

Fridge-to-Table: A Multi-modal Edge-to-Cloud Pipeline for Personalized Recipe Creation

Sridhar Chadalavada Victor Inyang Kam Kasravi Naveen Purushotham

W251 Deep Learning in the Cloud and at the Edge

School of Information, UC Berkeley, Berkeley, CA

{sridhar, vinyang, kamkasravi, naveen.p}@berkeley.edu

Abstract

In this paper, we present Fridge-to-Table, a multi-modal edge-to-cloud pipeline that employs state-of-the-art models, including YOLOv8 for image recognition, GPT-4 for natural language processing and generative recipe creation, and DALL-E 2 for generative image synthesis. This pipeline is designed to provide users with personalized recipe recommendations, cooking instructions, meal descriptions, and images of completed meals based on their refrigerator contents and desired cuisine. Our pipeline incorporates edge-based image recognition using YOLOv8 for efficient ingredient identification and cloud-based generative processes with GPT-4 to create tailored recipes, cooking instructions, and meal descriptions. Meal image synthesis is performed using DALL-E 2. We discuss the architecture, implementation, and potential benefits of Fridge-to-Table, which include enhancing user experiences, optimizing resource utilization, and promoting sustainable food consumption by reducing food waste. This work demonstrates the promising potential of integrating multi-modal data processing with cutting-edge AI models in edge-to-cloud architectures, paving the way for personalized and context-aware applications in the Internet of Things (IoT) domain.

1 Introduction

For many households, cooking at home on a daily basis and deciding what to make can be a daunting task. As the prevalence of obesity and metabolic health syndromes continues to rise in the general US population (Ward et al., 2019) dietary choices have become increasingly important for reducing health risks. Studies have shown that eating food "out of home" is linked to a higher risk of obesity and poorer diet quality (Raber et al., 2016), while cooking at home more often is associated with improved diet quality (Wolfson et al., 2020).

We present a solution that aims to reduce the cognitive burden and barriers to cooking at home

by helping individuals quickly select from meal recipes based on the ingredients they already have in their fridge. This approach can streamline the meal planning process and make it easier for people to choose healthier options, ultimately contributing to better overall health and well-being. Gone are the days of struggling to come up with new meal ideas or trying to replicate a dish from memory. With our AI-powered recipe generator, one can easily discover new recipes based on the contents of your fridge.

By fine-tuning YOLOv8 and leveraging gpt-3.5-turbo and Dall·E 2, our multi-model solution uses a combination of computer vision, natural language processing, and image generation techniques to analyze the key ingredients present and to incorporate users' preference in creating a recipe and corresponding image of the meal. We also explore future extension of our application into ordering ingredients for replenishment using OpenAI's recently released plugins.

2 Background

Some recent areas of research that have informed the development of Fridge-to-Table include:

- **Image Recognition:** Object detection and recognition models, such as YOLOv8 (Terven and Cordova-Esparza, 2023), have been extensively used to identify and localize objects within images. In the context of our pipeline, YOLOv8 is employed for edge-based image recognition, allowing for the efficient identification of food items within a user's refrigerator.
- **Generative AI:** Generative models like GPT-4 (OpenAI, 2023) have demonstrated their ability to generate human-like text in various domains, including recipe generation and cooking instruction creation. The gpt-3.5-turbo model was utilized in our pipeline for

generating personalized recipes, cooking instructions, and meal descriptions based on the user’s available ingredients and desired cuisine. We also utilized gpt-4 in a ChatGPT plugin, as part of being granted early access to the ChatGPT plugin API. This is shown in our [Appendix A](#).

- **Multi-modal AI:** Multi-modal AI models, such as DALL-E 2 ([Ramesh et al., 2022](#)), have shown impressive capabilities in generating realistic images based on textual descriptions. In our pipeline, DALL-E 2 is employed to synthesize images of the completed meal, providing users with a visual representation of the dish they will prepare.

2.1 Large language models

Large language models (LLMs) have revolutionized natural language processing and generation in recent years. The most well-known large language models include OpenAI’s GPT series, with GPT-4 being the latest iteration in this family. In our Fridge-to-Table pipeline, we utilize gpt-3.5-turbo and gpt-4 to generate personalized recipes, cooking instructions, and meal descriptions based on the user’s available ingredients and desired cuisine. Both models are able to create diverse and creative recipe suggestions, along with cooking instructions and detailed meal descriptions. The generative capability of both of these models drastically simplifies the cloud portion of the pipeline. Without leveraging these models, equivalent functionality would need to access one or more recipe databases and parse the content to match on ingredients, cuisine, and other user preferences.

2.2 Prompt Engineering

Prompt engineering for recipe generation is a technique that leverages the power of gpt-3.5-turbo and gpt-4, both LLMs, by carefully crafting the input to guide the model’s generated output. In our case, as shown in [Figure 20](#) we asked gpt-4 to build a prompt from a specification. As [Figure 21](#) shows, gpt-4 provided both a suitable prompt and explanation of why the prompt would be effective.

2.3 Visual Models like DALL-E 2

Recent diffusion models like DALL-E 2 have shown significant improvements over earlier generations in various aspects, such as image synthesis, denoising, and inpainting tasks.

3 Project Architecture and Pipeline

3.1 Infrastructure Setup

We developed a proof of concept at the edge for object based inference of ingredients within a refrigerator and leveraged cloud containers within AWS for fine-tuning the edge model and for processing OpenAI API requests for recipe and image generation. [Figure 13](#) shows the overall set-up used. Although the components of the pipeline are in place, they are not production ready outside of the project’s current setup and require some manual staging.

3.1.1 Setup at the Edge

To prototype the pipeline, a Flask based python application was developed and served from within a locally staged x86 Ubuntu container with a GPU at the edge.

Flask in combination with Jinja and Werzkeug are Python libraries that allowed us to quickly stage a web server, messaging platform through HTTP standard functions between client and server, and create a simple user front-end as shown in [Figure 14](#) that allowed for interactive testing of fine-tuned weights on the edge and selection of user filters such as cuisine.

Communication between the client and server was facilitated through HTTP POST messages, and the detected ingredients were sent in JSON format with inference information preserved for potential future work in training or quality evaluation ([Figure 15](#)).

We leveraged the PyTorch framework in deploying YOLO, which can scale from extra-large down to nano for our edge instance, and similarly was required within our cloud instance for fine-tuning YOLO on refrigerated ingredients.

3.1.2 Setup in the cloud

Two cloud instances were staged. One was used with a NVIDIA Tesla T4, 15102MiB GPU to fine-tune YOLOv5 and YOLOv8 to generate weights to the edge device specific to our task. The other containerized AWS Ubuntu instance was deployed with a Flask application to receive the ingredient list from the edge and send back the recipe and generated image to the edge device for user review. This ran the OpenAI API framework in order to make calls to generate the recipes and image.

3.2 Recipe and Image Generation Algorithm

1. Collect images.
2. Use an object detection model (for example, YOLO) to detect and classify objects.
3. Convert object labels into a text description.
4. Input the text description and cuisine to either gpt-3.5-turbo or gpt-4 models to generate a recipe, cooking instructions and meal description.
5. input the meal description into dall-e 2 to generate a meal image

3.3 Datasets

We used an open-source dataset from Roboflow called ai-cook-lcv4d ver. 4¹, which contains images of day-to-day items in an open refrigerator. The classes consist of 30 common foods, such as apple, beef, bread, milk, spinach, and tomatoes. The ai-cook-lcv4d ver. 4 had 3,050 images in total. The original set of images of the open refrigerator had 516 images and was augmented to 3,050 images through rotation, exposure, blur, salt and pepper noise and cut-outs. The source [train:validation:test] set was split into a ratio of [2896:103:51].

Figure 1 shows the number and location distribution of the classes in the data set. Each class had good representation in the total number of images as seen in the bar plot. The distribution plots also show an x-by-y plot which indicates the physical distribution of the location of the ingredients in the different shelves of the refrigerator. Finally, the width-by-height plot shows the distribution of the size of the ingredients in the data set. Overall, the data set was well-balanced.

Figure 16 shows examples of the ground truth training batch. Our fine-tune training used a batch size of 16 augmented images. Each ingredient class is enclosed in a bounding box and annotated by the class number.

¹<https://universe.roboflow.com/karel-cornelis-q2qqg/aicook-lcv4d>

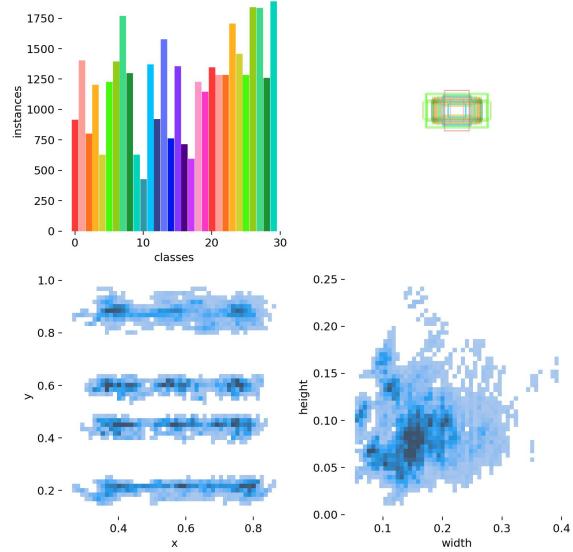


Figure 1: Exploratory data analysis of output class number and location distributions

3.4 Models

3.4.1 Edge Models

We explored YOLOv5 and YOLOv8 models for the edge node. The idea was to train the two models on the cloud and compare their accuracy and network size. We then deployed the best fit model to the edge for inference. From a transfer learning approach, we used Ultralytics YOLO models, trained on the COCO dataset and fine-tuned them with the ai-cook-lcv4d ver. 4 data set. We then imported the fine-tuned weights into the edge client for inference.

Figure 2 shows the loss and accuracy plot for YOLOv8 when fine-tuned on the ai-cook data set. The loss and accuracy plot for our YOLOv5 training can be referenced in 18, and Table 2 shows the comparison numbers. Both models performed well with a high precision and recall rate of 0.96 or higher. The mean-average-precision for an IoU > 0.5 (a good prediction) for YOLOv5 and YOLOv8 was 0.96 and 0.97, respectively. We used YOLOv8 as the final model to be deployed for inference at the edge as it had a slightly higher mAP50 and mAP50:95 values.

Figure 17 shows the confusion matrix for the validation run. Overall, the True and Predicted classes were well correlated. There was a slight false positive rate for 'confused' ingredients with the background. This could be minimized in the future by processing images with higher contrast. Figure 19 shows the final prediction bounding boxes and IoU

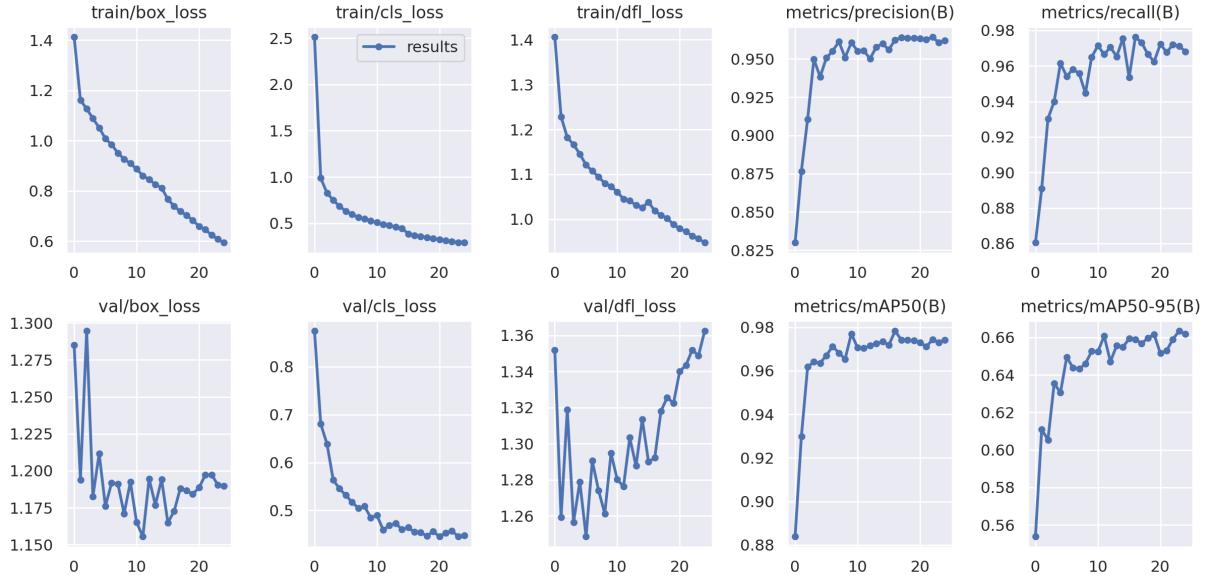


Figure 2: YOLOv8 Fine-tune Training Results

values for the ingredient detection in the validation stage.

3.4.2 Cloud Models

Table 1 lists available models from OpenAI². We used gpt-3.5-turbo to generate text-based recipes, as shown in **Figure 22**. The flow consisted of constructing a few-shot prompt using OpenAI’s ChatML specification³. As noted in the specification, we added a one-shot example using the example_user and example_assistant names. This provides a better context for gpt-3.5-turbo to generate new recipes. **Figure 23** shows the OpenAI call to generate a realistic image of what the meal should look like. We experimented with various approaches to generating an appealing image of the final dish to the user. Using only the recipe name often created images of the ingredients stacked together or a visually unappealing output. The saying "we eat with our eyes first" echoes the sentiment that the appearance of our food is an important factor in our perception of its taste and quality and being able to create an image using DALLE-2 required more than just the recipe name.

Figure 24 shows different images of a Mexican chicken fajita recipe that were generated using the meal name, the recipe, and the detailed generated meal description, respectively. The imaged created from the generated meal description was most vi-

sually appealing, based upon this paper’s authors’ humble opinions.

4 Discussion

4.1 Limitations

A key project limitation was the single data set of images used. They were primarily frontal images of an open refrigerator, which would limit recognition of ingredients captured in any other orientation.

Another limiting factor was not deploying to a mobile edge device such as a mobile phone or Jetson Nano to test our application. This would have provided the opportunity to integrate our solution into a real-time and real world system with realistic edge hardware constraints.

4.2 Next Steps

Based on what is available in your kitchen, the recipe generation for a particular type of cuisine is a straightforward pipeline. What we do not consider is what is missing in the recipe. In this respect, the pipeline could be extended to generate a shopping list of the missing items or even a way to integrate with a grocery service such as instacart⁴, uber-eats⁵ or amazon groceries as we explored in our ChatGPT plugin example (See [Appendix A](#)).

Currently, the input to our model takes the ingredient and cuisine type, We could expand our pipeline to take in more user input such as the the

²See <https://platform.openai.com/docs/models/overview>

³See <https://github.com/openai/openai-python/blob/main/chatml.md#few-shot-prompts>

⁴<https://www.instacart.com/store>

⁵<https://www.ubereats.com/>

prep time, difficulty level, number of servings and ingredients. This will help personalize the experience for the user.

We could also explore leveraging federated learning for data privacy, continuous quality improvement by updating trained weights, and reducing the need to transfer data to cloud.

To further improve our model’s performance, we would label self-generated refrigerator data using tools such as RoboFLow that was not completed due to time constraints. This would provide more classes and real-world scenarios to improve model performance.

Another lever beyond existing Smart Fridges is using the app to capture food items in a user’s pantry to provide a larger view of the availability or lack of ingredients at home.

The current pipeline allows for processing static images and video streams so we could expand by generating a real-time list of ingredients and adding optical character recognition (OCR) to the pipeline to capture ingredients not well classified by our model or to track expiration dates to alert users.

5 Conclusion

The pipeline detailed in this paper combines multiple models at the edge and in the cloud to create an intuitive and user-friendly experience with general utility to most users. By integrating edge devices for efficient object detection and cloud-based generative models, our solution allows users to receive context-aware recipes that include cooking instructions and a meal image quickly. We underscore the importance of large language models and prompt engineering (using ChatML grammar) in creating an effective interaction between the user’s input and the recipe outputs. We hope this particular application can inspire further research in multi-modal pipelines and their applications in various domains, such as personalized nutrition, meal planning, and even dietary management for individuals with specific health conditions.

6 Acknowledgments

We would like to thank Ryan DeJana, Prabhakar Attaluri, Alexandra Savelieva, and the w251 instruction team who supplied feedback and guided the direction of the paper. We acknowledge gpt-4 for its contribution in generating the prompt instruction, and the Roboflow contributors for their generation of the data set.

References

- OpenAI. 2023. [Gpt-4 technical report](#).
- Margaret Raber, Yichen Hua, Elizabeth Barerra, and Michelle Malicki. 2016. Away-from-home food environment and obesity-related health outcomes: A state-level analysis. *Journal of Environmental Health*, 78:8–13.
- Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.
- Juan Terven and Diana Cordova-Esparza. 2023. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. *arXiv preprint arXiv:2304.00501*.
- Zachary J Ward, Claire S Hruska, Christopher D Longnecker, Nikolas H Kassebaum, Stephen S Lim, and Steven L Gortmaker. 2019. Projected u.s. state-level prevalence of adult obesity and severe obesity. *New England Journal of Medicine*, 381:2440–2450.
- Julia A Wolfson, Jacob Lahne, and Michaelanne Raj. 2020. Frequency of cooking at home and diet quality among us adults: National health and nutrition examination survey 2007-2010. *Public Health Nutrition*, 23:997–1006.

Appendix A Recipe Generation Example

KA

Ingredients: chicken breast, onions, garlic, bell pepper
Cuisine: Mexican

Figure 3: Mexican Recipe Input

The screenshot shows a user interface for a recipe generator. At the top left is a red button with the letters 'KA'. To its right, the text 'Used Recipe Generator' is followed by a red button with a fork and knife icon labeled 'LOCALHOST'. A dropdown arrow is positioned next to the localhost button. On the far right are two small icons: a thumbs up and a thumbs down. Below this header, there is a text area containing a generated recipe:

Here's a simple recipe for a Mexican-inspired dish using chicken breast, onions, garlic, and bell pepper:

Recipe: Mexican Chicken Fajitas

Ingredients:

- 1 pound chicken breast, thinly sliced
- 1 large onion, thinly sliced
- 3 cloves garlic, minced
- 1 bell pepper, thinly sliced (any color)
- 2 tablespoons olive oil
- 1 tablespoon chili powder
- 1 teaspoon ground cumin
- 1/2 teaspoon paprika
- Salt and pepper to taste
- Flour tortillas, for serving
- Optional toppings: sour cream, salsa, guacamole, shredded cheese

Figure 4: Mexican Recipe Ingredients

Instructions:

1. In a small bowl, mix together the chili powder, ground cumin, paprika, salt, and pepper. Set aside.
2. Heat the olive oil in a large skillet over medium heat. Add the sliced onions and bell pepper, and cook until they start to soften, about 5 minutes.
3. Add the minced garlic to the skillet and cook for an additional minute, until fragrant.
4. Add the sliced chicken breast to the skillet, along with the spice mixture. Cook, stirring occasionally, until the chicken is cooked through and the vegetables are tender, about 6-8 minutes.
5. Warm the flour tortillas in a dry skillet or in the oven.
6. Serve the chicken fajita mixture in the warm tortillas, topped with your favorite optional toppings such as sour cream, salsa, guacamole, and shredded cheese.

Figure 5: Mexican Recipe Instructions

These Mexican chicken fajitas are a colorful and flavorful dish that will satisfy your cravings for something spicy and savory. The chicken is seasoned with a blend of chili powder, paprika, and cumin, which gives it a smoky and spicy flavor. The onions, garlic, and bell peppers add a touch of sweetness and crunch to the dish. Served on warm tortillas with your favorite toppings, this meal is sure to become a family favorite.

Figure 6: Mexican Fajitas Description

Enjoy your delicious Mexican chicken fajitas with your family and friends!

Would you like to create an Instacart shopping list for this recipe?

Figure 7: Mexican Recipe Instacart Question

KA yes

Figure 8: Mexican Recipe Instacart Answer

The screenshot shows a conversational AI interface with two main sections: "REQUEST TO INSTACART" and "RESPONSE FROM INSTACART".

REQUEST TO INSTACART:

```
{  
  "title": "Mexican Chicken Fajitas",  
  "ingredients": [  
    "1 pound chicken breast",  
    "1 large onion",  
    "3 cloves garlic",  
    "1 bell pepper",  
    "2 tablespoons olive oil",  
    "1 tablespoon chili powder",  
    "1 teaspoon ground cumin",  
    "1/2 teaspoon paprika",  
    "Salt and pepper to taste",  
    "Flour tortillas",  
    "Optional toppings: sour cream",  
    "Optional toppings: salsa",  
    "Optional toppings: guacamole",  
    "Optional toppings: shredded cheese"  
],  
  "question": "Ingredients: chicken breast, onions, garlic, bell pepper\nCuisine:  
Mexican",  
  "partner_name": "OpenAI"  
}
```

RESPONSE FROM INSTACART:

```
{  
  "url": "https://inst.cr/t/bU80emEwakln"  
}
```

Below the interface, there is a message:

Here's your Instacart shopping list for Mexican Chicken Fajitas: [Instacart Shopping List](#).

You can use this link to purchase the ingredients and have them delivered to your doorstep.
Enjoy cooking and savoring your delicious Mexican Chicken Fajitas!

Figure 9: Mexican Recipe Instacart



Figure 10: Mexican Fajitas Recipe

Appendix B Cross Reference Tables and Figures

Table 1: OpenAI Model Descriptions

Models	Description
GPT-4 Limited beta	A set of models that improve on GPT-3.5 and can understand as well as generate natural language or code
GPT-3.5	A set of models that improve on GPT-3 and can understand as well as generate natural language or code
DALL·EBeta	A model that can generate and edit images given a natural language prompt
WhisperBeta	A model that can convert audio into text
Embeddings	A set of models that can convert text into a numerical form
Moderation	A fine-tuned model that can detect whether text may be sensitive or unsafe
GPT-3	A set of models that can understand and generate natural language
CodexDeprecated	A set of models that can understand and generate code, including translating natural language to code

Model	Epochs	P and R	mAP50	mAP50:90
YOLOv5	100	0.96 and 0.96	0.96	0.616
YOLOv8	25	0.96 and 0.97	0.97	0.663

Table 2: Comparison of YOLOv5 and YOLOv8

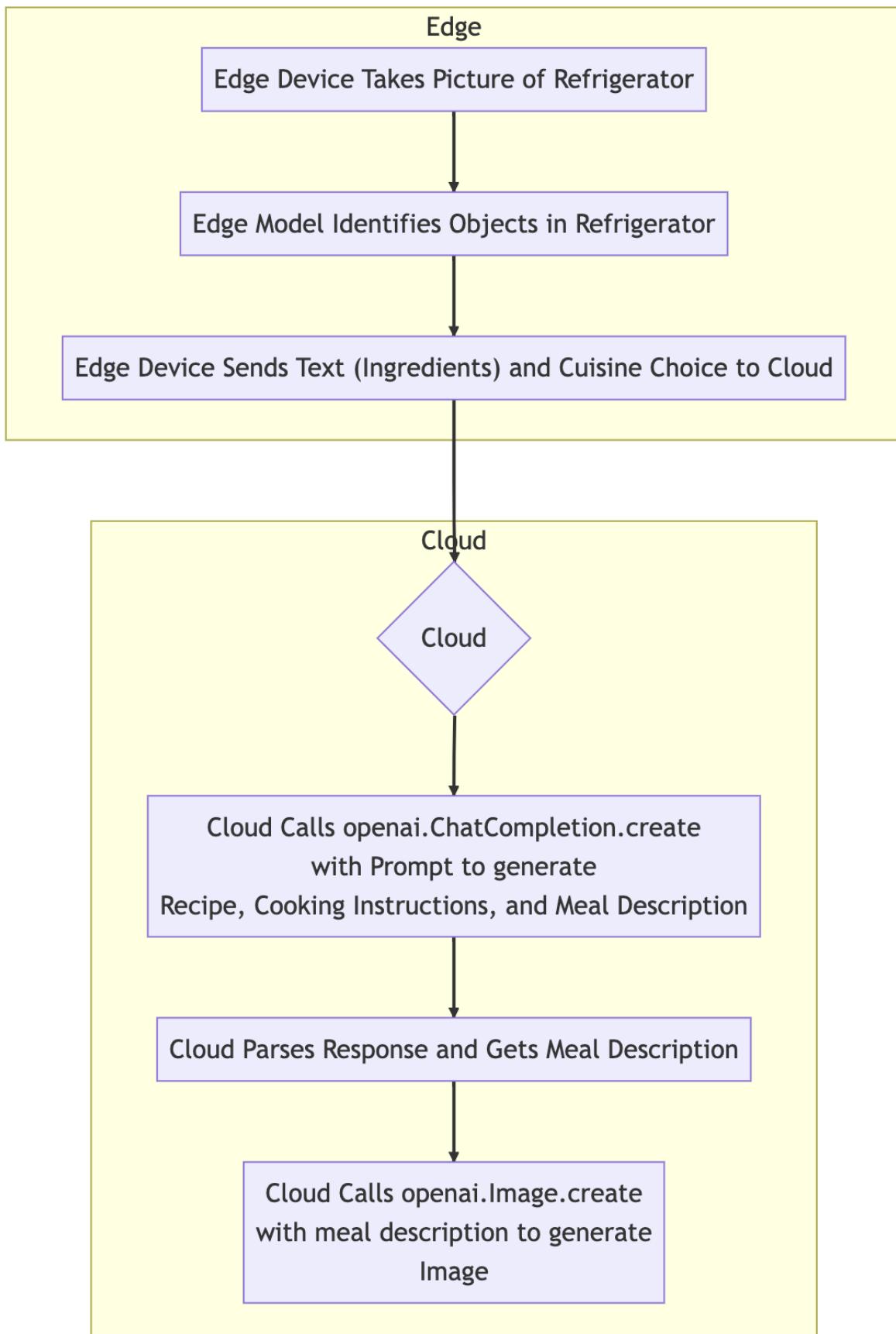


Figure 11: Pipeline Edge to Cloud Diagram

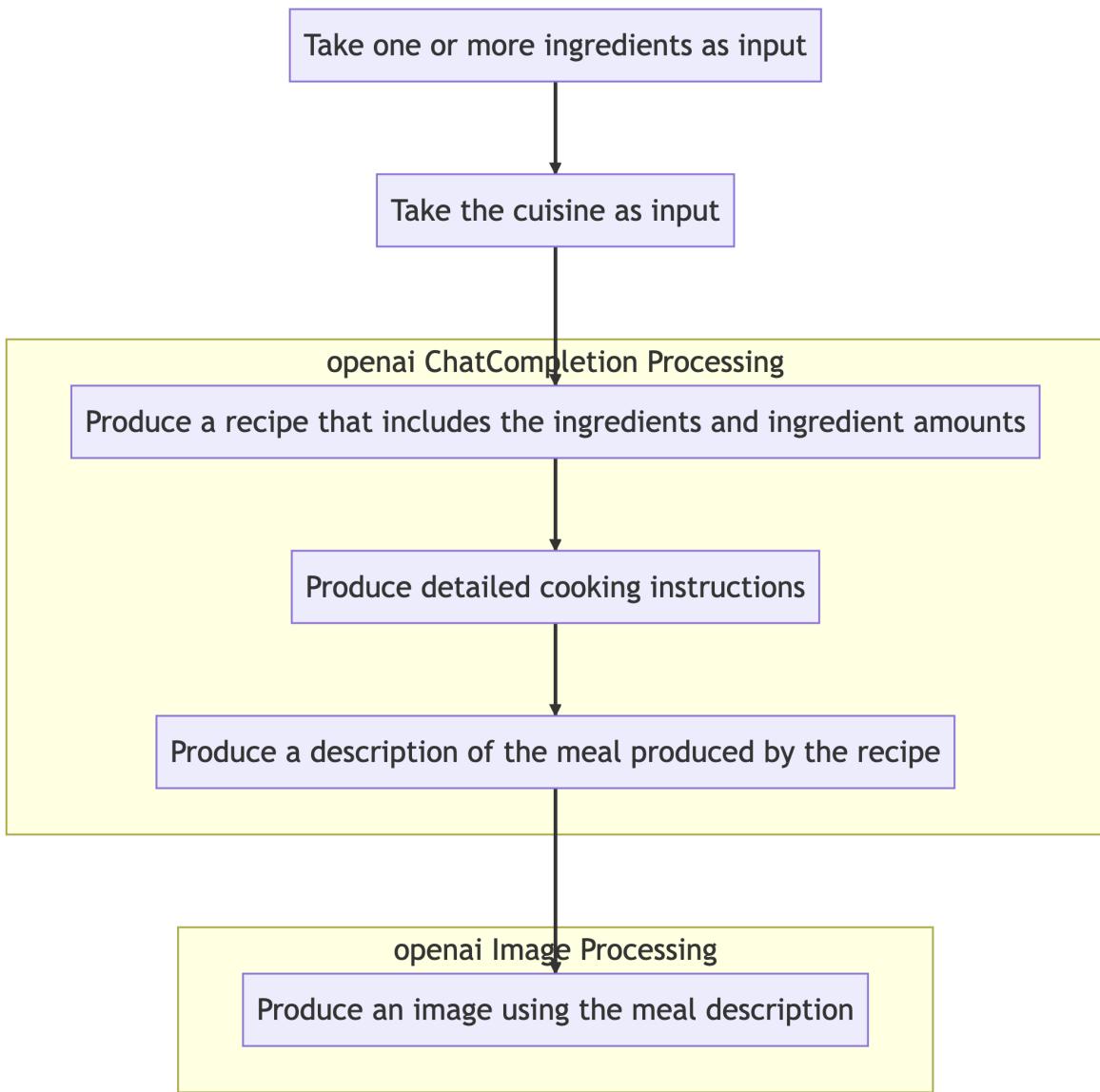


Figure 12: Pipeline in the Cloud Diagram

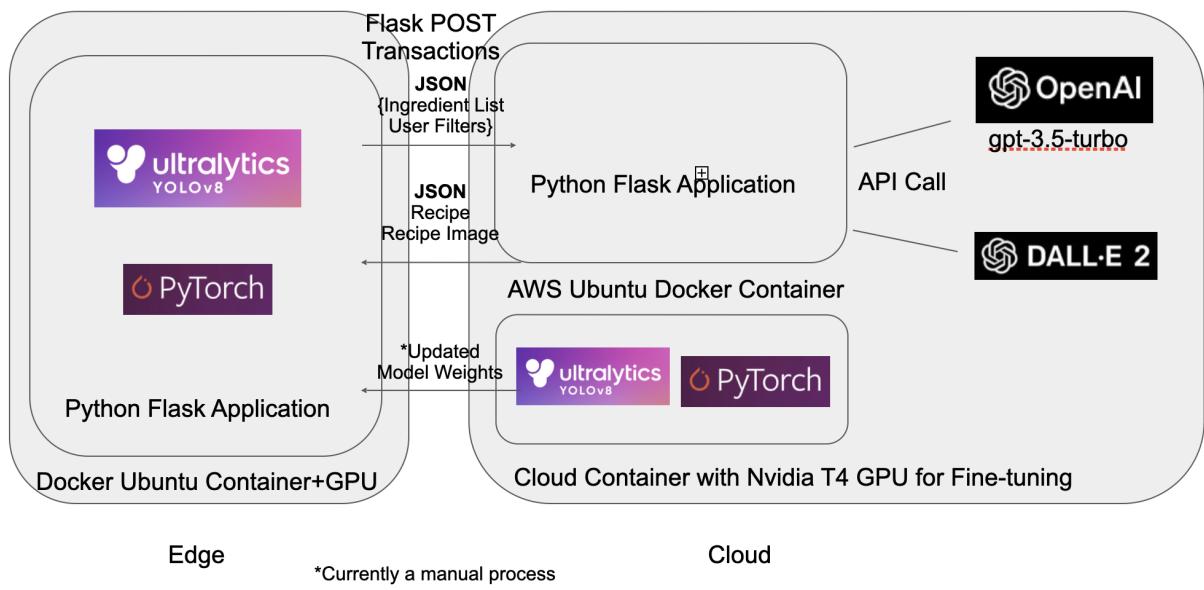


Figure 13: Architecture



Figure 14: Simple Front-end App Prototype on Edge

```
{'predictions': [
    {
        'x': 646.310546875,
        'y': 565.3239135742188,
        'width': 124.0819091796875,
        'height': 119.5931396484375,
        'confidence': 0.8978670239448547,
        'class': 'milk',
        'image_path': 'NA',
        'prediction_type': 'NA'
    },
    {
        'x': 933.970458984375,
        ...
        'cuisine': 'Japanese'
    }
]}
```

Figure 15: Example of how data is stored when transmitted between client and server.

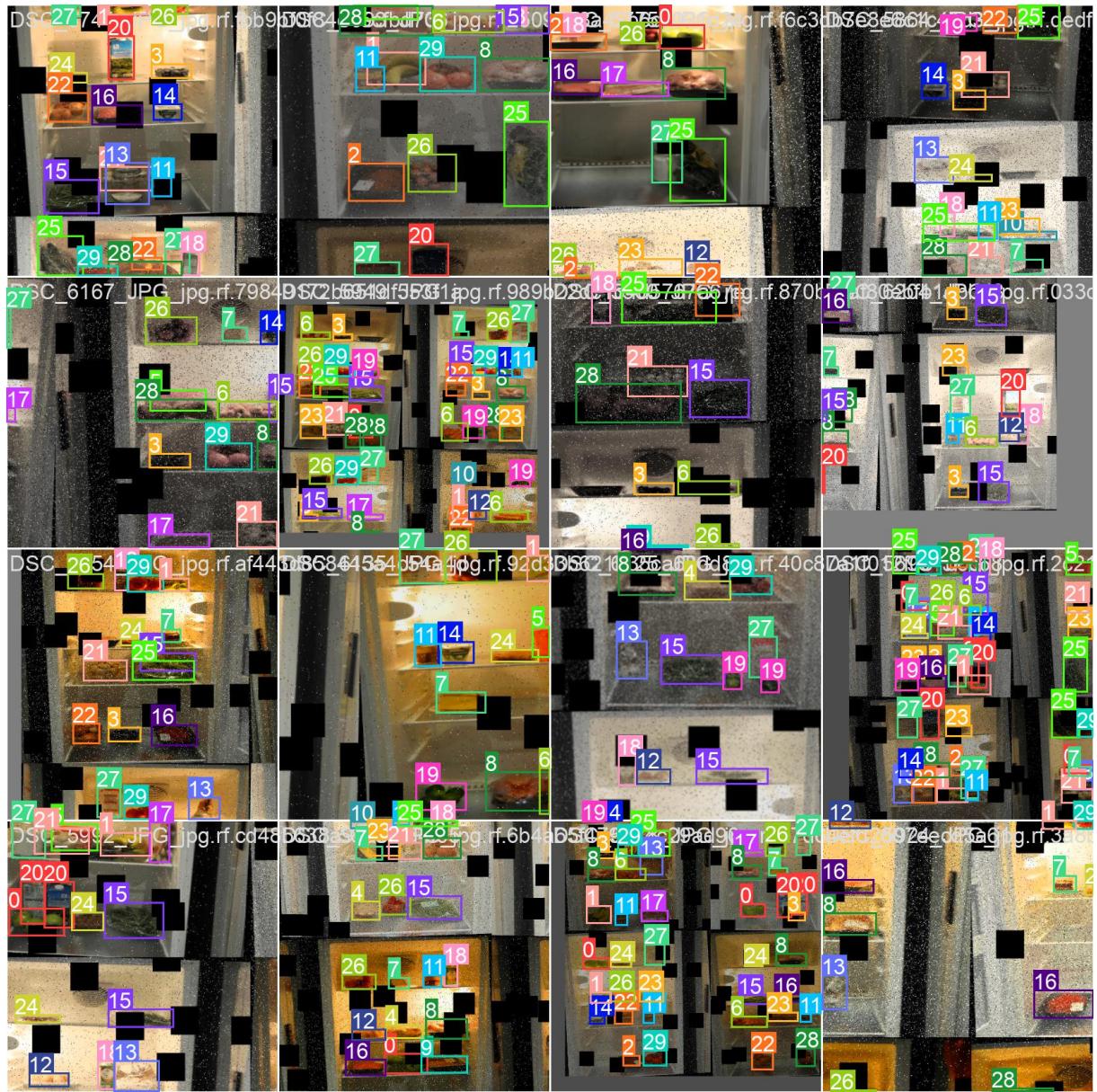


Figure 16: Ground truth training batch

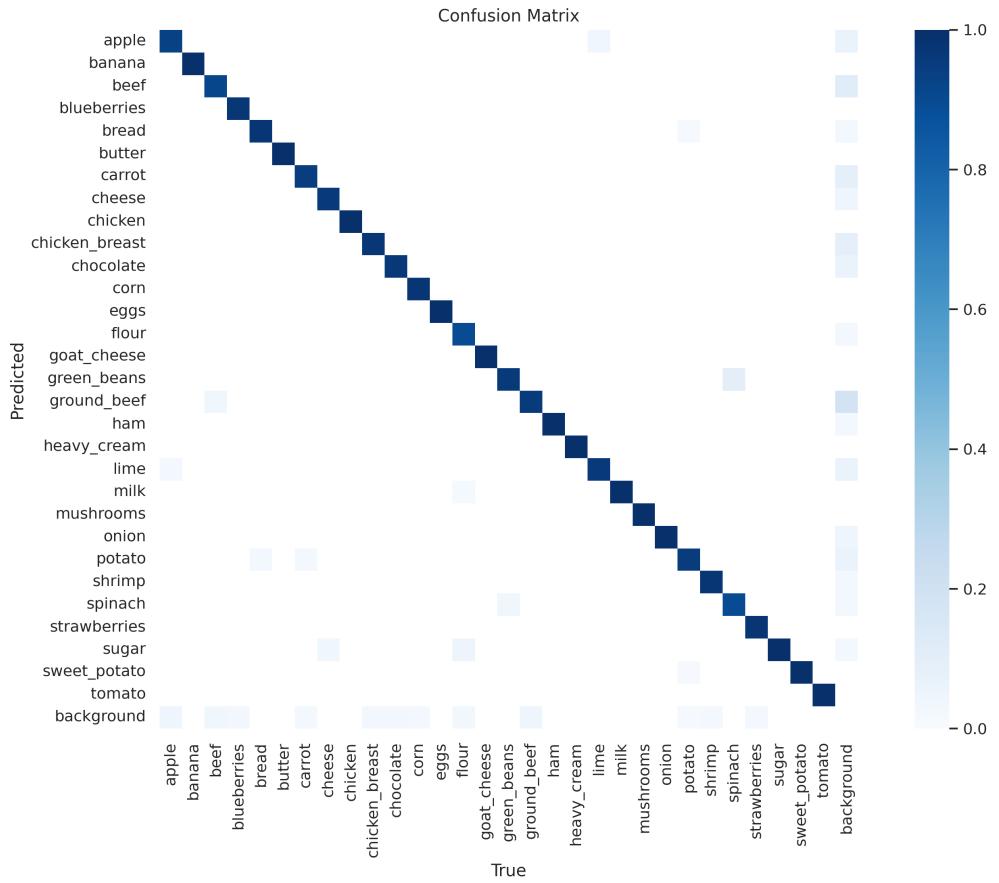


Figure 17: Confusion Matrix

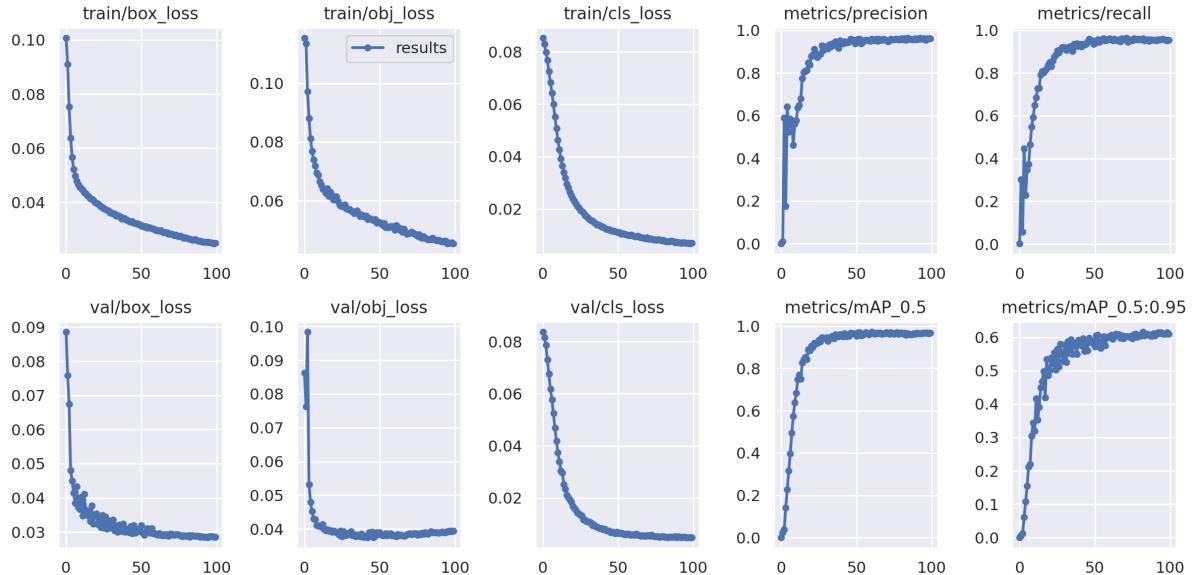


Figure 18: YOLOv5 Results

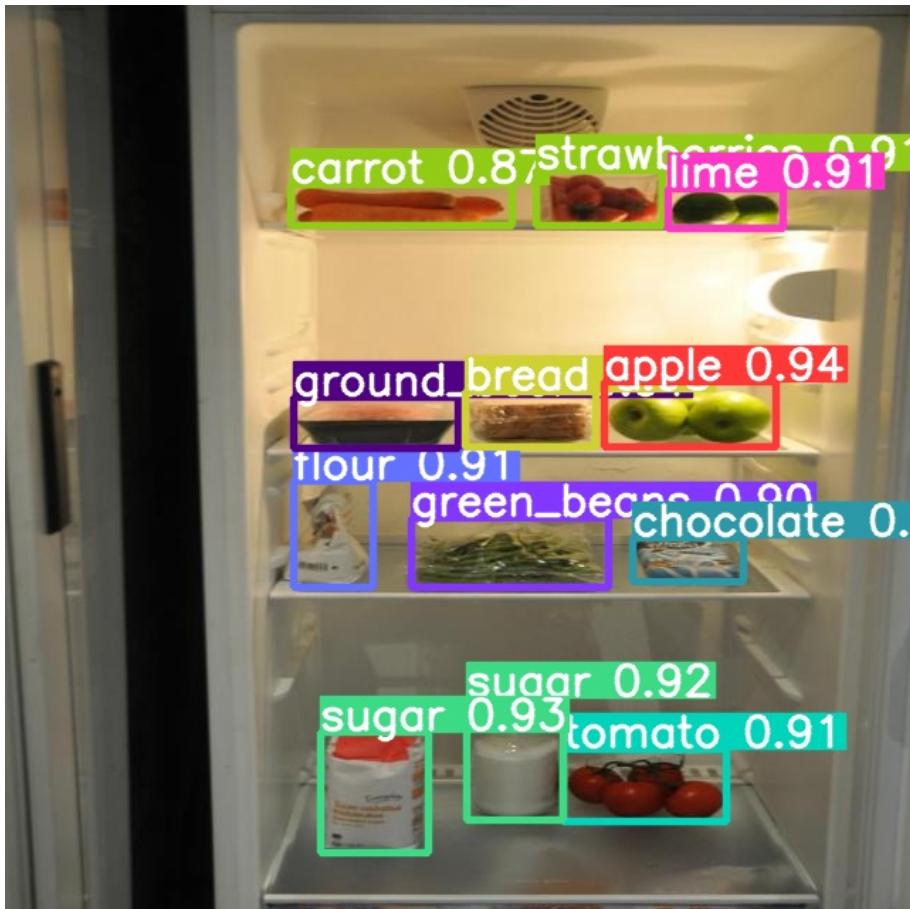


Figure 19: Inference Bounding Box and Class Predictions

KA Provide guidance on how a few-shot prompt should be structured for a recipe generator. 
This recipe generator should do the following:
1. Take one or more ingredients as input.
2. Take a cuisine as the type of recipe that should be generated.
3. Produce a recipe that includes the ingredients and ingredient amounts.
4. Produce detailed cooking instructions.
5. Produce a description of the meal from the recipe that can be used to generate an image.

Figure 20: Recipe Specification



Sure, here is an example of how a few-shot prompt could be structured for a recipe generator that can take ingredients and cuisine as input and produce a recipe with detailed cooking instructions and a meal description that can be used to generate an image:

The screenshot shows a code editor window titled "vbnet". The code is a template for generating a recipe. It starts with two placeholder variables: "Ingredients: [list of ingredients]" and "Cuisine: [type of cuisine]". Below these, a main instruction reads: "Generate a [type of cuisine] recipe that includes [list of ingredients] as the primary ingredients." An "Example:" section follows, with "Ingredients: chicken breast, onions, garlic, bell pepper" and "Cuisine: Mexican". A final line of text, "Generate a Mexican recipe that includes chicken breast, onions, garlic, and", is partially visible at the bottom.

```
Ingredients: [list of ingredients]
Cuisine: [type of cuisine]

Generate a [type of cuisine] recipe that includes [list of ingredients] as the primary ingredients.

Example:
Ingredients: chicken breast, onions, garlic, bell pepper
Cuisine: Mexican

Generate a Mexican recipe that includes chicken breast, onions, garlic, and
```

In this prompt, the user specifies the ingredients they want to use and the type of cuisine they want the recipe to be based on. The prompt then instructs GPT-4 to generate a recipe that includes the specified ingredients and cuisine, as well as ingredient amounts and cooking instructions. The final dish should be described in a way that can be used to generate an image. The example prompt above illustrates how the prompt could be filled out with specific ingredient and cuisine values.

By structuring the prompt in this way, the user can easily generate a recipe for a specific set of ingredients and cuisine without having to provide a full recipe template or complex programming logic. The few-shot prompt is flexible enough to handle a wide range of ingredient and cuisine combinations, and the generated output is customized based on the specific ingredients and cuisine specified in the prompt.

Figure 21: Recipe Specification Response

The screenshot shows a code snippet from "pipeline.py" demonstrating a ChatCompletion API call. The code imports "ChatCompletion" from "openai", initializes a "model" variable to "text-davinci-003", and then creates a message object with a "role" of "system" and a "content" of "SYSTEM_INPUT". It then adds two message objects representing a user and an AI assistant, each with a "role" of "user" or "assistant" and a "content" field containing a JSON-like string. Finally, it calls "ChatCompletion.create" with the "messages" list and returns the response.

```
from openai import ChatCompletion
model="text-davinci-003"
messages = [
    {"role": "system", "content": "SYSTEM_INPUT"},  
    {"role": "user", "content": "Dish: Mexican Chicken and Potatoes Casserole Recipe\nInstructions: \nThis Mexican-inspired casserole is the perfect dish for a cozy night in. With tender chicken breasts, sweet onions, and spicy jalapeño peppers, this dish is sure to satisfy your cravings! Follow these simple steps to make this delicious meal:\n1. Heat the oven to 375 degrees F. In a large skillet over medium-high heat, add olive oil and the chicken breasts. Cook until the chicken is no longer pink, about 5 minutes per side.\n2. Add the onions and jalapeños to the skillet and cook until the vegetables are softened, about 5 minutes.\n3. Stir in the cumin, chili powder, salt, and black pepper. Cook for an additional minute.\n4. Add the shredded cheese and stir until melted.\n5. Remove the chicken and vegetables from the skillet and place them in a baking dish. Top with the melted cheese.\n6. Bake uncovered for an additional 10-15 minutes, or until the cheese is melted and bubbly.\n7. Serve hot with a side of rice and beans.\n8. Enjoy!\n\nNotes: This dish is perfect for a weeknight meal. The flavors come together in a creamy, cheesy sauce that's sure to satisfy any craving. Served over a bed of white rice, this dish is the ultimate comfort food for any night of the week. Enjoy!"},  
    {"role": "assistant", "content": "example_assistant"},  
    {"role": "user", "content": "user_input"},  
    {"role": "assistant", "content": "example_assistant"}  
]
```

Figure 22: ChatCompletion API call to generate a recipe from pipeline.py in our github repo.

```

def generate_image(description):
    response = openai.Image.create(
        prompt=description,
        n=1,
        size="1024x1024"
    )

    image_url = response['data'][0]['url']
    return image_url

```

Figure 23: Image API call to generate an image from pipeline.py in our github repo.



Figure 24: Images generated from dalle-2

Title	URL	Year
Babyagi	https://github.com/yoheinakajima/babyagi	2023
Auto-GPT: An Autonomous GPT-4 Experiment	https://github.com/Torantulino/Auto-GPT	2023
LangChain	https://github.com/hwchase17/langchain	2022

Table 3: Reference table for @misc and @software entries