



Fridgeat

Fridge와 Eat에 합성어



Made with Gamma

목차

1 프로젝트 최종 예상 결과

2 프로젝트 배경

3 프로젝트 수행 절차 및 방법

4 프로젝트 팀 구성 및 역할

5 프로젝트 수행 결과

6 느낀점

1. 프로젝트 배경

선정배경

우리는 요리 후 남은 식재료는 냉장고에 보관한다. 하지만, 이 후, 그 남은 식재료를 사용하여, 어떤 요리를 할 수 있을지 판단이 서지 않아 다시 방치하게 되거나 다른 식재료를 추가로 구매하여 요리를 한다. 그렇게 다시 남는 식재료가 생기고, 사용되지 못한 식재료들은 다시 장기간 방치되어 결국 폐기되는 상황이 반복된다.

"요리 지식 부족"

- 식재료 활용 능력 부족
- 레시피 검색/검증으로 인한 시간 소요



개발목적

요리에 필요한 지식을 대신할 수 있는 AI App을 만들어, 식재료 활용에 도움을 주고, 생각하는 시간을 줄여 사람들이 보다 손쉽게 요리할 수 있도록 하여, 가정에서 발생하는 **음식물 폐기량**을 줄인다.

TV프로그램 "냉장고를 부탁해"



"연예인의 냉장고에 방치되는 여러가지 식재료들을 활용해서 요리를 만든다라는 컨셉이었지만, 연예인들에게만 국한되지 않고 대부분의 가정에서 일어나는 흔한 현상이라는 점에서 공감대를 불러일으켜 사람들의 큰 인기를 끌었다."

App의 특징 및 차별성

기존에 존재하는 유사 APP



만개의 레시피



우리의 식탁



samsung Food

오이 콩나물 시금치

추천순 통합 레시피 키친가이드 상품 #스타일

총 41개 검색결과 정확도순 ▼

제목	설명	등급	시간	원작자
[간단 자취요리] 요청쇄도! 맛 있는 콩불, 콩나물 불고기 만들기	암무	★★★ 5.0 (421)	20분	우리의식탁
시금치 파스타			20분	우리의식탁

기존 앱들과의 차이점

'Fridgeat'은 유저가 단순히 휴대폰 카메라로 **식재료를 스캔**하여, 보유하고 있는 조리도구 및 식재료 내에서 요리 가능한 **레시피를 추천**한다.

강점 : 확장성

- 레시피의 정보를 보다 자세히 물어볼 수 있는 Chat Bot기능
- 커스텀 레시피 창작 및 공유
- 대체 식재료 추천

기대효과

- 식재료 효율화 증가 → 음식물 폐기량 감소 → 쓰레기 처리 공간문제 감소 → 쓰레기 처리 비용 감소 → **탄소 발자국 감소**
- 개인 요리 빈도 증가 → 간편음식(밀키트) & 배달음식 이용 감소 → **식비 감소 & 일회용품 사용량 감소**

2. 프로젝트 팀 구성 및 역할



조대희

Frontend

Flutter UI 구성

Flutter 기능 구현

GPT 프롬프트 엔지니어링



전재영

Frontend

GPT API Flutter 연동

실시간 객체 감지 모델 탑재

API key 관련 보안



신지만

Backend

Firebase Flutter 연동

Firebase 모델 연동

모델 지원



박영수

Model

데이터 수집 및 정제

데이터 EDA



서원영

Model

모델 학습

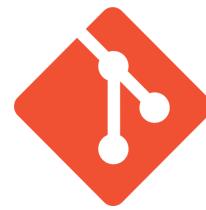
개발환경



Figma



Notion



git



기술스택



Flutter



Firebase



Tensorflow

3. 프로젝트 수행 절차 및 방법

주요 Task

데이터 설정

Firebase 서버구축 및
Flutter 연결

App UI 구성

GPT API 코딩

모델 학습

모델 테스트

모델 저장

1차 MVP



Made with Gamma

단계별 세부 Task

Task	계획 기간	활동
데이터 선정	10월 23일 ~ 24일	식재료 데이터 선정
데이터 수집	10월 24일 ~ 25일	Object Detection을 위한 식재료 데이터 수집
Firebase 서버 구축 및 Flutter 연동	10월 23일	YOLO, GPT4에서 얻은 결과를 저장하기 위한 서버 구축 및 Flutter 연동
App UI 구성	10월 23일 ~ 30일	Flutter UI구성
GPT API 관련 코딩	10월 23일 ~ 30일	GPT API 연동 및 프롬프트 엔지니어링
데이터 전처리	10월 25일 ~ 27일	데이터 전처리
모델 구현	10월 27일 ~ 30일	YOLO 모델구현
모델 학습	10월 30일 ~ 11월 24일	식재료 식별을 위한 YOLO 모델 학습
모델 테스트	11월 2일 ~ 11월 24일	YOLO 버전 별 정확도 테스트
1차 MVP	11월 10일	Flutter 1차 패키징 및 테스트
최적화	11월 10일 ~	코드 개선 & 기능개선
총 개발 기간	5주	

앱 시연

앱 시연 영상



Made with Gamma

앱 시연



가지고 있는 조미료와 양념 선택하기

조미료

선택하신 조미료는 레시피 추천 과정에 반영됩니다.

- 소금
- 후추
- 설탕
- 고춧가루

계 +

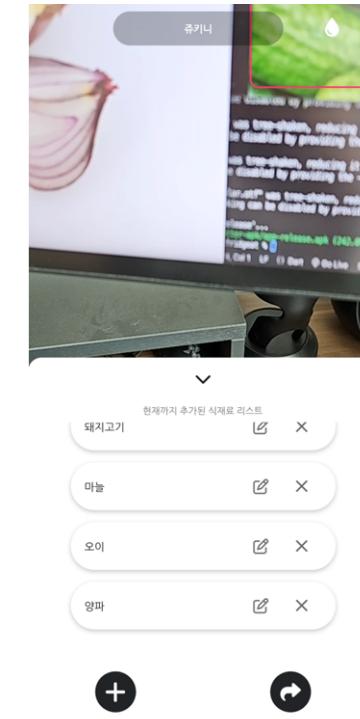
양념류

선택하신 양념은 레시피 추천 과정에 반영됩니다.

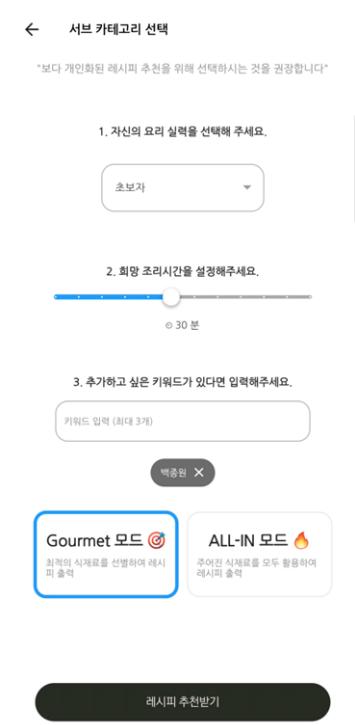
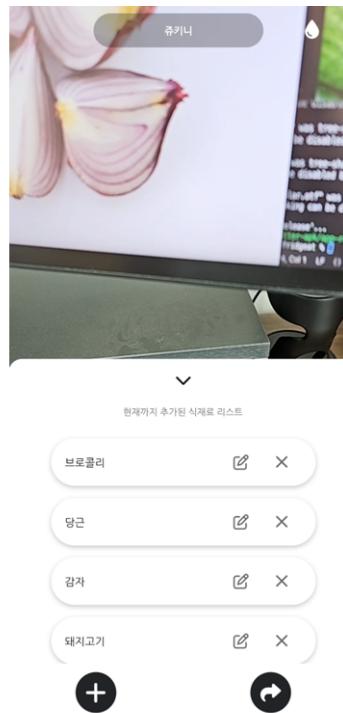
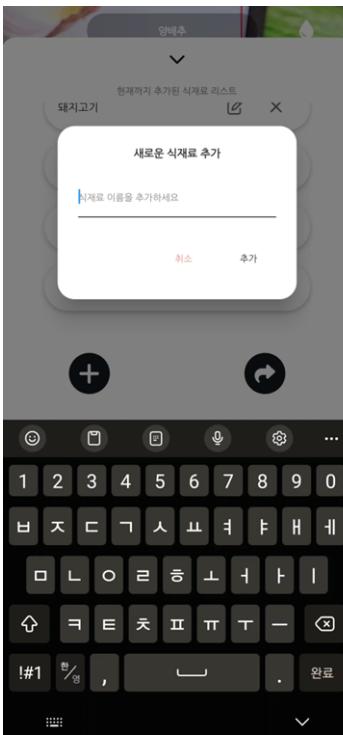
- 다진마늘
- 고추장
- 된장
- 케첩

마요네즈 +

다음



앱 시연



앱 시연



돼지고기 볶음밥

돼지고기와 야채를 활용한 맛있고 간편한 볶음밥입니다.

난이도 ★☆☆

소요시간 ⏳ 30분 이내

분량 1인분



기본재료

- 돼지고기 (삼겹살 또는 목살) 150g
- 브로콜리 1/4개
- 당근 1/4개
- 감자 1/4개
- 마늘 1쪽
- 양파 1/4개
- 밥 1공기

조리방법

Step 1. 돼지고기는 곱게 썰어서 소금과 후추로 간을 한 후, 팬에 기름을 두르고 익힌다.

Step 2. 익힌 돼지고기를 팬에서 꺼내고, 돼지고기를 제외한 야채들을 적당한 크기로 썬다.

Step 3. 돼지고기가 익은 팬에 다진 마늘을 넣고 볶다가 양파와 야채들을 넣고 함께 볶는다.

Step 4. 야채들이 익으면 밥을 넣고 고추장, 간장, 설탕, 소금, 후추로 간을 한다.

4. 프로젝트 수행 결과

YOLO

One-stage detection

- 객체 감지를 단일 단계로 처리
- 빠른 속도로 실시간 객체 감지와 같은 빠른 응용에 적합

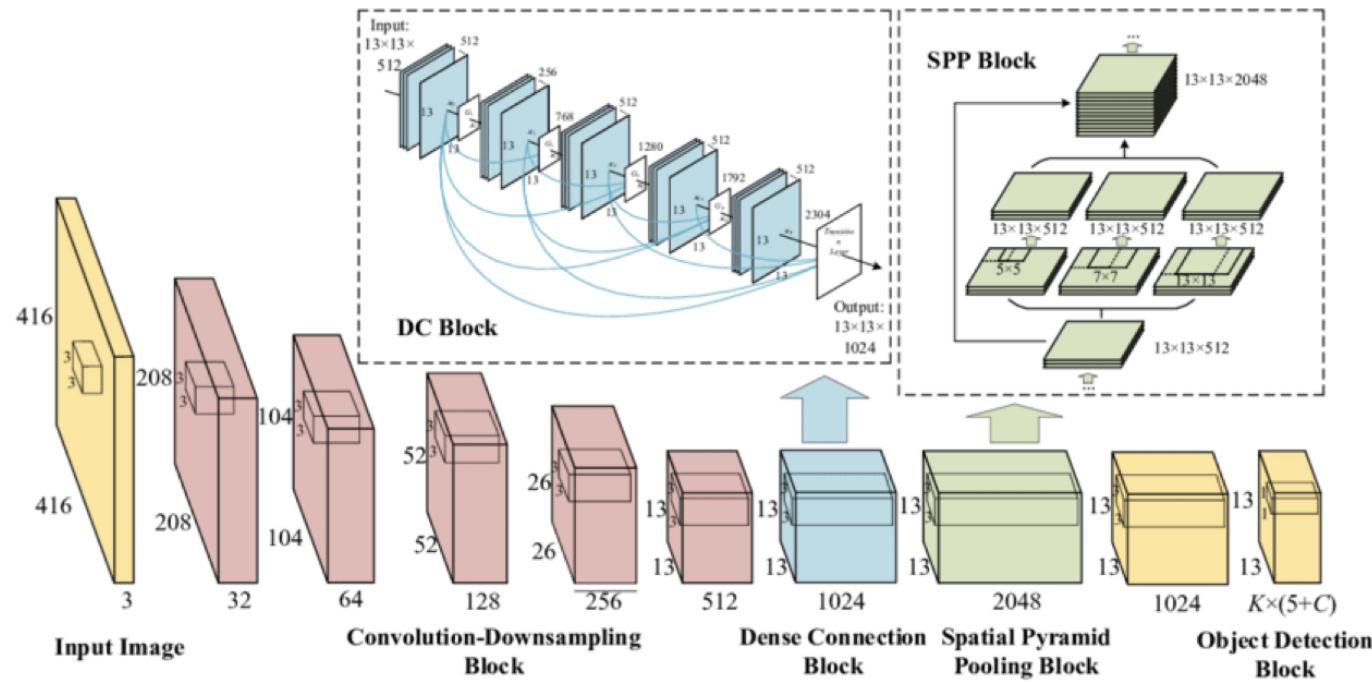
YOLO (You Only Look Once)

- 객체 감지 작업을 수행하는 딥 러닝 기반의 모델로, 이미지나 비디오에서 객체의 Bounding Box와 클래스를 예측하는 데 사용



베이스라인 모델 선정

- YOLO v5 모델은 **높은 실행 속도와 정확성을 결합하여 다양한 객체 감지 응용 분야에서 사용되며, 커뮤니티에서 활발하게 사용되고 개선되고 있다.**
- YOLOv5 모델은 사용이 편리하며 가장 많은 레퍼런스를 가지고 있다.



탐색적 분석 및 전처리

데이터 수집

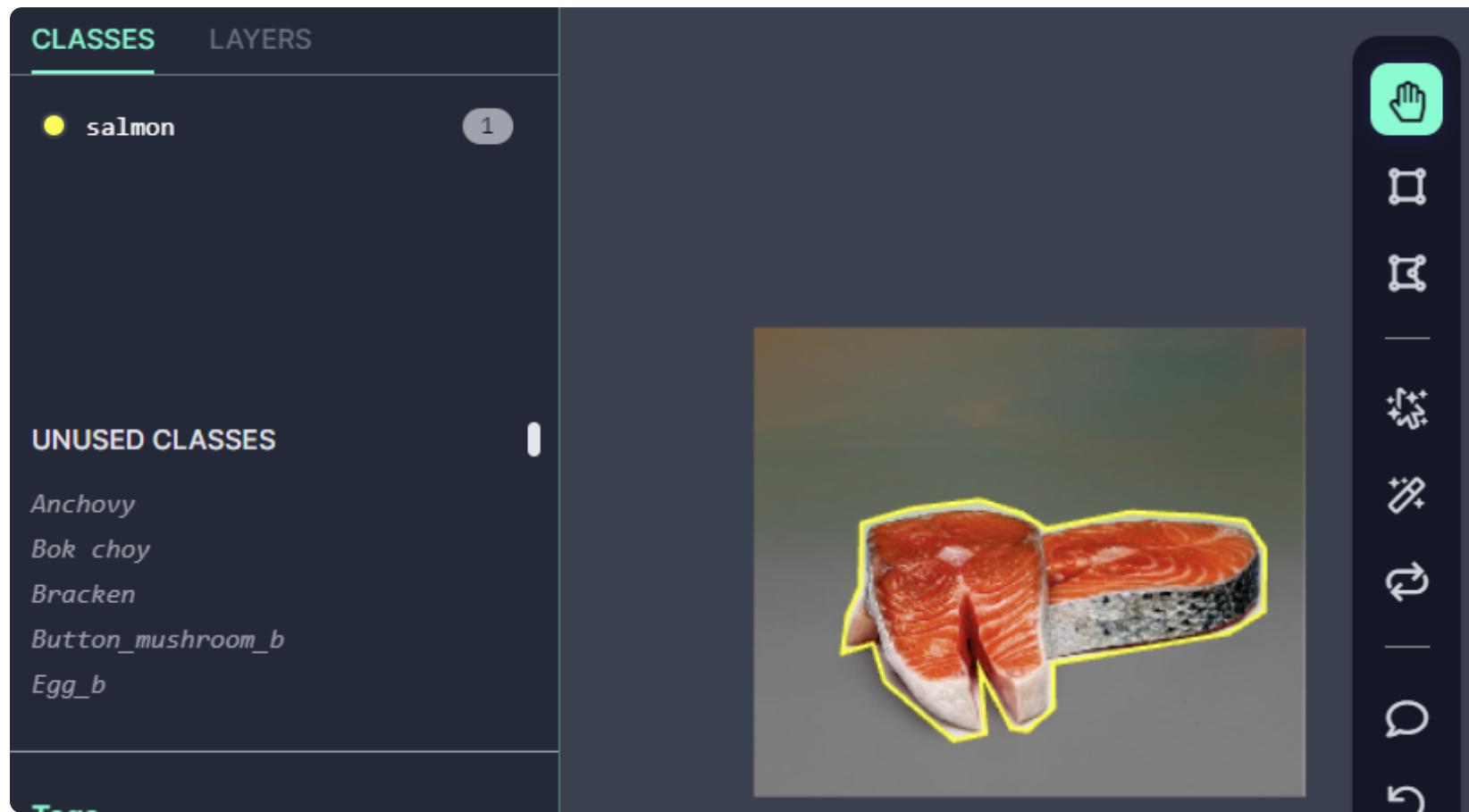
- 상업적 이용을 염두해 저작권 라이선스 확인 후 수집

→ 네이버 이미지 검색을 통해 상업적 이용이 가능한 라이선스 확인 후 크롤링

```
  * WEBSCRAPING.py
10  from bs4 import BeautifulSoup
11  from selenium import webdriver
12  from selenium.webdriver.common.keys import Keys
13  from selenium.webdriver.common.by import By
14
15  # 필요한 정보를 입력 받기
16  print("=" *80)
17  print("네이버에서 이미지를 검색하여 수집")
18  print("=" *80)
19
20  query_txt = input('크롤링할 이미지의 키워드 : ')
21  cnt = int(input('크롤링할 건 수 : '))
22
23  f_dir = "D:\\webcrawling"
24  if f_dir == '':
25      f_dir = "c:\\temp\\"
26
27  print("\n")
28
29  driver_path = "D:\\webcrawling\\chromedriver-win64\\chromedriver.exe"
30  options = webdriver.ChromeOptions()
31  options.add_argument("--timeout=YOUR_TIMEOUT_VALUE")
32  options.add_experimental_option("excludeSwitches", ["enable-logging"])
33  driver = webdriver.Chrome(service=ChromeService(ChromeDriverManager().inst
```

데이터 전처리

- roboflow를 활용 yolo모델에 활용 가능한 데이터 형식에 맞춰 라벨링
- 이미지 resize 640 x 640
- 이미지 Argumentation 적용 → vertical, horizontal, mixup, mosaic
- 40개의 class, class별 평균 이미지 약 100장
- Argumentation까지 적용한 Total Set : 8010장



베이스라인 (YOLOv5)

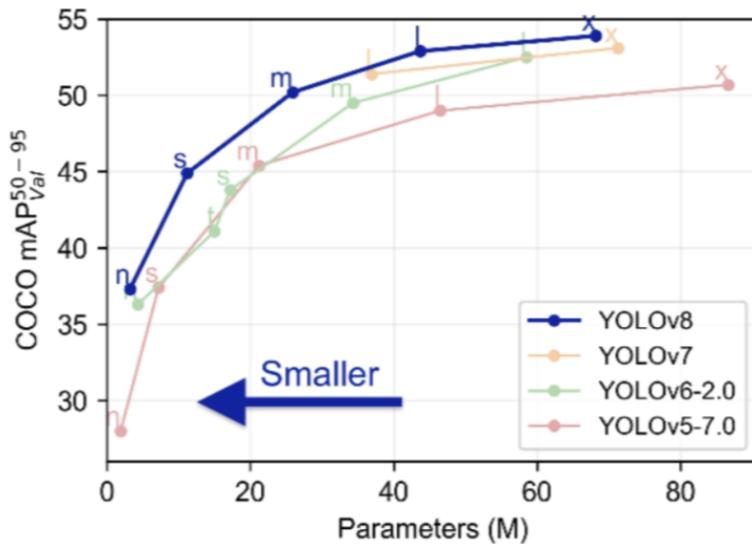
- 평균 MAP = 0.56, 일부 지나치게 낮은 mAP를 보이는 클래스들의 존재
- 초기 lr을 낮춰 훈련된 모델을 기반으로 하여, 새로운 데이터셋에 대해 모델을 미세조정
- 초기 학습층을 freeze해 사전 학습에서 얻은 지식을 유지한채 모델이 새로운 데이터에 대한 더 높은 수준의 추상적 특성을 학습하는 데 집중하도록 함

YOLOv5s summary: 157 layers, 7123399 parameters, 0 gradients, 16.1 GFLOPs						
Class	Images	Instances	P	R	mAP50	mAP50-95:
all	837	1080	0.597	0.548	0.56	0.37
Bok choy	837	46	0.661	0.635	0.69	0.477
King trumpet mushroom	837	15	0.492	0.333	0.401	0.274
Paprika	837	34	0.711	0.5	0.561	0.345
Sausage	837	36	0.712	0.833	0.76	0.574
bean sprouts	837	21	0.841	0.757	0.798	0.431
beef	837	30	0.611	0.567	0.643	0.439
broccoli	837	51	0.848	0.922	0.943	0.825
button mushroom	837	26	0.212	0.154	0.191	0.128
cabbage	837	24	0.56	0.542	0.648	0.466
carrot	837	24	0.726	0.417	0.578	0.362
chicken	837	22	0.948	0.818	0.885	0.697
chives	837	19	0.545	0.506	0.541	0.302
cucumber	837	24	0.385	0.417	0.331	0.202
egg	837	79	0.519	0.683	0.523	0.368
eggplant	837	24	0.538	0.34	0.409	0.229
enoki mushroom	837	20	0.659	0.85	0.67	0.384
garlic	837	30	0.267	0.167	0.158	0.12
ginger	837	15	0.736	0.2	0.386	0.254
green onion	837	22	0.417	0.273	0.364	0.227
jukini	837	26	0.472	0.5	0.399	0.32
lettuce	837	21	0.596	0.619	0.649	0.404
mackerel	837	15	0.732	0.867	0.823	0.582
mussel	837	21	0.912	0.857	0.89	0.655
napa cabbage	837	23	0.411	0.365	0.382	0.216
onion	837	39	0.467	0.538	0.426	0.282
oyster mushroom	837	23	0.635	0.652	0.639	0.405
perilla leaves	837	23	0.729	0.826	0.704	0.427
pork	837	22	0.563	0.545	0.622	0.35
potato	837	38	0.476	0.368	0.437	0.318
radish	837	21	0.472	0.286	0.384	0.201
salmon	837	27	0.902	0.778	0.832	0.604
shepherd-s purse	837	18	0.707	0.667	0.799	0.495
shiitake	837	33	0.375	0.242	0.236	0.155
shrimp	837	39	0.548	0.497	0.407	0.222
small octopus	837	19	0.461	0.421	0.424	0.273

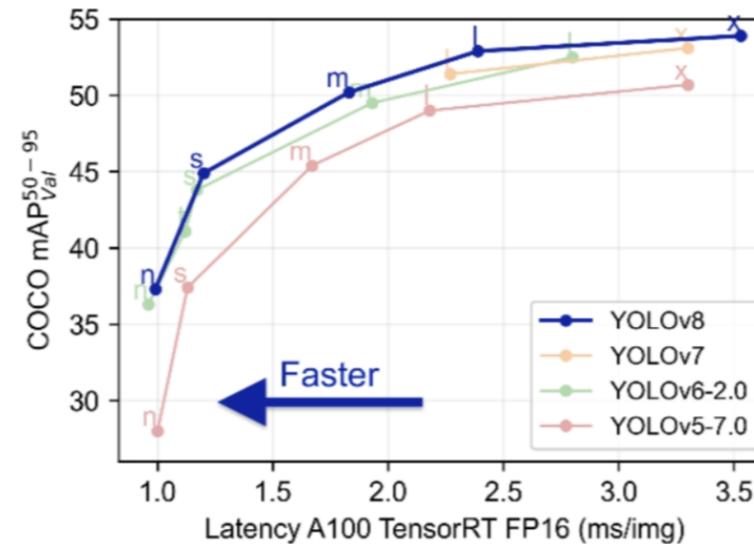
베이스라인 피드백

YOLO V5 & V8 비교

V5



V8

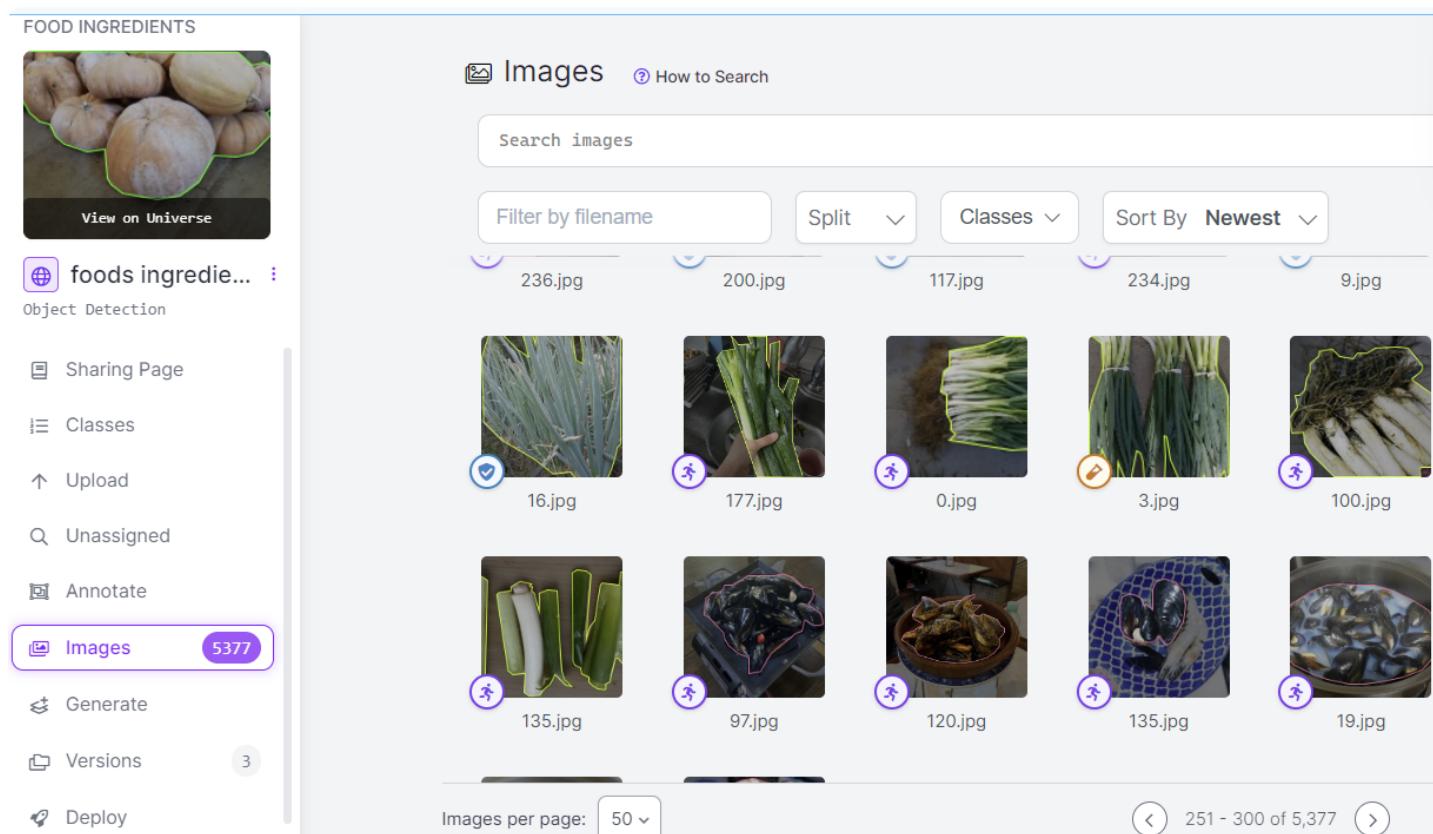


YOLO 모델 중 V8이 정확도와 fps의 성능이 가장 높음

탐색적 분석 및 전처리 (EDA)

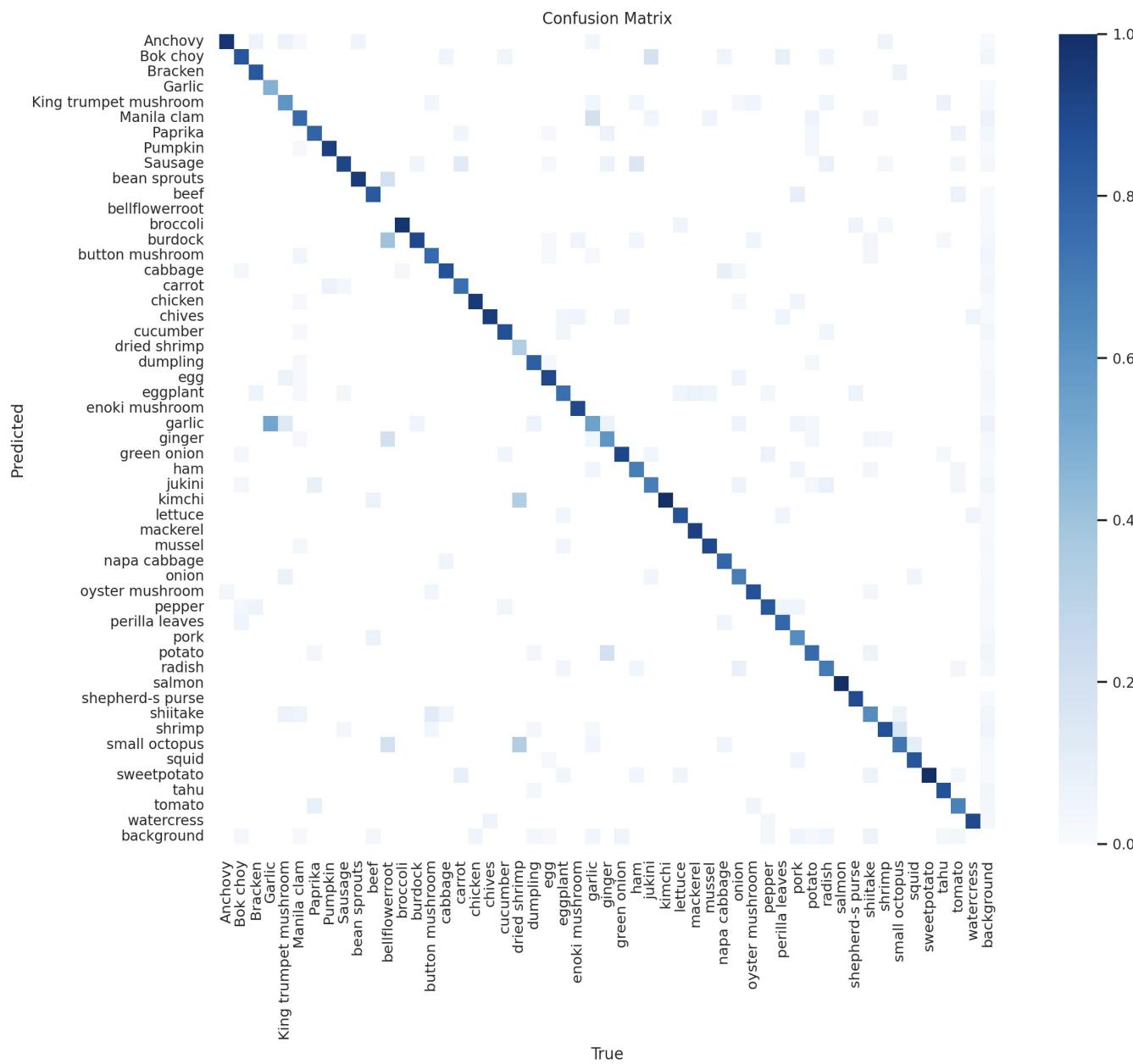
추가 데이터 수집

- Roboflow를 활용 YOLO모델에 활용 가능한 데이터 형식에 맞춰 Labeling
- 이미지 Resize 640 x 640 - 이미지 Augmentation 적용 → Vertical, Horizontal, Mixup, Mosaic
- 52개의 class, class별 평균 이미지 약 100장
- 성능이 낮았던 일부 class의 이미지 추가 수집
- Augmentation까지 적용한 Total Set : 13157장



YOLO v8s

평균 mAP **0.56 > 0.9**, mAP 50-95 **0.37 > 0.899** 가장 낮은 레이블의 mAP, 또한 Garlic (0.158>0.633)로 성능 향상



ML 모델 서빙

ML 모델 서빙 방법

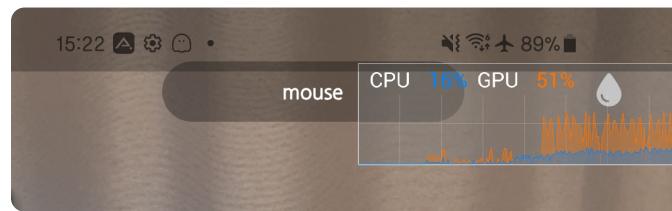
- 로컬(Local) 디바이스
- 원격(Remote) 서버

ML 모델 서빙 시 고려 사항

- 식재료 인식 정확도
- 레시피 추천 정확도
- 앱 사용 편의성
- 모델 보안성

로컬 모델 탑재시 문제점

- 발열 문제
- 높은 GPU사용량
- 높은 CPU사용량



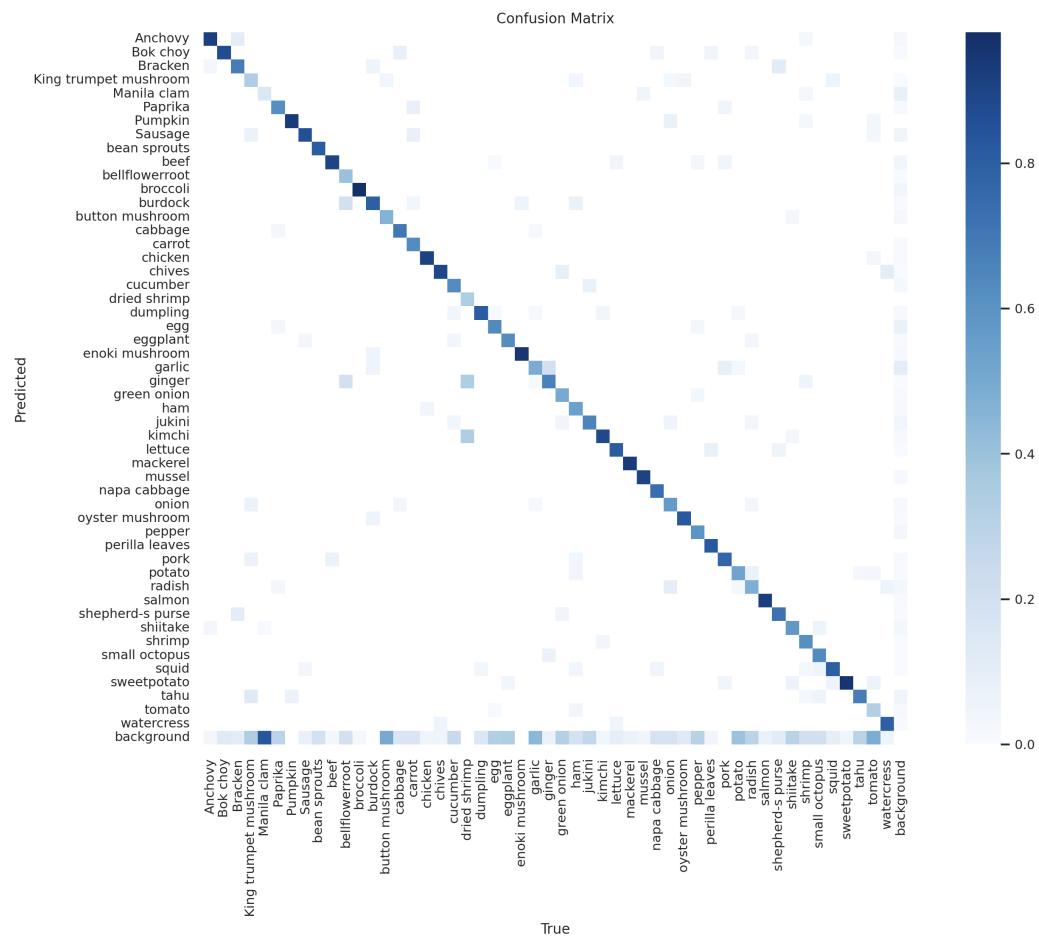
모델 버전 비교

V8s vs V8n

Model	mAP	Speed CPU	Speed GPU	Params	FLOPs
YOLOv8n	37.3	80.4	0.99	3.2	8.7
YOLOv8s	44.9	128.4	1.20	11.2	28.6
YOLOv8m	50.2	234.7	1.83	25.9	78.9
YOLOv8l	52.9	375.2	2.39	43.7	165.2
YOLOv8x	53.9	479.1	3.53	68.2	257.8

YOLOv8n

- 평균 mAP는 0.7, 가장 낮은 레이블 mAP는 0.26으로 **평균 20%정도의 성능 하락**
 - 로컬 탑재시 기존보다 퍼포먼스 향상
 - 모델의 추가적인 성능 향상을 위해 평균 이하의 mAP값을 가지는 클래스의 데이터셋을 분석중



5. 프로젝트 최종 예상 결과

아이펠톤 기간 내 결과

아이펠톤 기간 내로 학습된 52개의 식재료 클래스에 한해, 인식률은 실제 유저가 사용이 가능할 정도로 충분할 것으로 판단된다. 또, 사용자가 선택한 정보에 기반한 레시피 제공 단계까지의 주요 기능 구현도 가능할 것으로 보인다. 하지만 사람들에게 서비스를 제공하기 까지 개선해야 할 사항들이 있다.

이후 계획

- Cloud Firestore에 유저가 이전에 선택했던 정보를 저장함으로써 매번의 선택 과정을 줄이는 사용자 경험의 **최적화**
- GPT API를 통해 제공되는 레시피를 Cloud Firestore에 저장하여 자체적인 레시피 데이터베이스를 구축하고, 이후 유저의 요청에 활용될 수 있도록 **서비스 고도화**
- **수익화**(요청 횟수 제한, 광고)

6. 느낀 점

프로젝트 수행상 어려움 극복 사례

- 자체 레시피 데이터베이스 활용의 제한 → **GPT API 활용**
- 모델 서빙에 대한 고민 → **초기 기획에 기반, 로컬 운용 방식 결정**
- 모델 로컬 운용을 위한 경량화 모델 필요 → **A/B 테스트 결과에 근거하여 YOLOv8 nano로 최종 결정**

프로젝트 수행 중 잘한 점

- **비즈니스 마인드**로 프로젝트 진행
- **격려 및 긍정적인 분위기** 속 팀 내 회의 및 의사소통
- 공공 데이터셋에 없는 **자체 데이터셋** 구축 (한식 식재료 데이터셋 구성)

프로젝트 개선점

- 다양한 식재료 인식 및 일부 **식재료의 인식률 저하** 문제
- 모델 로컬 운용에 따른 **패키지 / 모델 크기 / 정확도 / 유지보수** 문제
- 추천된 레시피의 **상세 정보 제공 부족** 문제
- 레시피 생성 과정 **실시간 표현** 문제
- 지속적인 ChatGPT API 사용량 증가에 따른 비용 및 **서비스 종속성** 문제



감사합니다.