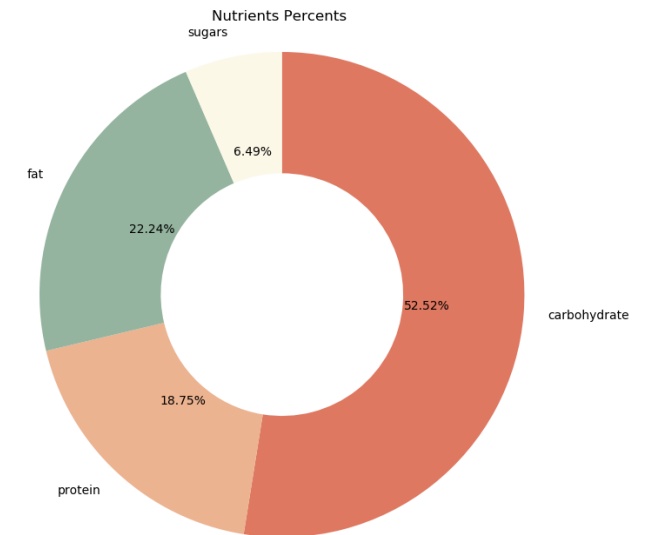


```
# dict = {'food' : [g, kcal, 탄수화물, 단백질, 지방, 당류]}
food_dic = {'apple': [100, 57, 14.36, 0.2, 0.03, 11.14], 'bibimbab': [450, 537, 84.76, 30.89, 19.44, 0.22],
            'bulgoggi': [300, 486, 14.77, 48.38, 23.86, 0], 'chocolate_cake': [100, 420, 43.6, 5.3, 25, 23],
            'fried_egg': [46, 89, 0.43, 6.24, 6.76, 0.38], 'hamburger': [93, 270, 26.8, 14.6, 11, 0],
            'jajangmyeon': [450, 785, 129.41, 26.81, 19.98, 6.56], 'kimbap': [200, 318, 57.6, 7.3, 6.5, 0.5],
            'kimchi_stew': [200, 121, 6.47, 7.51, 7.57, 1.77], 'pizza': [150, 404, 43.7, 15.6, 18.5, 5.4],
            'ramen': [369, 515, 87.8, 11, 15.8, 0], 'sandwich': [146, 352, 33.35, 20.69, 15.48, 0],
            'steak': [200, 686, 33.79, 29.3, 46.22, 0], 'sushi': [125, 179, 37.38, 5.37, 0.52, 0]}

foods = paths.split('/')
food = foods[-2]

calorie = (volume / food_dic[food][0]) * food_dic[food][1]
```

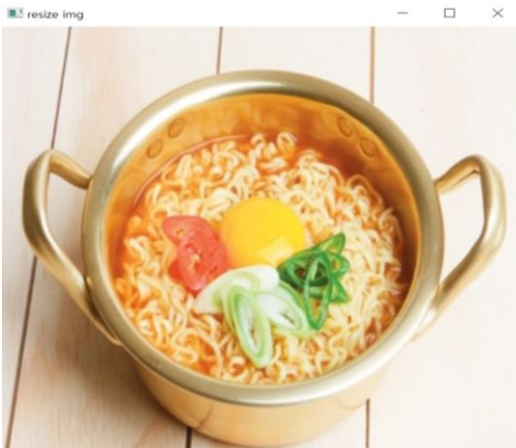
```
FOOD Volume: 1878.1165839467803 g
pizza calorie: 5058.394 Kcal
탄수화물: 43.7 % 단백질: 15.6 % 지방: 18.5 % 당류: 5.4 %
```



### ■ 향후 계획

- GPU환경에서 학습시키기.
- Food detection 정확도 개선
- 음식 종류 세분화하여 detection 해보기 (ex. 새우초밥, 연어초밥 등)
- 부피 측정 시, 정확도 개선
  - 현재는 detection한 음식 뿐 아니라 그 외 음식까지 부피 측정됨.

Detection 잘 안된 경우



음식 정답 : ramen  
예측한 음식 : 4, fried\_egg  
2022-05-25 10:47:18.648029: I tensor

Detection 잘 된 경우



Answer : pizza  
일치도가 높은 클래스 : 9, pizza

- [https://github.com/AlexGraikos/food\\_volume\\_estimation](https://github.com/AlexGraikos/food_volume_estimation)
- [https://github.com/AlexGraikos/food\\_volume\\_estimation](https://github.com/AlexGraikos/food_volume_estimation)
- <https://hangjastar.tistory.com/188?category=980564>

2022-1 기계학습과 인공지능

Thank You

지능시스템학과

김준용, 길다영, 장재훈