Отчет о проекте

Выполнил: Шарипов Булат

Stepik ID: 312344730

Поиск проблемы и описание предполагаемого решения

Проблема: Для людей, которые следят за питанием, довольно тяжело учитывать количество белков, жиров, углеводов, которые они едят за один день. Для этого сейчас нужно смотреть на каждой упаковке БЖУ, самому считать вещества в приготовленной еде, или постоянно искать информацию.

Решение: Веб-сервис, в который можно загрузить только лишь фотографию твоей тарелки, и система сама определить тебе количество белков / жиров / углеводов в вашей еде, и разобьет все по компонентам

Поиск модели, датасета, и обучение модели

Перед обучением модели, требовалось найти хороший датасет. Для этого, я попытался поискать на kaggle. На этом сайте нашелся один подходящий датасет:

https://www.kaggle.com/datasets/sainikhileshreddy/food-recognition-2022

К сожалению, при дальнейшем его изучении выяснилось, что он совершенно непригоден - там допускаются серьезные ошибки в bounding box-ах, а также неправильно распределены лейблы (допустим на кофе стоит лейбл капусты (условно)), поэтому его пришел выбросить.

Дальше я искал на huggingface - перешел во вкладку datasets, здесь сделал фильтр по задаче - object detection, а впоследствии в поисковой строке ввел “food". В итоге, я нашел этот датасет:

https://huggingface.co/datasets/pajama912/FoodLogAthl-218

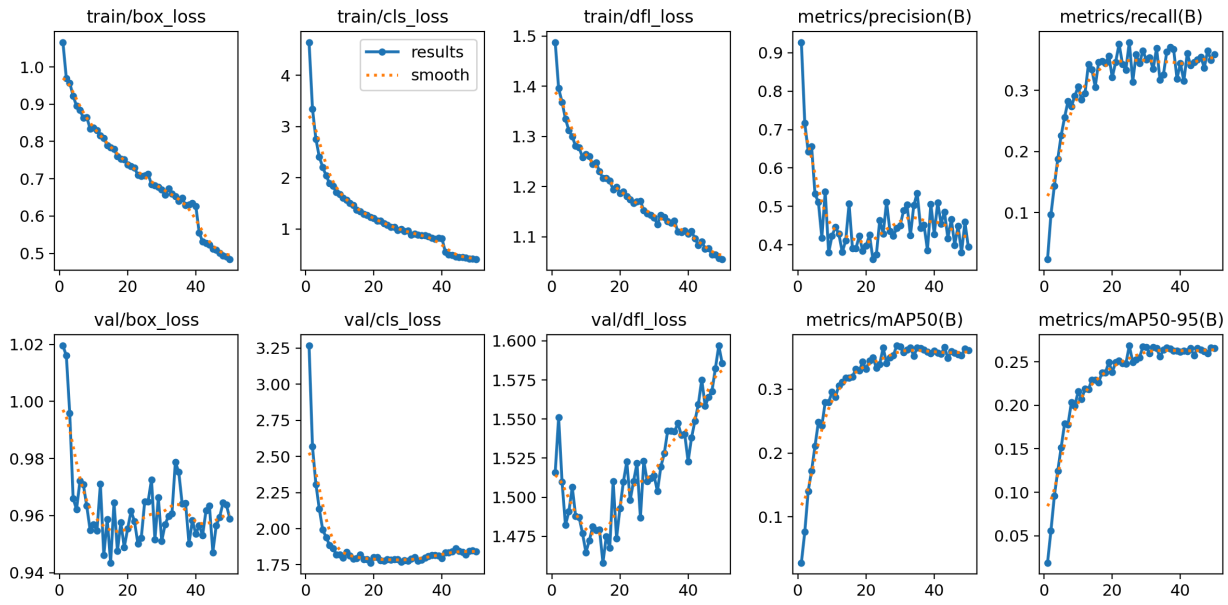
Датасет оказался хороший, и там не было никаких проблем. Однако, он был в другом формате (coco), поэтому я написал свой скрипт чтобы переконвертировать этот датасет в формат ultralytics. Соответствующий скрипт вы можете найти в репозитории, в папке notebooks.

Я начал тренировать модель Ultralytics YOLO v11 medium. Вот некоторе параметры

- Я заморозил все слои, кроме feature extractor-а, так как feature extractor был пред тренирован на обычном мире (Допустим, если бы у меня была задача детектить что-то на рентгеновских снимках, мне стоило бы разморозить feature extractor, либо взять другой backbone)
- В качестве learning rate я взял константу Карпаты
- Также включил стандартные аугментации ultralytics.
- Тренировка происходила на бесплатных GPU T4 ×2 на kaggle
- Соответствующий скрипт тренировки вы можете найти в репозитории

К сожалению, модель натренировалась плохо, результирующий recall ~ 0.35, precision ~ 0.4. Вот несколько возможных причин

- Маленький датасет. Всего было 7к изображений, что наверное маловато для модели с 20 миллионами параметров. Также стоит отметить, что на 212 классов существует всего 7к изображений
- Мало эпох. На графике видно, что в целом модель не переобучалась (хотя в конце видно, что лосс на валидационном сете совсем чуть-чуть начал увеличиваться на последних эпохах)
- Слабая модель. Я взял только medium, так как считал, что на большей модели она легко переобучиться с 7к изображениями.
- Слабые аугментации. Стандартных аугментаций ultralytics очень мало, поэтому, наверное стоило бы добавить больше аугментаций с помощью albumentations.



Так как модель была неуспешна, я пошел другим путем. Я взял модель Florence-2, которая может делать zero-shot object detection. Чтобы детектить еду с помощью этой модели, я сделал следующим образом:

- Просил модель сделать детальное объяснение картинки с помощью токена <MORE_DETAILED_CAPTION>.
- Далее я проходился по всем блюдам, которые у меня есть в базе и проверял, есть ли это блюдо в объяснении картинки модели. Если есть, то я добавлял ключевое слово блюда в лист для будущего промпта.
- Далее я проходился по всем блюдам, которые упомянула модель и в объяснении и которые есть в базе, и просил модель задетектить это блюдо на изображении, с помощью токена <OPEN_VOCABULARY_DETECTION> и ключевого слова блюда.
- В итоге я получал нужные мне bounding box-ы.
- Соответствующий скрипт inference-а можете найти в репозитории

Выбор фреймворка для разработки демо

- Для разработки бекенда я решил использовать FastAPI, так как уже знаком с ним и делал некоторое количество проектов с помощью него

- Для разработки фронтенда я использовал React, TypeScript, TailwindCSS, потому что мне это предложил Claude Sonnet

Разработка демо

- Для разработки бекенда, я попросил Claude Sonnet обернуть мой jupyter notebook в API endpoints и все это еще обернуть в Dockerfile
- Фронтенд я полностью завайбкодил, написав Claude Sonnet-у желаемый результат моего проекта

Встраивание модели-детектора в демо

- Модель детектора встроена в единственный полезный эндпоинт моего проекта. Детектор находит еду на изображении, далее передает все это в калькулятор БЖУ и калорий, и все это отрисовывается на беке. Так как определение граммовки еды - довольно тяжелая задача, пользователь сам может редактировать граммовку для получения более точных вычислений

Тестирование демо

https://drive.google.com/file/d/1zZAocyLnWXWwn-_JrSeTPLUdxpeBd6Oy/view?usp=sharing