

개인 맞춤형 식품 알레르기 관리 시스템



21012047 인공지능학과 강민성

제출일 : 2025년 12월 13일

목차

1. 서론
 - 1.1. 프로젝트 배경 및 필요성
 - 1.2. 프로젝트 목표 및 범위
2. 시스템 설계 및 기술 스택
 - 2.1. 전체 시스템 아키텍처
 - 2.2. 사용 기술 및 개발 환경
3. 데이터 구축 및 전처리
 - 3.1. 웹 크롤링 및 데이터 수집
 - 3.2. 이미지 데이터 최적화
 - 3.3. 데이터베이스 설계(ERD)
4. AI 모델 개발 및 적용
 - 4.1. 모델 선정 및 데이터셋
 - 4.2. 학습 과정 및 하이퍼파라미터
 - 4.3. 성능 평가
5. 주요 기능 구현
 - 5.1. 사용자 맞춤형 필터링
 - 5.2. 이미지 검색 기능
 - 5.3. 성분표 분석 및 자동 입력
 - 5.4. 관리자 기능
 - 5.5. 사용자 피드백 기반 데이터 수집 기능
6. 문제 해결 및 성과
 - 6.1. 기술적 이슈 해결
 - 6.2. 최종 결과물
7. Docker 기반 배포 및 운영 환경
 - 7.1. Docker 기반 배포 환경 설계 및 이미지 구성
 - 7.2. 동적 파일 관리 및 데이터베이스 연동
 - 7.3. Docker 실행 결과 및 배포 검증
8. 결론 및 향후 과제
 - 8.1. 프로젝트 요약
 - 8.2. 한계점 및 향후 개선 방안

1. 서론

1.1. 프로젝트 배경 및 필요성

현대 사회에서 식품 알레르기는 단순한 개인의 기호 문제가 아니라 생명과 직결되는 중요한 건강, 안전 문제로 인식되고 있다. 가공식품의 소비가 증가함에 따라 식품에 포함된 알레르기 유발 성분을 정확히 확인하는 것이 필수적이 되었지만, 소비자가 제품 포장지의 깨알 같은 성분표를 직접 확인하거나 여러 웹사이트를 번갈아 검색하는 과정은 번거로울 뿐만 아니라, 작은 실수도 심각한 건강 문제로 이어질 수 있다는 위험이 존재한다.

본 프로젝트의 기반은 1학기 데이터베이스 기말 프로젝트로 수행한 '웹 크롤링 기반 식품 알레르기 정보 관리 시스템'이다. 해당 프로젝트는 오투기물을 대상으로 웹 크롤링을 수행하여 식품, 성분, 알레르기 정보를 수집하고, 이를 MySQL 기반 관계형 데이터베이스로 구축하였다. 또한 Python 콘솔 프로그램을 통해 제품 검색, 개인 알레르기 등록, 교차 반응군 확인, 관리자 통계 기능 등을 구현함으로써 분산된 알레르기 정보를 하나의 시스템에서 통합 관리할 수 있는 기반을 마련하였다.

그러나 기존 DB 프로젝트는 텍스트 기반 검색과 콘솔 환경 중심의 서비스 구조라는 한계를 가지고 있었다. 사용자는 정확한 제품명을 알고 있어야만 검색이 가능했으며, 실제 사용 환경에서 중요한 '이미지를 기반으로 한 직관적인 검색' 기능은 제공하지 못했다. 또한 성분표 해석 역시 사람이 직접 확인해야 하는 부분이 많아 자동화 수준에서도 한계가 존재하였다.

최근 딥러닝 기반 이미지 인식 기술과 대규모 언어 모델(LLM)의 비약적인 발전은 이러한 한계를 극복할 새로운 가능성을 제시한다. 이에 본 프로젝트에서는 기존의 탄탄한 데이터베이스 인프라 위에 YOLO v8 기반의 객체 탐지 기술과 Gemini 2.5 Flash API를 활용한 지능형 성분 분석 기능을 결합하고자 한다. 이는 단순한 데이터 관리를 넘어, 사용자가 사진을 찍는 직관적인 행위만으로 위험 정보를 제공받을 수 있는 '사용자 맞춤형 AI 서비스'로의 확장을 의미한다.

1.2. 프로젝트 목표 및 범위

본 프로젝트의 궁극적인 목표는 기존에 구축한 식품 알레르기 데이터베이스 인프라를 기반으로, 딥러닝과 대규모 언어 모델(LLM) 기술을 결합하여 사용자가 보다 직관적이고 정확하게 알레르기 정보를 확인할 수 있는 지능형 서비스를 구현하는 것이다. 기존의 텍스트 중심 검색 방식을 넘어, 이미지를 통한 자동 음식 인식과 성분표 자동 분석 기능을 도입함으로써 사용자 편의성을 극대화하고, 데이터베이스에 축적된 개인 알레르기 정보를 바탕으로 실시간 위험 판단이 가능한 통합 시스템 구축을 지향한다.

또한 본 프로젝트는 단순한 정보 제공 서비스에 그치지 않고, 사용자의 상태를 지속적으로 반영하여 YOLO 기반 이미지 인식 결과와 LLM 기반 성분 분석 결과를 종합적으로 판단하는 개인화 판단 구조를 서비스에 적용하는 것을 하나의 목표로 설정하였다.

세부적인 목표는 다음과 같다.

- 데이터셋 구축 및 모델 학습 : AIHub의 '건강관리를 위한 음식 이미지' 데이터셋을 활용하여 YOLO v8 Small 모델을 전이 학습하고, 571개 음식 클래스에 대한 강건한 인식 모델을 구축한다.
- 이미지 기반 자동 검색 : YOLO 모델을 서비스와 연동하여, 사용자가 업로드한 음식 이미지를 분석하고 식품명을 자동으로 인식하는 이미지 검색 기능을 구현한다.
- LLM 기반 성분 분석 자동화 : Gemini 2.5 Flash API를 활용하여 비정형 데이터인 성분표 이미지나 텍스트를 분석하고, 알레르기 유발 물질을 자동으로 추출하여 데이터베이스와 연동한다.
- 개인 맞춤형 위험 판단 : 사용자의 알레르기 정보와 구축된 DB(교차 반응군, 대체 식품 정보)를 실시간으로 대조하여, 검색된 식품의 위험 여부를 '위험/안심' 뱃지로 시각화하여 제공한다.
- 웹 서비스 최적화 및 통합 : FastAPI 백엔드와 MySQL 데이터베이스, Tailwind CSS 기반 프론트엔드를 유기적으로 통합하고, 이미지 로컬 최적화(WebP)를 통해 실제 서비스 수준의 성능을 확보한다.

본 프로젝트에서 구현한 기능의 범위는 다음과 같다.

1) 인공지능 및 데이터 파이프라인

- YOLO v8 Small 모델링

- AIHub '건강관리를 위한 음식 이미지' 데이터셋(571개 클래스m 약 300만 장) 전처리 및 Train/Validation 데이터 분할
- YOLO v8 Small 모델 전이학습 적용 및 하이퍼파라미터 튜닝
- 사용자 업로드 이미지에 대한 실시간 추론 수행 및 Confidence Score 기반 1차 필터링 적용

- LLM 기반 지능형 분석 시스템

- Google Gemini 2.5 Flash API 연동 및 프롬프트 엔지니어링 적용
- 비정형 성분표 이미지, 텍스트를 정형화된 JSON 데이터로 변환하는 파싱 로직 구현
- YOLO 기반 사물 인식 실패 또는 신뢰도 저하 시 Gemini 기반 맥락 인식으로 보완하는 하이브리드 인식 알고리즘 적용

2) MLOps & Continuous Learning

- 사용자 피드백 루프 구축

- AI 이미지 검색 결과가 부정확한 경우, 사용자가 직접 정답을 입력할 수 있는 피드백 인터페이스 구현
- 사용자 상호작용을 통한 모델 오류 샘플 수집 구조 설계

- 학습 데이터 자동 수집 파이프라인

- 사용자가 피드백한 이미지를 static/dataset/images 디렉토리로 자동 분류 및 저장
- 정답 라벨, 타임스탬프, 사용자 입력 정보를 feedback_log.csv에 자동 기록
- 수집된 데이터를 Hard Negative Sample 기반 재학습 데이터셋으로 활용 가능하도록 설계하여, 향후 모델 재학습 및 성능 고도화를 위한 MLOps 기초 환경 구현

3) 백엔드 및 보안

- FastAPI 기반 RESTful API 설계

- 비동기(Async) 처리를 통한 고성능 API 서버 구축
- Pydantic 기반 엄격한 데이터 유효성 검사 및 API 스키마 정의

- 보안 인증 시스템

- OAuth2 표준 및 JWT(Json Web Token) 기반 안전한 로그인, 인증 체계 구현
- bcrypt 알고리즘을 이용한 비밀번호 단방향 암호화 저장
- 사용자와 관리자 권한 분리 및 접근 제어 로직 적용

4) 데이터베이스 및 시스템 최적화

- 데이터베이스 설계

- MySQL 기반 관계형 DB 설계
- 다대다(M:N) 관계 해소를 위한 교차 테이블 설계 및 Cascade 제약조건 적용

- 이미지 최적화

- 외부 이미지 핫링크 문제 해결을 위한 로컬 이미지 다운로드 자동화
- Pillow 라이브러리를 활용한 WebP 포맷 변환 및 압축(약 80% 품질) 처리로 저장 공간 및 트래픽 최적화

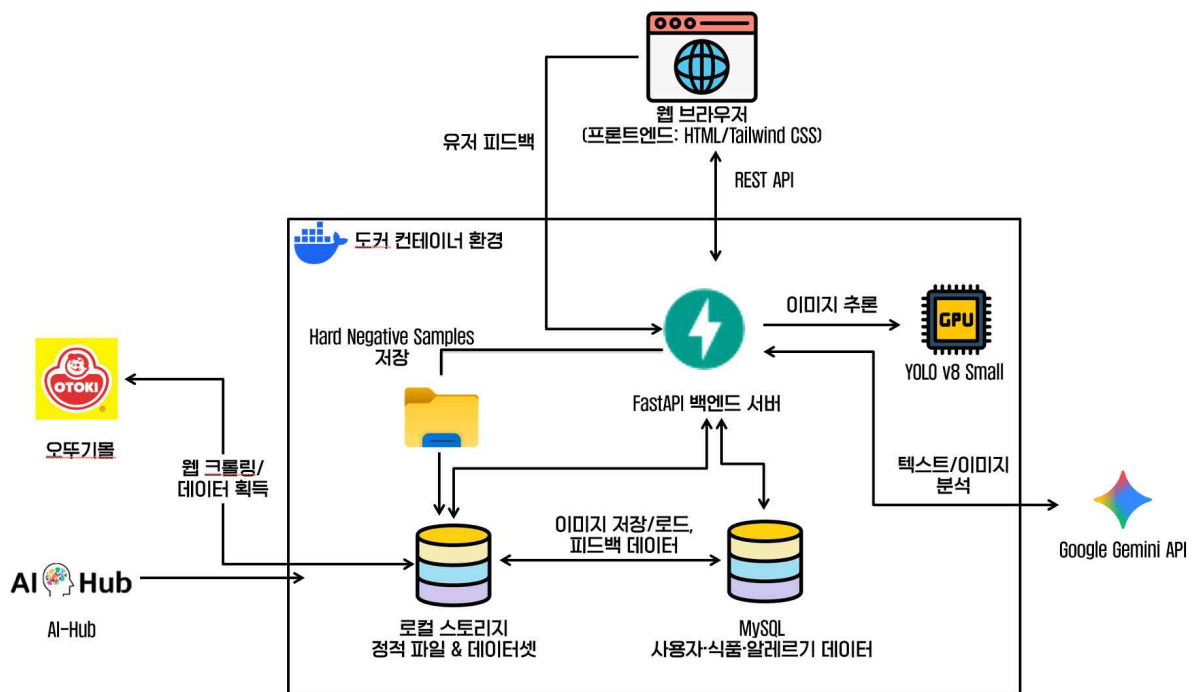
5) 프론트엔드 및 핵심 로직

- 개인 맞춤형 필터링 알고리즘
 - 사용자 알레르기 정보와 제품 성분 데이터를 실시간 매칭하는 개인화 필터링 로직 구현
 - DB 내 교차 반응군 데이터를 활용한 잠재적 위험 자동 탐지 및 경고 시스템 구축
- 반응형 인터페이스 및 관리자 기능
 - HTML5 및 Tailwind CSS 기반 모바일, 데스크탑 반응형 UI 구현
 - 관리자 대시보드를 통한 사용자 통계 시각화 및 식품 성분 데이터 CRUD 기능

2. 시스템 설계 및 기술 스택

2.1. 전체 시스템 아키텍처

본 프로젝트는 딥러닝 기반 이미지 인식(YOLO), LLM 기반 성분 분석(Gemini 2.5 Flash), FastAPI 백엔드, MySQL 데이터베이스, 그리고 프론트엔드 UI가 유기적으로 연동된 End-to-End AI 웹 서비스 구조로 설계되었다. 또한 단순한 기능 구현을 넘어, 사용자 피드백을 기반으로 데이터를 축적하고 모델을 재학습하는 MLOps 기반의 데이터 선순환 구조를 구축하였으며, Docker 컨테이너 기술을 도입하여 배포 환경의 일관성과 확장성을 확보하였다.

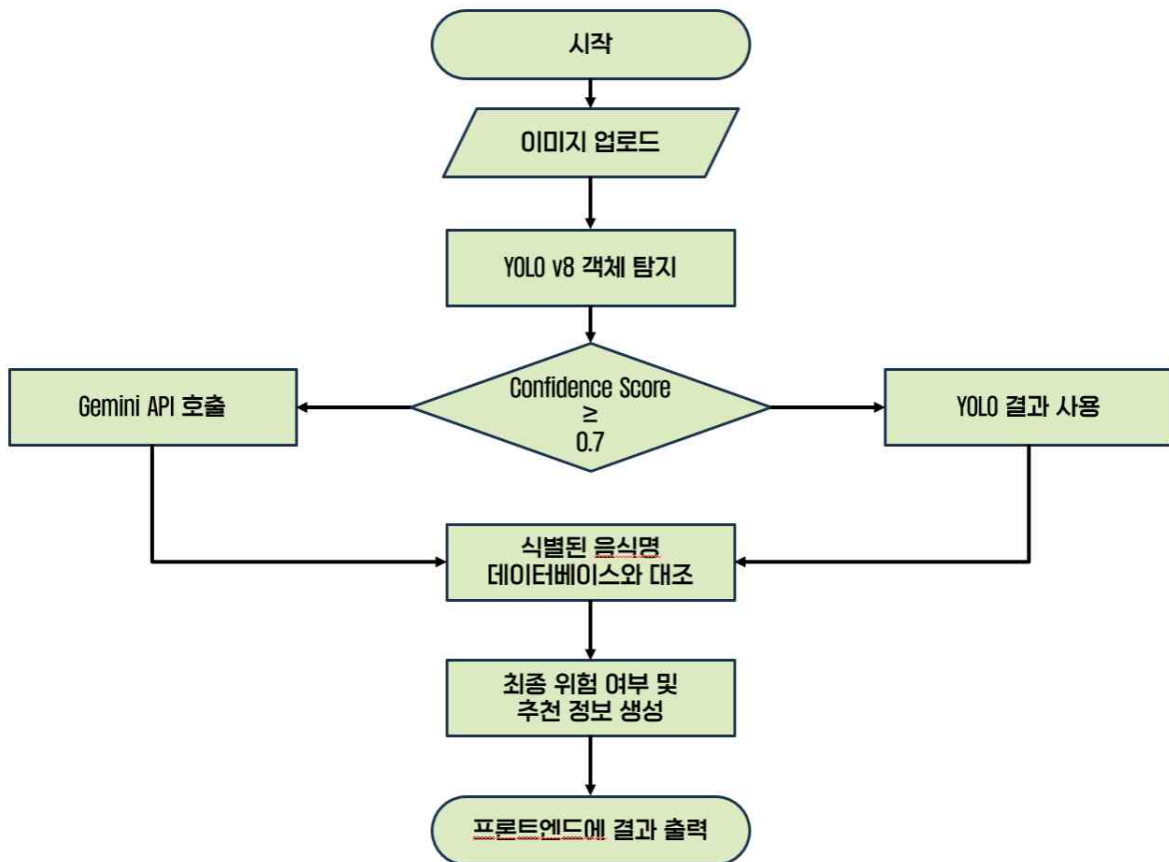


시스템 데이터 처리 흐름은 다음과 같다.

- 사용자 요청: 사용자는 웹 프론트엔드를 통해 음식 이미지를 업로드하거나 텍스트 검색을 요청한다.
- 보안 및 인증: 모든 요청은 FastAPI 백엔드 서버로 전달되며, JWT 인증을 통해 인가된 사용자만 접근하도록 제어된다.
- AI 인지: 업로드된 이미지는 서버에 임시 저장된 후, YOLO v8 Small 모델을 통해 1차적으로 음식 객체 탐지 및 분류가 수행된다.
- AI 추론: YOLO 추론 결과의 confidence score가 기준값(0.7) 미만이거나 성분표 분석이 필요한 경우, Google Gemini 2.5 Flash API를 호출하여 보완 추론 및 텍스트 분석

을 수행한다.

- 위험 판단: 식별된 음식명 및 성분 정보는 MySQL DB에 저장된 식품 정보, 사용자 개인 알레르기 정보, 교차 반응군 데이터와 실시간으로 대조되어 위험 여부가 판별된다.
- 결과 시각화: 최종 판단 결과는 '위험/안심' 배지, 대체 식품 추천, 주의 사항 등의 JSON 데이터로 프론트엔드에 전달되어 시각화된다.
- 데이터 선순환: 사용자가 AI의 인식 결과가 부정확하다고 판단하여 정답을 수정하면 해당 데이터는 '피드백 로그'로 저장된다.
- 모델 고도화: 수집된 피드백 데이터는 향후 YOLO 모델의 재학습을 위한 Hard Negative Sample로 분류되어, 지속적인 모델 성능 향상을 위한 핵심 데이터로 활용된다.



AI 하이브리드 추천 로직 순서도

2.2. 사용 기술 및 개발 환경

본 프로젝트는 딥러닝, 웹서비스, 데이터베이스, 보안, 프론트엔드, 배포 기술이 융합된 복합 기술 스택 기반 통합 AI 시스템으로 구현되었다. 사용된 주요 기술 및 개발 환경은 다음과 같다.

1) 인공지능 및 데이터 처리 기술

- Vision 모델: YOLO v8 Small (Ultralytics)
- 학습 방식: 전이학습
- 데이터셋: AIHub '건강관리를 위한 음식 이미지' (571개 클래스, 총 300만장 중 클래스별 300장씩 약 17만 장 사용)
- 성능 평가 지표: Precision, Recall, mAP50, mAP50-95
- 대규모 언어 모델(LLM): Google Gemini 2.5 Flash API
- LLM 역할: 성분포 이미지·텍스트 자연어 분석, JSON 구조화, YOLO 보완

2) 백엔드(Server & API)

- 프레임워크: FastAPI
- API 설계 방식: RESTful 아키텍처
- 비동기 처리: Async 기반 고성능 요청 처리
- 데이터 검증: Pydantic 기반 스키마 검증
- 서버 실행 환경: Uvicorn

3) 데이터베이스

- DBMS: MySQL
- 정규화 수준: 제 3정규화 적용
- 테이블 구성: Company, Food, Allergy, Alternative_Food, Cross_Reaction, Users 등 10개의 테이블로 구성된 스키마 설계 및 구축
- 관계 처리: 다대다(M:N) 관계 해소를 위한 Mapping Table 설계 및 Cascade 제약조건 적용

4) 보안

- 인증 방식: OAuth2 + JWT(Json Web Token)
- 비밀번호 보안: bcrypt 기반 단방향 해시 저장
- 권한 관리: 사용자 / 관리자 역할 기반 접근 제어

5) 프론트엔드

- UI 기술: HTML5, Tailwind CSS
- 스크립트 처리: JavaScript

- 데이터 통신: Fetch API 기반 비동기 통신
- UX 구성 요소: 모달, 토스트 알림, 스켈레톤 로딩, 반응형 레이아웃

6) 이미지 처리 및 최적화

- 이미지 처리 라이브러리: Pillow, OpenCV
- 포맷 변환: JPG → WebP 변환
- 압축 정책 : 약 80% 품질 설정을 통한 용량 및 트래픽 최적화
- 저장 방식: 서버 로컬 저장 구조

7) 개발, 운영 및 배포 환경

- 개발 언어: Python 3.10.11
- 개발 운영체제: Windows 11
- GPU 환경: NVidia RTX 4070
- 형상 관리: Github
- 컨테이너 기반 배포(Docker)
 - Docker를 활용한 FastAPI 서버, MySQL DB, AI 추론 환경 컨테이너화
 - 개발 환경과 운영 환경 간 의존성 불일치 문제 해결 및 재현성 확보
 - Dockerfile을 통한 Python, CUDA, 라이브러리 환경 일괄 구성
 - 컨테이너 기반 구조를 통해 향후 클라우드 환경으로의 확장 가능성 확보

3. 데이터 구축 및 전처리

3.1. 웹 크롤링 및 데이터 수집

본 프로젝트의 식품·성분 데이터는 '오투기물' 웹사이트를 대상으로 수행한 웹 크롤링을 통해 수집하였다. 크롤링은 Python 환경에서 Requests 및 BeautifulSoup 라이브러리를 활용하여 자동화하였으며, 수집 대상은 실제 웹 서비스 운영에 활용 가능한 실데이터를 기준으로 설정하였다.

데이터 수집은 정형 데이터 크롤링과 비정형 데이터 분석의 두 단계로 진행되었다.

- 기본 정보 수집: Python BeautifulSoup 라이브러리를 활용하여 제품명, 제조사, 상세페이지 URL과 같은 메타데이터를 수집하였다.
- 분석 정보 추출: 대부분의 식품 성분 정보가 텍스트가 아닌 '이미지' 형태로 제공되는 문제를 해결하기 위해, 상세 페이지의 제품 설명 이미지를 다운로드한 후 OCR 기술을 적용하였다. 이를 통해 이미지 속에 포함된 '원재료명'과 '알레르기 유발 물질' 텍스트를 추출하여 디지털 데이터로 변환 및 저장하였다.

초기 설계에서는 크롤링한 원본 이미지 URL을 서비스에 직접 호출하는 방식을 고려하였다. 하지만 이러한 방식은 외부 서버의 보안 정책에 의해 언제든지 접근이 차단될 수 있어, 서비스의 안정성이 외부 서버 환경에 종속되는 문제가 존재한다. 그리고 사용자가 페이지를 로딩할 때마다 외부 서버와 직접 통신해야 하므로, 로딩 속도 저하 및 불필요한 네트워크 트래픽 증가가 발생할 수 있다. 마지막으로 외부 서버의 도메인 변경, 리소스 삭제, 접근 정책 변경이 발생할 경우, 서비스 전체의 이미지 출력이 동시에 실패할 수 있는 리스크가 존재한다.

이를 해결하기 위해 크롤링 단계에서 User-Agent 헤더를 조작하여 보안을 우회하고, 모든 제품 이미지를 서버의 로컬 디렉토리로 다운로드하여 데이터의 안정성을 확보하였다.

3.2. 이미지 데이터 최적화

웹 서비스 환경에서 이미지 데이터는 사용자 경험(UX)과 서버 자원 효율성에 직접적인 영향을 미친다. 이에 본 프로젝트는 Pillow 라이브러리를 활용한 이미지 포맷 변환 및 압축 기반의 이미지 최적화 파이프라인을 구축하였다.

이미지 최적화 프로세스는 다음과 같다.

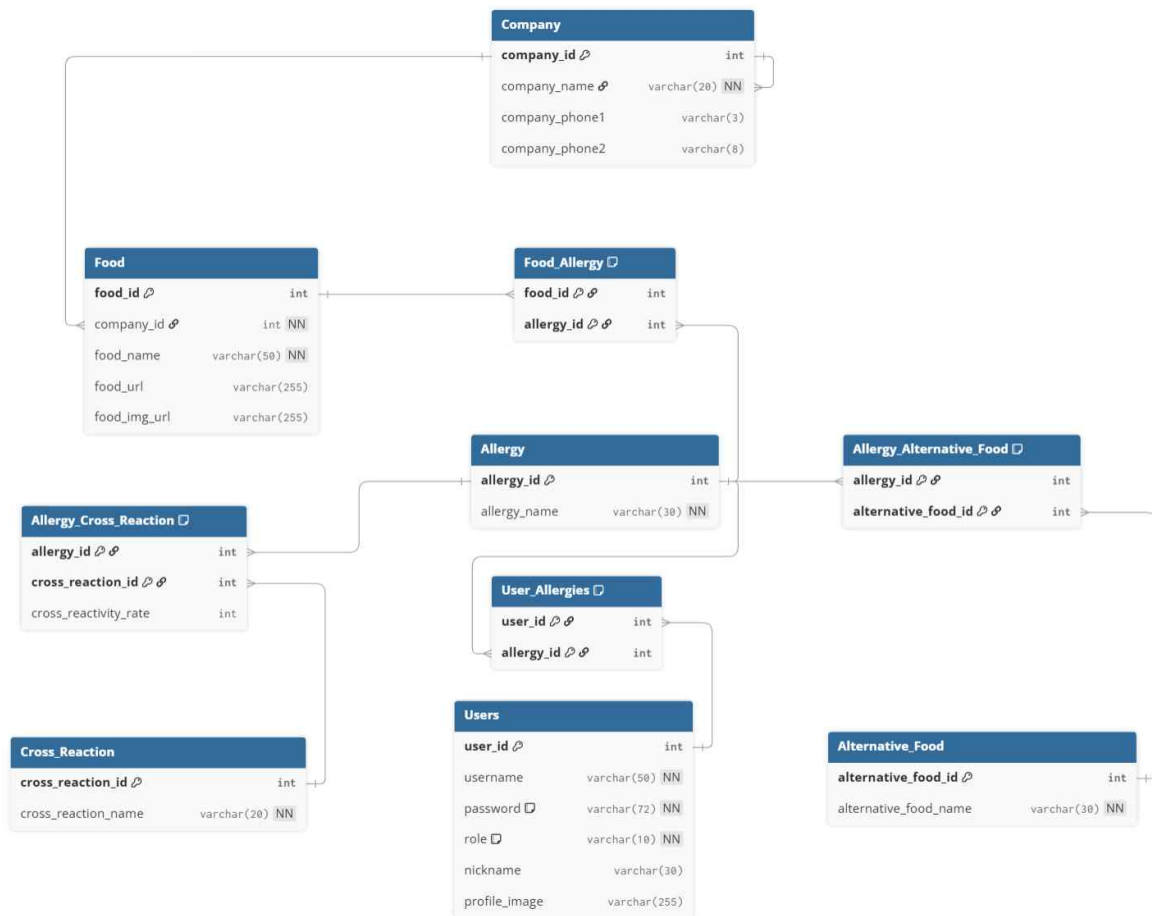
- 수집된 JPG/PNG 이미지를 구글이 개발한 차세대 웹 이미지 포맷인 WebP로 일괄 변환하였다.
- 인간의 눈으로 구분이 어려운 수준인 품질(80%)로 손실 압축을 적용하여, 시각적 품질은 유지하면서 파일 용량을 원본 대비 약 30~50% 절감하였다.

- 변환된 이미지는 FastAPI의 StaticFiles 설정을 통해 별도의 미디어 서버 없이도 고속으로 서빙 되도록 구성하였다.

이 전략은 웹 크롤링 데이터뿐만 아니라 사용자가 업로드하는 프로필/음식 이미지에도 동일하게 적용되어 서비스 전반의 성능 일관성을 유지하였다.

3.3. 데이터베이스 설계(ERD)

본 프로젝트의 데이터베이스는 단순한 정보 저장을 넘어, 사용자 정보, 식품 데이터, AI 분석 결과, 관리자 통계를 유기적으로 연결하는 통합 저장소로 설계되었다. MySQL을 사용하였으며, 데이터 중복 최소화와 무결성 보장을 위해 제3정규화(3NF) 원칙을 준수하여 구축하였다.



본 프로젝트의 데이터베이스는 총 10개의 핵심 테이블로 구성된다.

- 핵심 정보: Users(계정/인증), Food(식품/이미지 경로), Allergy(알레르기 성분), Company(제조사)
- 관계 매핑: User_Allergies, Food_Allergy, Allergy_Cross_Reaction, Allergy_Alternative_Food

- 알레르기 관련 정보: Cross_Reaction(교차 반응군), Alternative_Food(대체 식품)

데이터베이스 설계 과정에서 식품(Food)과 제조사(Company) 정보를 분리하여 이행적 함수 종속을 제거하고, 제조사 정보 변경 시 발생할 수 있는 갱신 이상(Anomaly)을 방지하였다.

또한 하나의 식품이 여러 알레르기 성분을 포함하거나, 하나의 알레르기가 여러 대체 식품을 갖는 복잡한 다대다 관계를 4개의 매핑 테이블로 구조화하여 데이터의 원자성을 보장하였다.

마지막으로 모든 관계 테이블에 ON DELETE/UPDATE CASCADE 제약조건을 설정하여, 회원 탈퇴나 제품 삭제 시 연관된 하위 데이터가 자동으로 정리되도록 구현하여 데이터의 고립을 원천적으로 방지하였다.

4. AI 모델 개발 및 적용

본 프로젝트는 사용자 편의성을 극대화하기 위해 '이미지 기반 검색'과 '지능형 성분 분석'이라는 두 가지 핵심 기능을 AI 기술로 구현하였다. 이를 위해 객체 탐지 모델(YOLO v8)을 학습시키고, 대규모 언어 모델(Gemini 2.5 Flash API)을 연동하여 하이브리드 추론 시스템을 구축하였다.

4.1. 모델 선정 및 데이터셋

4.1.1. 객체 탐지 모델 선정

본 프로젝트의 핵심 기능인 '이미지 기반 음식 인식'은 사용자가 업로드한 이미지를 대상으로 객체 탐지 문제로 정의할 수 있다. 단순한 이미지 분류와 달리, 본 서비스는 음식이 접시 위에 복수로 존재하거나 배경 요소가 함께 등장하는 환경에서도 주요 음식 객체의 위치와 클래스를 동시에 정확히 추론해야 하는 특성을 가진다. 이에 따라 단일 객체 분류 모델이 아닌 Bounding Box 기반 객체 탐지 모델을 채택하였다.

최근 객체 탐지 분야에서는 Faster R-CNN, RetinaNet, YOLO, EfficientNet, DETR 계열 등 다양한 딥러닝 모델이 제안되고 있다. 이들 중 Faster R-CNN은 높은 정확도를 보이지만 2-stage 구조로 인해 추론 속도가 느리고, DETR 계열은 Transformer 기반 구조로 연산량이 크고 대규모 학습 자원이 요구되는 한계가 있다. 반면 YOLO 계열은 one-stage 탐지 구조를 기반으로 속도와 정확도의 균형이 뛰어난 모델로 알려져 있다.

본 프로젝트에서는 이러한 특성과 실시간 웹 서비스 환경에서의 응답 속도, 구현 난이도, 학습 안정성 등을 종합적으로 고려하여 YOLO v8 Small 모델을 최종 객체 탐지 모델로 선정하였다.

4.1.2. 데이터셋 구축

한국인의 실제 식생활 환경을 반영한 실용적인 모델 구축을 위해 AI-Hub의 '건강관리를 위한 음식 이미지' 데이터셋을 활용하였다. 해당 데이터셋은 다양한 조명, 촬영 각도, 배경 환경에서 수집된 고해상도 음식 이미지를 포함하고 있어 실생활 환경에서의 강건성 확보에 적합하다. 이번 프로젝트에서는 571개 전체 클래스를 모두 유지한 상태에서 학습을 수행하였으며, 특정 클래스만 선별하여 사용하는 방식이 아닌 전체 클래스 기반의 대규모 다중 클래스 객체 탐지 문제를 그대로 반영하였다.

AI-Hub의 원본 데이터셋은 클래스당 수천 장 수준의 이미지가 포함되어 있어, 전체 데이터(약 300만 장)를 그대로 사용하는 경우 개인 개발 환경(GPU 메모리, 저장공간, 학습 시간)의 물리적 제약으로 인해 학습이 현실적으로 불가능한 문제가 존재하였다. 이에 따라 클래스 불균형을 최소화하면서도 학습 가능성을 확보하기 위해, 각 클래스당 300장씩 균등 샘플링하는 전략을 적용하였다. 이 과정을 통해 전체 571개 클래스에 대해 총 165,221 장의 학습용 데이터셋을 재구성하였다.

4.1.3. 데이터 정제 및 전처리

샘플링된 데이터는 다음과 같은 전처리 과정을 진행하였다.

- JSON 기반 annotation을 YOLO 포맷으로 변환
- YOLO v8 입력 해상도(640*640)에 맞춘 자동 리사이징 및 비율 보정

이와 같은 전처리를 통해 학습 데이터의 형식 일관성을 확보하고, 안정적인 모델 학습 환경을 구축하였다.

4.1.4. 데이터셋 분할 비율

최종 데이터셋은 Train : Validation = 9.1 : 0.9 비율로 분할하여 학습 및 검증에 활용하였다.

- Train 데이터: 165,221장
- Validation 데이터: 16,201장
- 총 사용 이미지 수: 171,300장
- 클래스 수: 571개

4.2. 학습 과정 및 하이퍼파라미터

4.2.1. 전이 학습 전략

본 프로젝트에서는 초기 가중치 대신, 대규모 객체 탐지 데이터셋(COCO)으로 사전학습된 YOLO v8 Small의 사전학습 가중치를 초기값으로 사용하는 전이 학습 기법을 적용하였다. 이를 통해 비교적 제한된 데이터 규모에서도 특징 추출 성능을 효과적으로 확보하고, 학습 수렴 속도를 가속화할 수 있었다.

4.2.2. 학습 환경

- 운영 환경: Windows 11 기반 로컬 개발 환경
- 프레임워크: Python 3.10, PyTorch
- GPU: NVIDIA RTX 4070
- 학습 방식: 로컬 환경에서 단일 GPU 기반 학습 수행

4.2.3. 입력 해상도

이미지 해상도는 YOLO 모델의 기본 입력 해상도인 640 * 640으로 설정하였다.

4.2.4. 옵티마이저, 학습률 및 데이터 증강 설정

옵티마이저는 별도로 변경하지 않고 YOLO v8 기본 옵티마이저 설정을 그대로 사용하였으며, 학습률 또한 기본값인 0.01을 유지하였다.

모델의 일반화 성능을 향상시키기 위해 다음과 같은 데이터 증강 하이퍼파라미터를 직접 설정하여 적용하였다.

- hsv_h: 0.015 (색조 변화)
- hsv_s: 0.7 (채도 변화)
- hsv_v: 0.4 (명도 변화)
- degrees: 10 (+/- 10도 회전)
- fliplr: 0.5 (좌우 반전)
- flipud: 0 (상하 반전, 음식은 뒤집히면 쏟아지기 때문에 0으로 설정)
- mosaic: 1.0 (모자이크 기법)

이를 통해 색상 변화, 회전, 좌우 반전, 배경 합성 등 실제 촬영 환경에서 발생할 수 있는 다양한 조건을 학습에 반영하여 모델의 강건성을 향상시켰다.

4.2.5. 학습 과정 모니터링

학습 과정에서는 Train Loss, Validation Loss, Precision, Recall, mAP 지표를 지속적으로 모니터링하였다. 모든 학습 로그와 결과 그래프는 Food_Detection_Project\train_result_300sample 디렉토리에 자동 저장되었으며, 이를 기반으로 성능 수렴 여부와 과적합 발생 가능성을 분석하였다.

4.3. 성능 평가

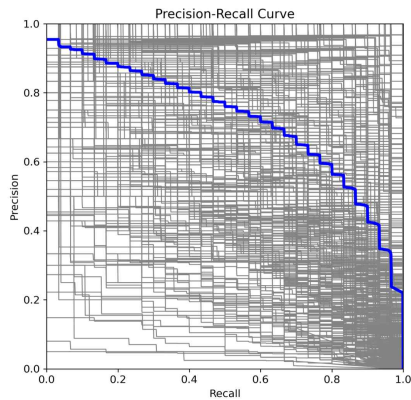
4.3.1. 정량적 성능 평가

모델의 객체 탐지 성능 평가는 객체 탐지 분야의 표준 지표인 Precision, Recall, mAP50, mAP50-95를 기준으로 수행하였다. 최종 Validation 데이터셋 기준 성능은 다음과 같다.

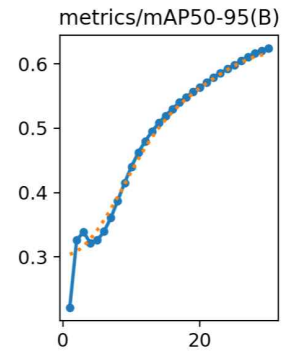
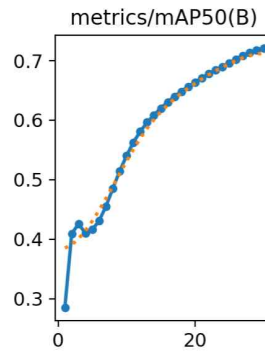
- Precision: 0.648
- Recall: 0.7
- mAP50: 0.721
- mAP50-95: 0.624
- Best F1-score: 0.66 (confidence threshold = 0.242)

mAP50 기준 0.72의 성능은 571개의 대규모 다중 클래스 객체 탐지 문제를 대상으로, 클래스 당 약 300장의 제한된 학습 데이터와 30 Epoch 학습 조건이라는 제약 환경을 고려할 때 상당히 유의미한 탐지 성능을 확보한 결과라고 판단된다.

특히 Recall 값이 0.7 수준으로 비교적 높게 형성되어 있어, 실제 서비스 환경에서 음식 객체를 놓치지 않고 탐지하는 능력이 우수함을 확인할 수 있다. 반면 Precision은 0.65 수준으로 나타나 일부 클래스에서는 오탐이 존재함을 시사한다.



Precision-Recall Curve



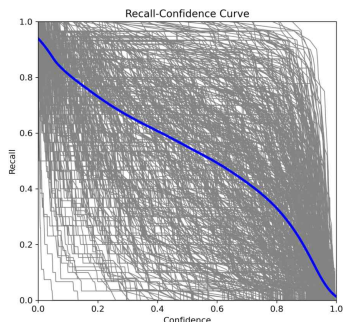
mAP50, mAP50-95 평가 결과

4.3.2. Confidence Threshold 기반 성능 분석

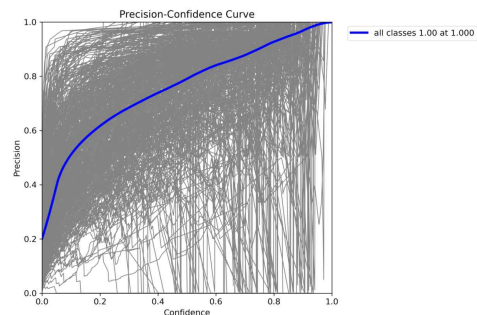
F1-Confidence Curve 분석 결과, confidence 값이 약 0.242에서 F1-score가 최대값 0.66을 기록하였다. 해당 지점은 Precision과 Recall의 균형이 가장 안정적으로 형성되는 구간으로, 본 실험 환경에서

일반적으로 confidence 값이 낮을수록 Recall은 증가하나 오탐지가 증가하며, 반대로 confidence 값이 높아질수록 Precision은 증가하지만 미탐지가 증가하는 트레이드오프 관계가 존재한다. 본 실험 결과 역시 이러한 경향을 동일하게 보였으며, 약 0.25 내외의 confidence 구간에서 Precision-Recall 균형이 가장 안정적으로 유지됨을 확인하였다.

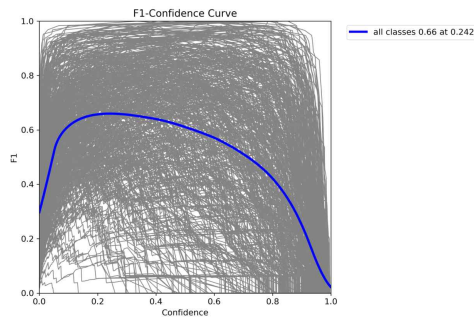
다만, 본 시스템에서는 confidence threshold를 고정값으로 강제 적용하지 않고, YOLO추론 결과와 Gemini 기반 보완 분석을 병행하는 구조로 설계하여 threshold에 대한 의존성을 최소화하였다.



Recall-Confidence Curve



Precision-Confidence Curve



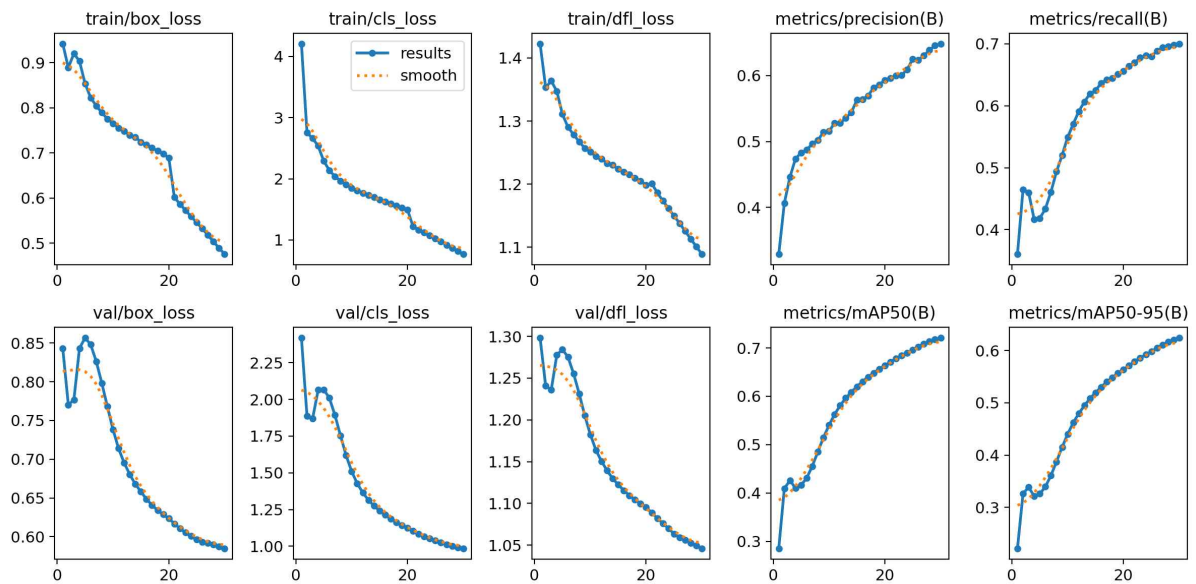
F1-Confidence Curve

4.3.3. Loss 곡선 기반 학습 수렴 분석

Train/Validation Loss 곡선 분석 결과, 다음과 같은 경향이 관측되었다.

- train/box_loss, train/cls_loss, train/df_l_loss: Epoch 증가에 따라 안정적으로 감소
- val/box_loss, val/cls_loss, val/df_l_loss: 초기 변동 이후 점진적 수렴
- metrics/mAP50, metrics/mAP50-95: Epoch가 증가함에 따라 지속적인 성능 향상

특히 20 Epoch 이후부터 Train Loss와 Validation Loss 간의 격차가 크게 벌어지지 않고 안정적으로 유지되어, 과적합 없이 일반화 성능이 확보된 상태에서 학습이 종료되었음을 확인할 수 있다.



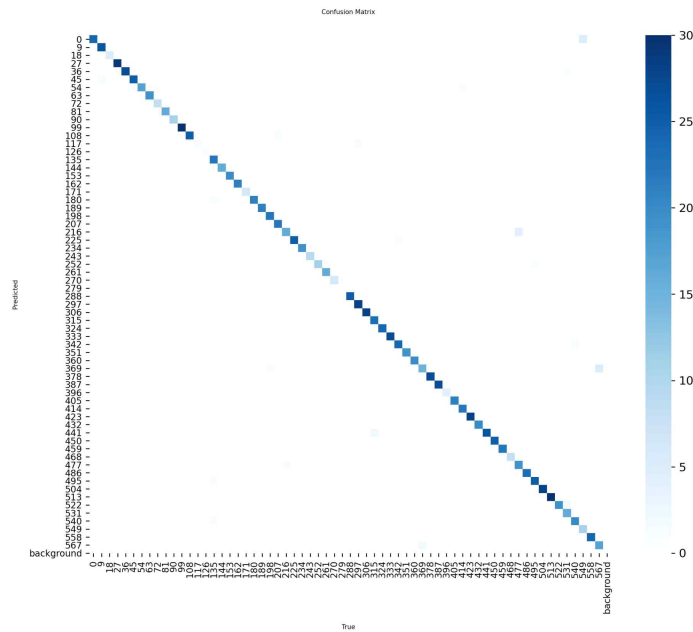
Train/Validation Loss 및 mAP 수렴 곡선

4.3.4. Confusion Matrix 기반 오인식 패턴 분석

Confusion Matrix 및 정규화된 Confusion Matrix 분석 결과, 대부분의 클래스에서 대각선 방향(True Positive)이 강하게 형성되어 정상적인 분류가 이루어졌음을 확인하였다. 즉, 전체적으로 모델이 각 음식 클래스를 비교적 명확하게 구분하고 있음을 의미한다.

다만, 형태와 색상이 유사한 국물 요리, 비슷한 조리 방식(튀김, 볶음)을 가진 음식, 배경이 복잡한 경우 오탐되는 경우가 발생하는 것을 확인하였다. 이는 음식 이미지 특성상 시각적 특징이 매우 유사한 클래스가 다수 존재하는 데이터셋 구조적 한계에 기인한 것으로 판단된다.

4.3.5. 하이브리드 보완 전략(Gemini 연동)의 효과



Confusion Matrix

본 프로젝트에서는 YOLO 모델의 탐지 confidence가 낮거나, 다중 후보가 동시에 검출되는 경우 Google Gemini 2.5 Flash API를 활용한 2차 텍스트.맥락 기반 검증 단계를 추가 적용하였다. 이를 통해 다음과 같은 효과를 확보하였다.

- YOLO 오탐 발생 시, 성분표·제품명 텍스트 기반 보정 가능
- 시각적으로 유사한 음식 간 오인식 발생률 감소
- 최종 사용자에게 제공되는 알레르기 위험 판정의 신뢰도 향상

즉, 단일 비전 모델의 한계를 LLM 기반 맥락 분석으로 보완하는 하이브리드 추론 구조를 통해 실서비스 적합성을 강화하였다.

4.3.6. 종합 성능 평가 요약

본 프로젝트에서 구축한 YOLO v8 Small 기반 음식 인식 모델은 다음과 같은 성능적 특징을 가진다.

- 571개의 대규모 클래스 처리 가능
- mAP50 기준 약 0.72 수준의 실용적 탐지 성능 확보
- 제한된 GPU환경 + 30 Epoch 학습 조건에서도 안정적인 수렴 달성
- LLM(Gemini) 연동을 통해 비전 모델 한계 보완 및 서비스 신뢰도 향상

이를 통해 본 프로젝트에서 개발한 모델은 추가적인 학습을 거친다면 실제 웹 서비스 환경에 충분히 적용 가능한 성능을 갖춘 시스템임을 확인하였다.

5. 주요 기능 구현

5.1. 사용자 맞춤형 필터링 기능

본 서비스는 사용자의 알레르기 정보를 기반으로 개인 맞춤형 식품 검색 결과를 제공한다. 사용자는 마이페이지에서 본인의 알레르기 유발 성분을 선택할 수 있으며, 해당 정보는 MySQL 데이터베이스에 저장된다.

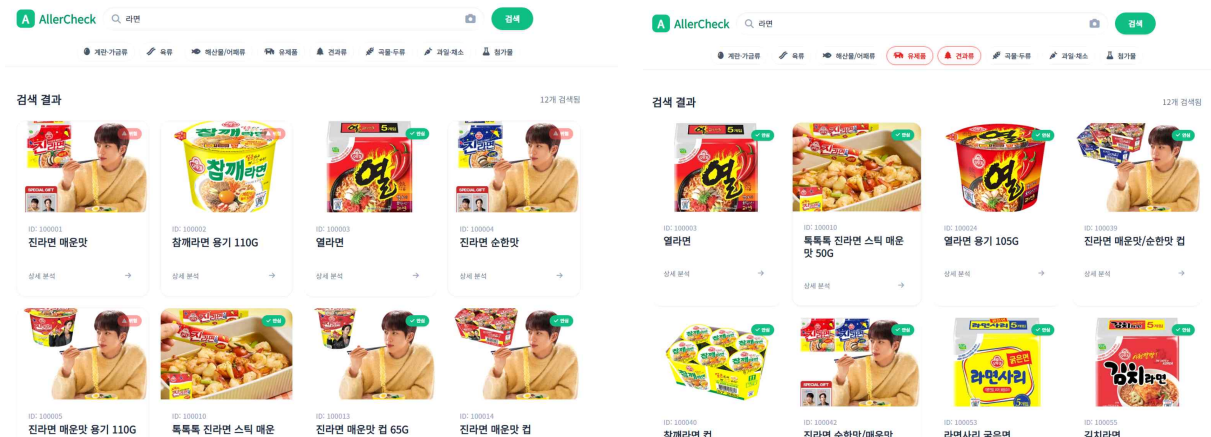
검색 시 서버는 다음 절차에 따라 필터링을 수행한다

- 사용자의 알레르기 정보 조회
- 선택된 식품에 포함된 성분 목록 조회
- 두 집합 간의 교집합 여부 판단
- 일치 성분이 존재할 경우 '주의' 표시

이를 통해 사용자는 단순한 제품 검색을 넘어, 자신의 건강 상태에 최적화된 안전한 식품 추천 결과를 제공받을 수 있다.



사용자 알레르기 정보 관리



필터링 전 검색 화면

필터링 후 검색화면

5.2. 이미지 기반 검색 기능

사용자는 텍스트 입력 없이 음식 이미지를 업로드하는 것만으로 제품을 검색할 수 있다. 해당 기능은 YOLO v8 기반 객체 탐지 모델을 이용하여 구현되었다. 처리 흐름은 다음과 같다.

- 사용자가 웹 UI에서 이미지 업로드
- 이미지가 FastAPI 서버로 전송
- YOLO v8 모델을 통해 음식 객체 탐지 및 클래스 예측
- 예측된 클래스 명을 사용자에게 반환

또한 YOLO의 예측 결과가 모호하거나 상위 클래스 간 점수 차이가 작을 경우 Gemini 2.5 Flash API를 호출하여 이미지에 대한 2차 분석을 수행함으로써 복잡한 음식 이미지에서도 인식 정확도를 보완하는 하이브리드 추론 구조를 적용하였다.

안심 먹거리, AllerCheck

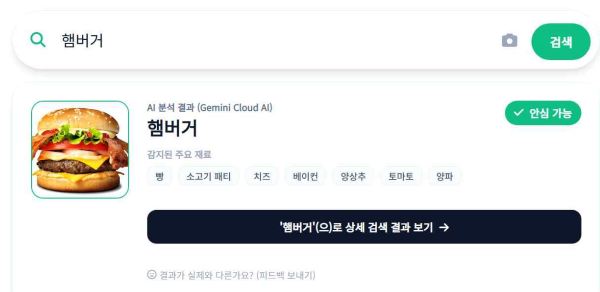
admin님, 등록된 알레르기(2개)를 기준으로 안전한 식품을 찾아드릴게요.



YOLO v8 이미지 검색

안심 먹거리, AllerCheck

admin님, 등록된 알레르기(2개)를 기준으로 안전한 식품을 찾아드릴게요.



Gemini 2.5 Flash API 이미지 검색

5.3. 성분표 분석 및 자동 입력 기능(관리자 전용 기능)

본 기능은 제품 포장지의 성분표 이미지를 자동으로 분석하여 텍스트 데이터로 변환하는 기능이다. 사용자(관리자)는 스마트폰으로 촬영한 성분표 이미지를 업로드하면, 시스템은 다음과 같은 과정을 통해 성분 정보를 자동으로 추출한다.

- 성분표 이미지 업로드
- Gemini 2.5 Flash API를 이용한 OCR 및 자연어 처리
- 추출된 성분 텍스트 정규화
- 알레르기 유발 성분 자동 매핑
- 결과를 사용자 화면에 출력 후 체크박스에 체크

이를 통해 사용자는 성분표를 직접 입력할 필요 없이 자동화된 성분 분석 결과를 즉시 확인할 수 있으며, 알레르기 성분 포함 여부 또한 자동으로 판별된다.

성분표 분석 결과

분석 후 자동 체크

5.4. 관리자 기능

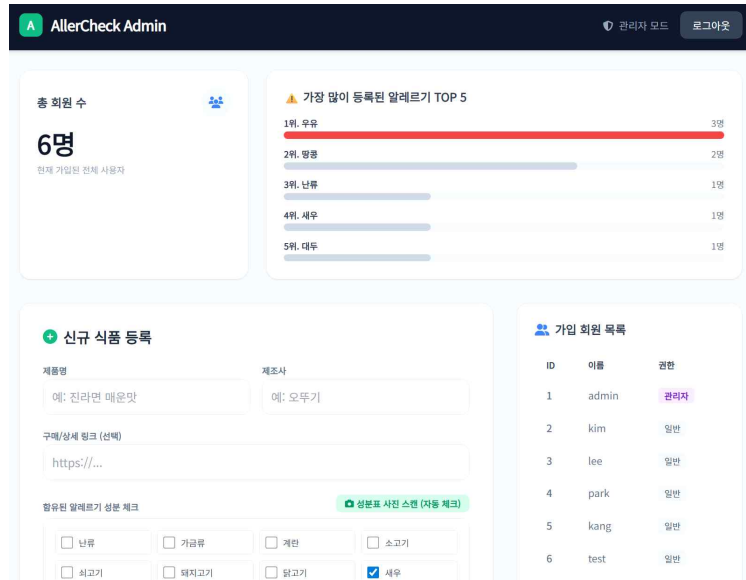
관리자 페이지는 시스템 데이터의 기본적인 관리와 통계 확인이 가능하도록 구성하였다. 관리자 기능은 일반 사용자와 권한이 분리된 계정으로만 접근이 가능하며, FastAPI 서버의 인증 로직을 통해 접근이 제한된다.

현재 구현된 기능은 다음과 같다.

- 상품 정보 등록 및 삭제
- 가장 많이 등록된 알레르기 Top 5 통계
- 전체 사용자 목록 조회

현재 구현된 관리자 기능은 기본적인 데이터 관리 및 통계 확인 중심의 최소 기능 수준으로 구현되었다. 향후 개선 과제로는 다음과 같은 확장이 가능하다.

- 사용자 권한 수정 기능
- 로그 기록 기반 이상 행위 탐지
- 상품 정보 수정
- 성분 데이터 수동 보정



관리자 페이지

5.5. 사용자 피드백 기반 데이터 수집 기능

본 프로젝트에서는 YOLO 기반 이미지 인식 결과의 품질을 향후 지속적으로 개선하기 위한 사용자 피드백 기반 데이터 수집 기능까지 부분적으로 구현하였다.

사용자가 이미지 검색 결과를 확인한 후, 예측 결과가 실제 음식과 다를 경우 피드백을 제출할 수 있는 버튼을 제공하며, 사용자는 해당 이미지의 실제 음식명을 직접 입력할 수 있다. 이때 시스템은 다음과 같은 절차로 데이터를 저장한다.

- 사용자가 잘못 인식되었다고 판단하여 피드백 버튼 클릭
- 실제 음식명(정답 라벨) 텍스트 입력
- 원본 이미지 파일을 서버 로컬 디렉토리에 별도 저장
- 정답 라벨, 파일 이름, 입력 시각 정보를 로그 파일로 별도 기록

이 과정을 통해 오탐지 이미지와 그에 대한 정답 라벨이 자동으로 누적되는 구조를 구축하였으며, 이는 향후 모델 재학습 시 Hard Sample로 활용 가능한 학습 자산 데이터셋으로 전환될 수 있는 기반이 된다.

현재 단계에서는 자동 재학습 파이프라인까지는 구현하지 못했으나, 실제 사용자 환경에서 발생하는 오인식 데이터를 운영 중에 지속적으로 축적할 수 있는 구조를 확보했다는 점에서 MLOps 관점의 데이터 선순환 구조의 기초를 구현했다고 생각한다.

6. 문제 해결 및 성과

6.1. 기술적 이슈 해결

6.1.1. 학습 시간 과다 문제 및 Epoch 조정

초기에는 충분한 모델 수렴을 기대하고 Epoch를 100으로 설정하여 학습을 시도하였다. 그러나 실제 실험 결과, 1 Epoch 당 약 40분 내외의 학습 시간이 소요되었고, 100 Epoch 전체 학습에는 수십 시간 이상의 연산 시간과 높은 전력 소모 비용이 요구되는 문제가 발생하였다.

이에 따라 현실적인 자원 제약을 고려하여 Epoch를 30으로 조정하였으며, 또한 Early Stopping(patience = 5) 전략을 함께 적용하여 불필요한 추가 학습을 방지하면서도 성능 수렴을 효율적으로 달성할 수 있도록 개선하였다.

6.1.2. GPU 메모리 부족(OOM) 문제 및 Batch Size 조정

초기 학습 설정에는 Batch Size를 32로 설정하여 학습을 시도하였으나, 학습 도중 GPU 메모리 부족 오류가 반복적으로 발생하여 약 10 Epoch 부근에서 학습이 강제 종료되는 문제가 발생하였다.

이를 해결하기 위해 Batch Size를 16으로 축소하여 재학습을 진행하였으며, 그 결과 GPU 메모리 오류 없이 30 Epoch 전체 학습을 안정적으로 완료할 수 있었다. 해당 조정은 연산 안정성과 학습 지속 가능성을 확보하기 위한 필수적인 선택이었다.

추가로 Batch Size가 32일 때의 학습 시간이 16일 때보다 무조건 짧을 것이라고 생각했는데, 실제로 모델을 돌려보니 32일 때의 예상 소요 시간이 16일 때보다 길게 나타나는 현상이 발견되었다. 정확한 이유는 파악하지 못했지만, GPU 메모리를 한계까지 사용하다보니 속도가 느려진 것이라고 예상하고 있다.

6.1.3. 시스템 RAM 부족 문제 및 DataLoader Workers 조정

데이터 로딩 속도를 개선하기 위해 초기에는 DataLoader의 workers 값을 8로 설정하였으나, 학습 진행 중 시스템 RAM 부족으로 인해 약 8 Epoch 내외에서 프로세스가 비정상 종료되는 문제가 반복적으로 발생하였다.

이에 따라 workers 값을 4로 하향 조정하여 메모리 사용량을 안정화하였으며, 이후에는 데이터 로딩 및 모델 학습이 RAM 오류 없이 정상적으로 유지됨을 확인하였다. 이를 통해 GPU뿐만 아니라 시스템 메모리 또한 딥러닝 학습의 중요한 병목 요인임을 실험적으로 확인할 수 있었다.

6.1.4. 시각적으로 유사한 음식 간 오인식 문제

Confusion Matrix 분석 결과, 형태·색감·조리 방식이 유사한 음식(예: 탕류, 찌개류, 튀김류 간)에 오인식이 일부 발생함을 확인하였다. 이는 음식 데이터 특성상 클래스 간 시각적 구분도가 낮은 경우가 다수 존재하는 구조적 한계에서 기인한 문제로 판단된다. 이에 대한 보완책으로 YOLO 모델의 confidence가 낮은 경우, Gemini 2.5 Flash API를 활용한 2차 텍스트·맥락 기반 분석을 수행하는 하이브리드 보완 전략을 적용하였다. 이를

통해 단일 비전 모델만 사용했을 때보다 오인식률을 완화하고, 최종 사용자에게 제공되는 알레르기 위험 판정의 신뢰도를 향상시킬 수 있었다.

6.2. 최종 결과물 및 성과

6.2.1. 딥러닝 기반 실서비스 연동 성공

본 프로젝트는 YOLO v8 객체 탐지 모델을 실제 웹 서비스에 연동하여 실시간 이미지 기반 검색 기능을 구현하였으며, 동시에 Gemini 2.5 Flash API를 이용한 성분표 자동 분석 기능까지 통합함으로써, 단순 모델 학습을 넘어 실제 서비스 형태의 AI 시스템을 완성하였다.

6.2.2. 제한된 환경에서의 실용적 성능 확보

제한된 GPU 자원, 30 Epoch 학습, 클래스당 약 300장 수준의 데이터라는 제약 조건에도 불구하고, 본 모델은 다음과 같은 성과를 달성하였다

- Precision: 0.648
- Recall: 0.7
- mAP50: 0.721
- mAP50-95: 0.624

이는 571개의 대규모 다중 클래스 음식 객체 탐지 문제에서 실용적으로 활용 가능한 수준의 성능을 확보했음을 의미한다.

6.2.3. 데이터베이스 - AI - 웹 서비스 통합 성과

1학기 데이터베이스 프로젝트를 통해 구축한 식품·성분·알레르기·교차 반응군 기반 DB 인프라를 바탕으로, 2학기에는 딥러닝 기반 이미지 인식 결과와 LLM 기반 성분 분석 결과를 DB와 실시간으로 연계하는 통합 구조를 완성하였다.

그 결과 다음과 같은 통합 성과를 달성하였다.

- 이미지 업로드 → 자동 음식 인식 → DB 상품 매칭 → 알레르기 위험 판단 → 사용자 화면 출력
- 성분표 이미지 업로드 → 자동 성분 추출 → 알레르기 성분 매핑 → DB 저장

6.2.4. 실무형 AI 프로젝트 수행 역량 확보

본 프로젝트를 통해 다음과 같은 실무형 역량을 종합적으로 확보하였다.

- 대규모 이미지 데이터 전처리 및 YOLO 포맷 변환
- 딥러닝 모델 학습 및 하이퍼파라미터 튜닝
- GPU·RAM 자원 제약 환경에서의 학습 안정화
- FastAPI 기반 AI 추론 서버 구축

- AI - DB - 웹 서비스 통합 아키텍처 설계 및 구현

이는 단순한 딥러닝 모델 실습을 넘어, 실제 산업 현장에서 요구되는 AI 서비스 개발 전 과정을 경험한 성과로 평가할 수 있다.

7. Docker 기반 배포 및 실행 환경 구성

7.1. Docker 기반 배포 환경 설계 및 이미지 구성

본 프로젝트의 Docker 이미지는 NVIDIA 공식 CUDA 이미지를 기반으로 생성하였다. nvidia/cuda:12.1.0-cudnn8-runtime-ubuntu22.04 이미지를 사용하여 GPU 가속이 가능한 실행 환경을 구축하였으며, Python 및 프로젝트 실행에 필요한 필수 라이브러리들은 requirements.txt를 기반으로 일괄 설치하였다.

Dockerfile은 다음과 같은 흐름으로 구성되었다.

- CUDA 및 cuDNN이 포함된 NVIDIA 공식 이미지 기반 설정
- Python 및 시스템 필수 패키지 설치
- requirements.txt를 통한 Python 라이브러리 설치
- FastAPI 서버 및 YOLO 모델 코드 복사
- Uvicorn 기반 FastAPI 서버 실행

해당 구조를 통해 딥러닝 모델 추론, 데이터베이스 연동, 웹 API 서버 기능이 단일 컨테이너 내에서 통합적으로 실행되도록 구성하였다. 이를 통해 개발 환경과 배포 환경 간의 불일치로 인한 오류를 효과적으로 방지할 수 있었다.

7.2. 동적 파일 관리를 위한 볼륨 마운트 및 데이터베이스 연동

본 프로젝트는 사용자 프로필 이미지, AI 분석 결과 이미지 등 실행 중 지속적으로 생성·변경되는 파일을 다수 포함하고 있다. 이러한 파일들을 Docker 이미지 내부에 고정적으로 포함할 경우, 컨테이너 종료 시 데이터가 소실되는 문제가 발생한다. 이를 해결하기 위해 Docker 볼륨 마운트 방식을 적용하였다.

볼륨 마운트 구조는 다음과 같다.

- static/profiles: 사용자 프로필 이미지 저장
- static/ai_temp: AI 분석 임시 이미지 저장
- static/dataset: AI 분석에 대한 사용자 피드백 정보 저장

해당 디렉토리들은 호스트 시스템의 실제 폴더와 컨테이너 내부 /app/static/ 경로에 연결되어, 컨테이너 재시작 이후에도 파일이 유지되도록 설계하였다. 이를 통해 사용자 프로필 이미지 변경, AI 분석 결과 저장 등 동적 파일 관리가 안정적으로 수행되었다.

데이터베이스는 호스트 PC에 설치된 MySQL 서버를 그대로 활용하였으며, Docker 컨테이너 내부에서는 host.docker.internal을 통해 호스트 DB에 접근하도록 구성하였다. 이 방식은 별도의 DB 컨테이너를 사용하지 않으면서도 안정적인 데이터 조회 및 저장이 가능하다는 장점이 있다.

7.3. Docker 실행 결과 및 배포 검증

본 프로젝트의 Docker 컨테이너는 GPU 가속 환경에서 정상적으로 실행되었으며, FastAPI 기반 웹 서버가 외부 포트 8000번을 통해 안정적으로 서비스됨을 확인하였다. 컨테이너 실행 이후 웹 브라우저를 통한 메인 페이지 접속, 사용자 로그인, 사용자 정보 조회, 프로필 이미지 로딩, AI 이미지 분석 기능이 모두 정상적으로 동작함을 검증하였다.

특히 본 시스템은 Docker 볼륨 마운트 구조를 적용함으로써 실행 중 생성되는 데이터가 컨테이너 종료 이후에도 유지되도록 설계되었다. 이에 따라 다음과 같은 운영 안정성을 확인할 수 있었다.

- 컨테이너 재시작 이후에도 사용자 프로필 이미지 정상 유지
- AI 분석 결과 이미지의 지속적인 저장 가능
- FastAPI 서버와 프론트엔드 간 정적 파일 요청 정상 처리

이러한 검증 결과를 통해 본 프로젝트는 딥러닝 모델, 웹 서버, 데이터베이스, GPU 가속 환경이 통합된 실사용 가능한 서비스 수준의 배포 구조를 성공적으로 구축하였음을 확인하였다.

8. 결론 및 향후 과제

8.1. 프로젝트 요약

본 프로젝트에서 수행한 핵심 내용과 성과는 다음과 같이 요약할 수 있다.

첫째, AI-Hub '건강관리를 위한 음식 이미지' 데이터셋을 기반으로 YOLO v8 Small 모델을 전이학습 방식으로 학습하여, 대규모 다중 클래스 음식 객체 탐지 모델을 구축하였다. 제한된 GPU 자원과 30 Epoch 학습 조건에서도 mAP50 기준 약 0.72 수준의 실용적인 성능을 확보하였다.

둘째, 학습된 딥러닝 모델을 FastAPI 기반 웹 서버에 연동하여 사용자가 업로드한 음식 이미지를 실시간으로 분석하고, 자동으로 제품 정보를 검색하는 이미지 기반 검색 기능을 구현하였다.

셋째, Google Gemini 2.5 Flash API를 활용한 성분표 자동 분석 기능을 구축하여, 비정형 형태의 성분표 이미지 및 텍스트로부터 성분 정보를 자동 추출하고, 이를 알레르기 데이터베이스와 연동함으로써 알레르기 유발 성분을 자동 판별하는 지능형 분석 시스템을 완성하였다.

넷째, 사용자가 등록한 알레르기 정보를 기반으로 개인 맞춤형 위험 경고 기능을 구현하여, 검색된 식품에 대해 '안심/주의'와 같은 직관적인 시각 정보로 위험 여부를 제공하였다. 이를 통해 단순 검색 서비스가 아닌 사용자 건강 상태를 고려한 맞춤형 AI 서비스를 구현하였다.

다섯째, 관리자 페이지를 통해 상품 정보 등록·삭제, 가장 많이 등록된 알레르기 TOP 5 통계, 전체 사용자 목록 조회 기능을 구현하여, 최소한의 운영이 가능한 서비스 관리 구조를 갖추었다.

종합적으로 본 프로젝트는 데이터베이스 - 딥러닝 - LLM - 웹 서비스가 유기적으로 결합된 End-to-End AI 서비스 파이프라인을 실제로 구축하고 검증한 실무형 프로젝트라는 점에서 의의가 있다.

8.2. 한계점 및 향후 개선 방안

본 프로젝트가 실서비스 수준의 AI 웹 시스템을 구현하는 데 성공하였으나, 다음과 같은 한계점 또한 존재한다.

첫째, 학습 데이터의 수가 부족했다. 원본 데이터셋은 약 300만 장 규모이지만, 하드웨어 자원 한계로 인해 클래스당 약 300장만 사용하여 모델을 학습하였다. 이로 인해 일부 시각적으로 유사한 음식 클래스 간 오인식이 발생하였다. 향후에는 데이터 수를 확대하거나, 하드 네거티브 샘플을 추가 반영한 재학습을 통해 인식 정확도를 추가로 향상시킬 필요가 있다.

요가 있다.

둘째, 딥러닝 학습 자원의 제약이다. 본 프로젝트는 로컬 환경 단일 GPU에서 학습을 수행하였기 때문에, 학습 시간과 하이퍼파라미터 튜닝에 제약이 있었다. 향후에는 클라우드 GPU 환경 또는 Docker 기반 분산 학습 환경을 구축하여 보다 대규모 실험을 수행할 계획이다.

셋째, 성분표 자동 분석의 정확도 한계이다. Gemini 2.5 Flash API를 활용한 OCR 및 텍스트 분석은 대부분의 경우 안정적인 성능을 보였으나, 조명이 어둡거나 글자가 왜곡된 이미지에서는 일부 인식 오류가 발생할 수 있었다. 또한 텍스트 분석이 오래 걸린다는 단점이 존재한다. 향후에는 전용 OCR 모델과의 결합을 통하여 인식 신뢰도를 더욱 개선할 필요가 있다.

넷째, 관리자 기능의 기능적 한계이다. 현재 관리자 페이지는 상품 관리 및 통계 조회 중심의 최소 기능만 제공하고 있다. 향후에는 사용자 관리 권한 세분화, 로그 기반 비정상 행위 감지, AI 오탐 사례 수집 및 재학습 연계 기능, 상품 정보 수정 기능, 성분 데이터 수동 보정 기능 등 보다 고도화된 운영 기능을 추가할 수 있을 것이다.