Project Progress Report for IngrAIdients: An AI tool for ingredient identification.

Felicia Liu
Student# 1006950042
`lfelicial.liu@mail.utoronto.ca`

Anipreet Chowdhury
Student# 1006914396
`anipreet.chowdhury@mail.utoronto.ca`

Siddharth Khanna
Student# 1006773341
`Sid.khanna@mail.utoronto.ca`

Arthur Zhuang
Student# 1006233997
`arthur.zhuang@mail.utoronto.ca`

—-Total Pages: 10

1 Introduction

This project is motivated by the growing trend toward health-conscious living, where the ability to quickly identify ingredients in food can enhance meal preparation, dietary tracking, and nutrition management. As more individuals aim to improve their well-being, there is a demand for tools that facilitate nutrition insights. Our objective is to develop a deep learning algorithm that can accurately detect a short list of ingredients (output) from a single image of a prepared dish (input). The project's value lies in its ability to enable users to monitor their food intake, and understand the nutritional content of their meals in a streamlined manner, thereby simplifying dietary management for a wide range of users. An example of a specific tool user would be those with specific dietary restrictions or severe allergies, who can simply take a picture of their meal, and the tool will identify the ingredients, helping them make safer and more informed food choices.

Machine learning, particularly deep learning, is well-suited for our task of ingredient identification from food images. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) each contribute distinct strengths to the model. CNNs are effective for image analysis, capturing hierarchical features from basic elements like edges and textures to complex patterns such as shapes and specific food items. RNNs complement this by processing sequential information, allowing the model to account for contextual patterns across ingredient sequences. Together, these models enable nuanced differentiation between similar ingredients, which is essential for accurate identification in diverse culinary contexts. With a large dataset of varied food images, this architecture can generalize well to new examples, ensuring reliable classification and enhancing detection accuracy. A diagram of our model is shown in Figure 1.

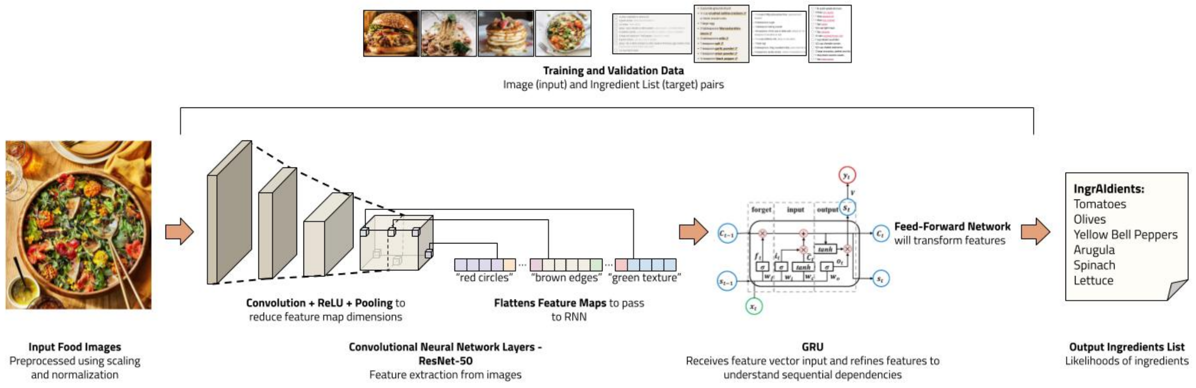

Training and Validation Data
Image (input) and Ingredient List (target) pairs
Convolution + ReLU + Pooling to reduce feature map dimensions
"red circles"
"brown edges"
"green texture"
Flattens Feature Maps to pass to RNN
forget
input
output
Feed-Forward Network will transform features
IngrAIdients:
Tomatoes
Olives
Yellow Bell Peppers
Arugula
Spinach
Lettuce
Input Food Images
Preprocessed using scaling and normalization
Convolutional Neural Network Layers - ResNet-50
Feature extraction from images
GRU
Receives feature vector input and refines features to understand sequential dependencies
Output Ingredients List
Likelihoods of ingredients

Figure 1: System Context Diagram depicting deep-learning integration

2 Data Processing

In this section, we will outline the sources of our collected datasets and explain our rationale for selecting them. We will also provide a detailed description of the data cleaning and formatting steps we followed, with enough specificity such that our data processing procedures could be easily reproduced and plan for testing on never before seen data.

2.1 Collected Datasets

In our initial proposal, we intended to utilize both the Recipe1M+ (Rec) dataset and the Food Ingredients and Recipe Dataset with Images (FIRDI) (Foo) for training our deep learning model. Recipe1M+ contains 1 million recipes and

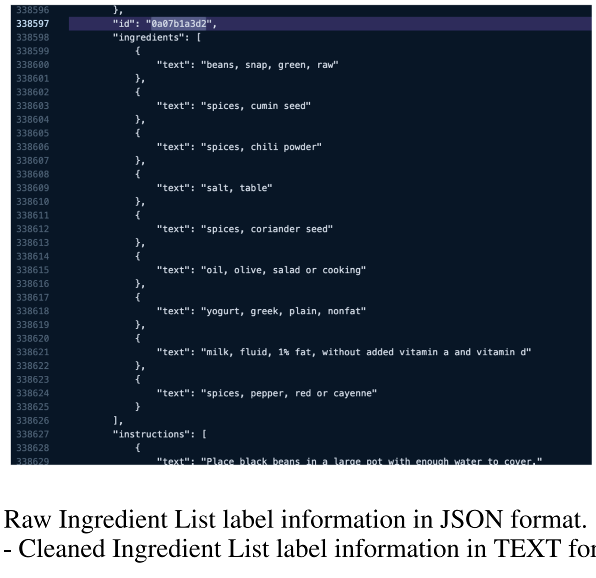
13 million food images, designed for tasks such as recipe-to-image retrieval while the FIRDI offers a variety of recipes and images for ingredient recognition and image-based analysis. As we investigated our datasets further and began processing them for our model, we noticed that FIRDI was scrapped entirely from Epicurious (Epi). Recipe1M+ also includes data from Epicurious as one of the many websites it draws from. By using both datasets to train our model, this overlap could introduce bias for results in this dataset.

Additionally, the CSV structure of FIRDI formats its ingredients as complex strings (e.g., ”['2 large eggs, '1 teaspoon finely chopped rosemary..']”) which complicates our preprocessing, as parsing relevant ingredient names could easily become inconsistent, leading to labeling errors that could impact model performance. Recipe1M+, in contrast, organizes data using JSON objects with a well-structured nutritional layer. Ingredients in this dataset follow a consistent noun-first format (e.g., ”wheat flour, white, all-purpose”) which allows us to easily split the string and selectively include additional descriptors, simplifying the extraction of relevant ingredient names while retaining useful detail. With over 51,000 valid[1]recipes and 700,000 images, Recipe1M+ offers sufficient data to train our model effectively, hence, given the superior data structure, variety, and scale, we have chosen to prioritize it for our model training and processing efforts.

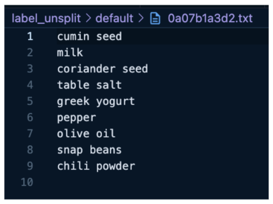
2.2 Ingredient List (Label) Text Based Processing

The goal of our text processing was to transform the JSON nutritional list organization of individual ingredient strings into a form easier for our model to read. For each valid recipe in the dataset, identified by a unique ID, its ingredients are organized in text string objects (Figure 2 (left)) and thus, the primary effort in our text-based processing focuses on these individual strings, aiming to parse out ingredient names that are sufficiently detailed yet still generalizable.

First, brackets were removed from ingredient names (Figure 3 (top)) which often contained only irrelevant metadata or extra descriptions which detract from generalisability. The string was then divided into components based on commas, removed any leading or trailing whitespace, and organized ingredients and descriptors into a list. Next, base ingredient names were assigned using heuristic rules developed through inspection. Our goal was to keep the name tokens concise and focused on useful information. Generally, each ingredient name consists of the first descriptor (index 1) and the main name (index 0). However, for specific descriptors (e.g., ”raw”) and names (e.g., ”spices”), we chose to exclude them, as they did not provide meaningful information (Figure 3 (middle)). Lastly, base ingredient names were processed (Figure 3 (bottom)) by fixing common naming inconsistencies (e.g. plurals are the same ingredient), removing brand names (e.g. delallo) and in individual cases, remapping a name entirely (e.g. ketchup). A final cleaning was performed to ensure that any trailing whitespace is removed before returning the resulting ingredient name. For each recipe ID, its ID is used to name its corresponding labels text file, where we store the processed ingredient list, shown in Figure 2 (right). The processed labels consist of individual ingredients for that recipe, with duplicates removed.



```
        },
        "id": "0a07b1a3d2",
        "ingredients": [
            {
                "text": "beans, snap, green, raw"
            },
            {
                "text": "spices, cumin seed"
            },
            {
                "text": "spices, chili powder"
            },
            {
                "text": "salt, table"
            },
            {
                "text": "spices, coriander seed"
            },
            {
                "text": "oil, olive, salad or cooking"
            },
            {
                "text": "yogurt, greek, plain, nonfat"
            },
            {
                "text": "milk, fluid, 1% fat, without added vitamin a and vitamin d"
            },
            {
                "text": "spices, pepper, red or cayenne"
            }
        ],
        "instructions": [
            {
                "text": "Place black beans in a large pot with enough water to cover."
```

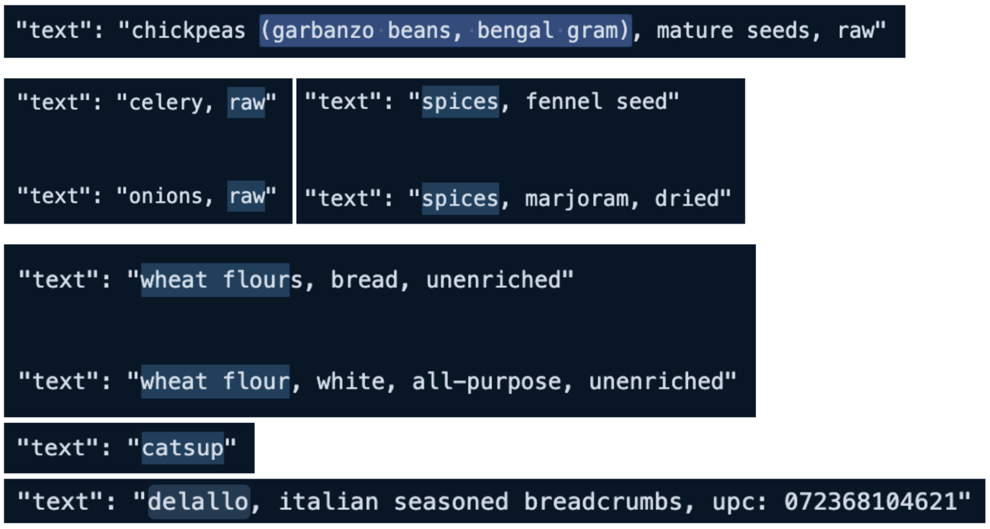


label_unsplit > default > 0a07b1a3d2.txt

```
cumin seed
milk
coriander seed
table salt
greek yogurt
pepper
olive oil
snap beans
chili powder

```

Figure 2: (left) - Raw Ingredient List label information in JSON format.
(right) - Cleaned Ingredient List label information in TEXT format.

[1]Some recipes in Recipe1M+ may lack ingredients or images, as the dataset includes diverse information (e.g., cooking instructions, nutritional facts). For our project, we focus on data points that meet our criteria: containing ingredient-related text and at least one image.

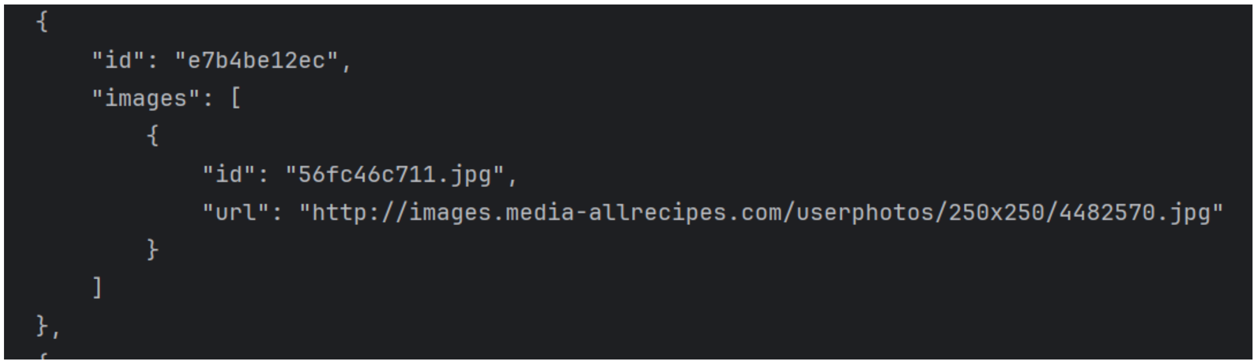


```
"text": "chickpeas (garbanzo beans, bengal gram), mature seeds, raw"

"text": "celery, raw"    "text": "spices, fennel seed"

"text": "onions, raw"    "text": "spices, marjoram, dried"

"text": "wheat flours, bread, unenriched"

"text": "wheat flour, white, all-purpose, unenriched"

"text": "catsup"

"text": "delallo, italian seasoned breadcrumbs, upc: 072368104621"
```

Figure 3: (top) - Brackets contained unnecessary metadata and were removed. (middle) - Heuristics for descriptor (index 1) and the main name (index 0) exclusions. A complete list of first descriptor and the main name exclusions as well as removed brand names, and other remapped ingredient names can be found in our processing code. (bottom) - Processing base Ingredient names by fixing common naming inconsistencies, removing brands, and remapping.

2.3 Food Photo (Data) Image Based Processing

Recipe1M+ provides a second JSON layer containing an ID corresponding to the recipe ID in the ingredient layer (Figure 4), as described earlier. This second object, labeled "images," includes multiple images associated with each recipe ID. Given that some original URLs are now inactive (mostly due to broken links), we chose to download all images directly from the MIT server to ensure we have a complete image set. Only recipes that included images were downloaded, afterwhich, image processing was performed. First, all images were resized to a machine learning-friendly format of 224x224 pixels. Since the images come in various dimensions (e.g., 164x512 or 2456x3423), center cropping was used to maintain the aspect ratio. This involved resizing each image to fit within the 224x224 target dimensions based on its smaller dimension, followed by a center crop to achieve the exact 224x224 size. All images were standardized to RGB format, discarding those in formats like RGBA. To optimize storage, the original 0-255 RGB values were retained rather than normalizing to 0-1, as the latter would consume additional storage space in float notation. Due to the dataset's diverse web-scraping sources, we noticed the Recipe1M+ images already exhibited a wide range of lighting conditions and camera angles. With over 51,000 recipes totaling 718,154 images, and many images already in low resolutions, with occlusions, in various resolutions, and with applied cropping, we decided against additional data augmentation to avoid introducing further noise, which could hinder model learning. Completed image processing photos are given in Figure 5.



```
{
    "id": "e7b4be12ec",
    "images": [
        {
            "id": "56fc46c711.jpg",
            "url": "http://images.media-allrecipes.com/userphotos/250x250/4482570.jpg"
        }
    ]
},
```

Figure 4: Recipe id and Raw Food Image links in JSON format.

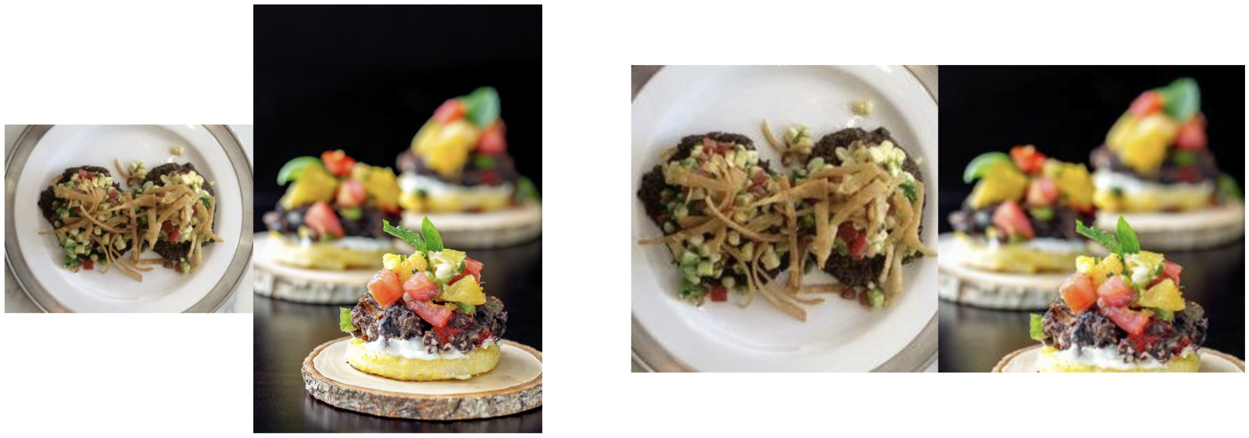


Figure 5: Comparison of Recipe ID: 0a07b1a3d2 before and after cropping.

2.4 Final Cleaned Dataset Structure

Our final data structure is organized into two main folders. The first, image_folder, contains three subfolders for the train, validation, and test sets. The second, labels_folder, also contains corresponding train, validation, and test subfolders. Using the split-folder library, we divided the overall dataset into 70% training, 15% validation, and 15% testing sets. Within the image_folder, each recipe has its own subfolder named after its recipe ID, containing all images associated with that recipe. In the labels_folder, each recipe ID has a corresponding .txt file that lists all ingredients for that recipe. This structure ensures a clear and organized link between each recipe's images and ingredient data.

2.5 Data Processing Statistics

Firstly, by analyzing links in the second JSON layer, we found that the data spans over 78,000 unique websites, with the majority of image URLs originating from major recipe sites. The top 10 websites by image count are listed below in Table 1.

Table 1: Top 10 websites with images extracted.

Website URL	**Number of images extracted from this URL**
https://s-media-cache-ak0.pinimg.com	**155130**
https://i.pinimg.com	**25959**
http://images.media-allrecipes.com	**23404**
http://img.sndimg.com	**22772**
https://i.ytimg.com	**10940**
http://1.bp.blogspot.com	**9243**
http://2.bp.blogspot.com	**9113**
http://3.bp.blogspot.com	**8998**
http://4.bp.blogspot.com	**8910**
https://images-na.ssl-images-amazon.com	**7856**

In our processed and cleaned dataset, we identified a total of 345 unique ingredients across 50,797 recipes with over 700,000 images. On average, each recipe contained 14.14 images and ranged from a minimum of 1 to a maximum of 499. On average, each recipe also contained 5.81 ingredients, with the shortest ingredient list comprising just 1 ingredient for ”Cranberry Juice (ID: 00443b51d0)” and the longest ingredient list featuring 23 ingredients for a recipe titled ”Granola Supreme (ID: fffb3bbff2).” The figures below visually represent the Ingredient Distribution Across Recipes (Figure 6 (left)) and the Number of Ingredients Across Recipes (Figure 6 (right)).

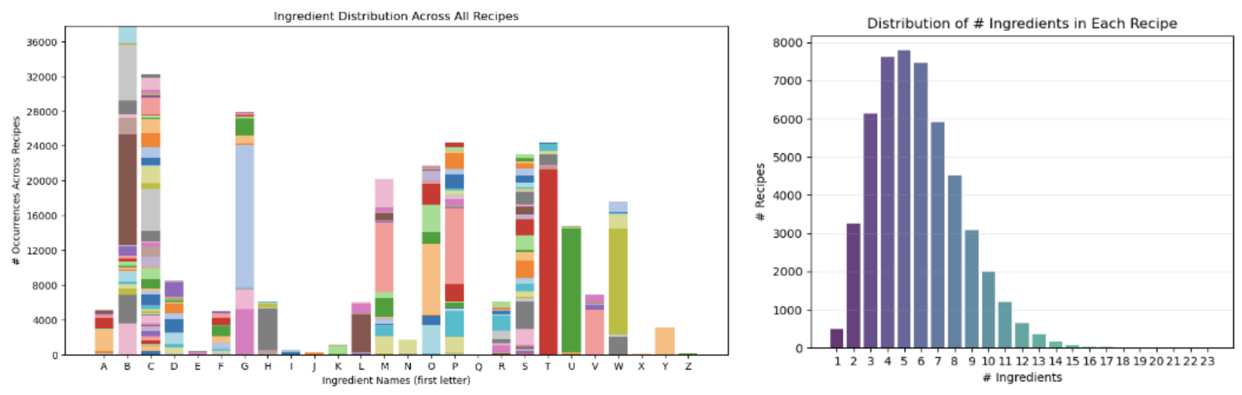


Figure 6: Ingredient Distribution Across Recipes (left) and Number of Ingredients Across Recipes (right)

Using these graphical visuals we were also able to confirm that our data splitting was balanced as a proportional Ingredient Distribution Across Recipes and the Number of Ingredients Across Recipes was seen across the training, validation, and test sets at a 70%, 15%, 15% split respectively.

We also analyzed the most and least common ingredients across recipes as seen in Table 2. As expected, the most common ingredients are cooking essentials like salt, sugar, water, flour, and oil. In contrast, the least common ingredients are rare items that appear only in highly specific recipes.

Table 2: 10 most and least common ingredients found in the dataset.

Top 10 Most Common Ingredients:	**Top 10 Least Common Ingredients:**
• table salt (21385)	• refined sorghum flour (8)
• granulated sugars (16438)	• goose fat (7)
• unsalted butter (14285)	• tapioca puddings (7)
• bottled water (12834)	• goldsaft sugar beet syrup (7)
• wheat flour (12293)	• toffee (6)
• pepper (8878)	• quaker cereals (5)
• olive oil (8230)	• spicy buffalo sauce (4)
• milk (8039)	• liver chicken (4)
• brown sugars (6400)	• indian bean masala smart soup (2)
• garlic powder (5190)	• ancho peppers (1)

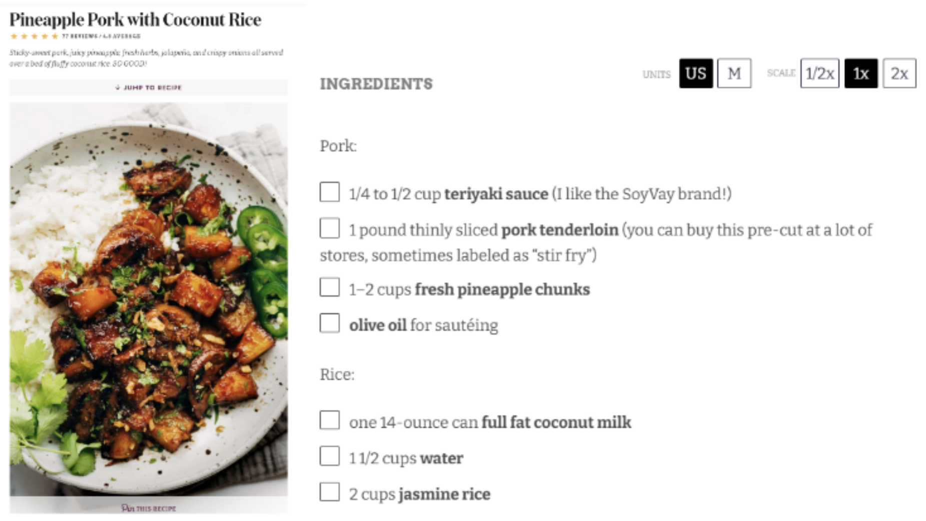
We've considered the potential issues with these rare ingredients, as their infrequent occurrence makes our dataset imbalanced. If they appear solely in validation and testing, the model would have no prior exposure to them during training. In the next section, we address how our model may handle these challenges.

2.6 Plan for Testing on Never Before Seen Data

To handle unfamiliar ingredients, our model will employ two strategies: unknown class handling and incremental learning. When an ingredient is not recognized with high confidence across any existing ingredient classes, the model can classify it as ”unknown,” reducing the risk of misclassification and impacts on our evaluation metrics. Time permitting, we aim to periodically update the model with information about this new data, enabling it to learn about previously not encountered ingredients over time given these new ingredients form a substantial group. This plan for testing on never before seen data prioritizes accuracy while balancing opportunity for adaptability, allowing the model to improve without requiring immediate recognition of new ingredient items.

2.7 Data Collection and Web Scraping for Test Data

As mentioned in our project proposal, we will perform some web scraping to obtain our own secondary test data. Since this test data will not affect our model learning for now, we will create a collection plan and collect data as we progress further. Recipe 1M+ comprises images from 21 well-known recipe websites; thus, we selected an alternative source, Pinch of Yum (Pin), which is not included in this list, for our test data. This site was chosen for its suitable ingredient formatting and broad range of recipes as Pinch of Yum displays core ingredients in bold text, facilitating straightforward extraction through HTML tags (e.g., ”strong”) as shown in Figure 7. Using the Selenium library, we will collect a substantial sample of URLs from the homepage, iteratively accessing each to retrieve image URLs and ingredient text. Our target is to compile around 1,000 recipes for analysis.


Pineapple Pork with Coconut Rice

Sticky-sweet pork, juicy pineapple, fresh herbs, jalapeño, and crispy onions all served over a bed of fluffy coconut rice. SO GOOD!

↓ JUMP TO RECIPE

INGREDIENTS

UNITS US M SCALE 1/2x 1x 2x

Pork:

- 1/4 to 1/2 cup **teriyaki sauce** (I like the SoyVay brand!)
- 1 pound thinly sliced **pork tenderloin** (you can buy this pre-cut at a lot of stores, sometimes labeled as "stir fry")
- 1–2 cups **fresh pineapple chunks**
- **olive oil** for sautéing

Rice:

- one 14-ounce can **full fat coconut milk**
- 1 1/2 cups **water**
- 2 cups **jasmine rice**

Figure 7: Example recipe from Pinch of Yum with bolded ingredients.

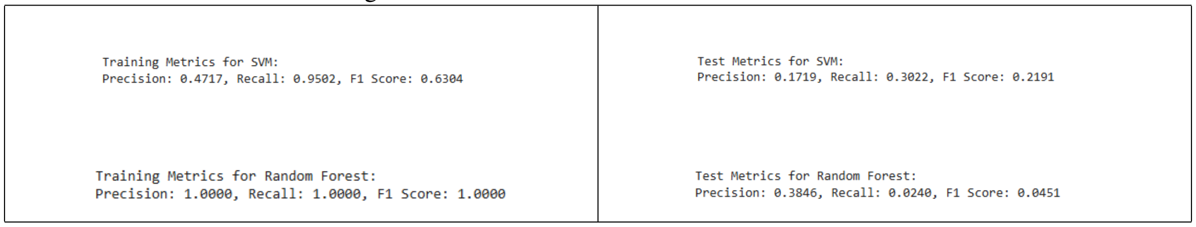
3 Baseline Model

Our approach to clear a multi-label classification problem involves developing a baseline model outlined in Food-101 – Mining Discriminative Components with Random Forests (Bossard and et al., 2014). The model leverages standard classifiers like Support Vector Machine (SVM) and Random Forest to establish initial performance metrics. This allows us to gain insights into the problem before moving to more complex neural network architectures. We will be measuring the F1 score, as well as the accuracy to allow us to compare with the primary model.

3.1 Quantitative Results

As we can see from the below table, the SVM model achieved a net F1 score of 0.63 on the training set and a F1 score of 0.21 on the test set.Meanwhile, the RF model achieved a training F1 score of 1.0 while achieving a low F1 score of 0.045 on the test set.

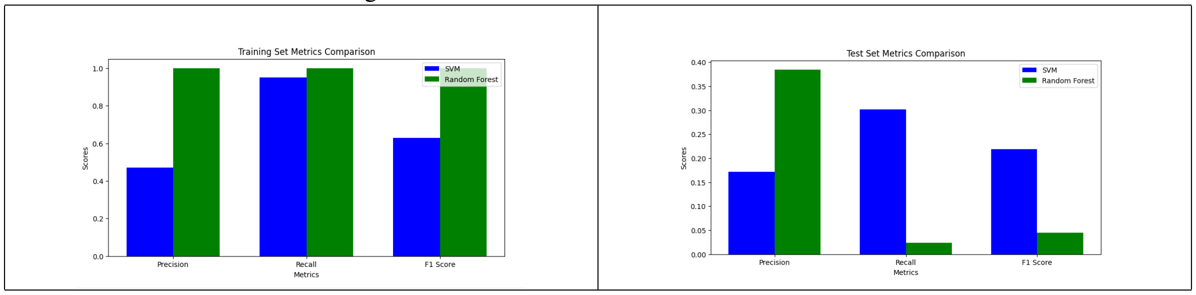
Table 3: Training and test results for both Random Forest and SVM models.


Training Metrics for SVM: Precision: 0.4717, Recall: 0.9502, F1 Score: 0.6304	Test Metrics for SVM: Precision: 0.1719, Recall: 0.3022, F1 Score: 0.2191
Training Metrics for Random Forest: Precision: 1.0000, Recall: 1.0000, F1 Score: 1.0000	Test Metrics for Random Forest: Precision: 0.3846, Recall: 0.0240, F1 Score: 0.0451

3.2 Qualitative Results

We can also see a graphical representation of these test results, showing the difference in precision, accuracy and F1 scores for Random Forest and SVM models.

Table 4: Training and test results for both Random Forest and SVM models.



From these graphs, we can clearly understand that the Random Forest model is over-fitting. We understand this from the fact that its performance is perfect in the training phase, while not giving any good results in the test phase. This is a clear indication that the model is memorizing and not learning, and hence cannot generalize well.

3.3 Challenges Faced

During the SVM training process, we encountered several data-related challenges, including issues with data preprocessing and data imbalance. Many labels lacked sufficient samples (e.g. least common ingredients in Table 2), complicating classification. Additionally, the training dataset size needed to be reduced to compensate for the extensive time required to preprocess input images into superpixels. We also faced overfitting with the Random Forest model, which achieved near-perfect accuracy on the training set but performed poorly on validation and test sets due to its inability to generalize beyond the training data.

3.4 Iterations and Refinements

To address the challenges we faced and enhance our baseline model, we iteratively refined our approach. With a large dataset of 70,000 images, we decided to reduce the dataset size to address the long training times as preprocessing was beyond our available compute power. By sampling only 10% of the original data, training became more manageable and computational costs were reduced. While this reduction expedited model training, it also posed challenges in terms of model performance and data representation. Additionally, we removed the Random Forest algorithm due to severe overfitting, despite various tuning attempts. This shift allowed us to concentrate on more generalizable models like SVM, simplifying our baseline model and ensuring a straightforward benchmark that was easier to train and implement.

4 Primary Model

4.1 Current Model Architecture

Our current model architecture for ingredient detection combines a ResNet-50 backbone for feature extraction with a GRU layer for sequential modeling. This setup is designed to capture essential features from input images while modeling the temporal dependencies between them, which is crucial for accurate multi-label classification in ingredient detection. The CNN Backbone consists of ResNet-50, which extracts high-dimensional feature embeddings from each image. These embeddings provide robust representations that are critical for identifying ingredients effectively. Following feature extraction, the model employs a GRU layer to process these embeddings and model sequential dependencies across image regions, enhancing the model's ability to understand relationships between various features. Finally, a fully connected layer maps the RNN output to the ingredient prediction space. For this multi-label classification task, we use Binary Cross-Entropy with Logits Loss, which allows the model to independently predict the presence or absence of each ingredient.

4.2 Current Model Performance and Limitations

The ResNet-GRU model shows some promising results but also highlights areas for improvement. During training, the loss stabilizes early, suggesting potential underfitting or limited learning capacity for complex representations. Meanwhile, the validation loss fluctuates across epochs, indicating possible instability in generalization to new data. Additionally, evaluation metrics reveal that the F1 score consistently falls below 0.5, with low precision and recall scores, indicating that the model has difficulty accurately identifying multiple ingredients. These performance results suggest that the current CNN-RNN setup may not fully capture the intricate dependencies required for effective multi-label ingredient prediction.

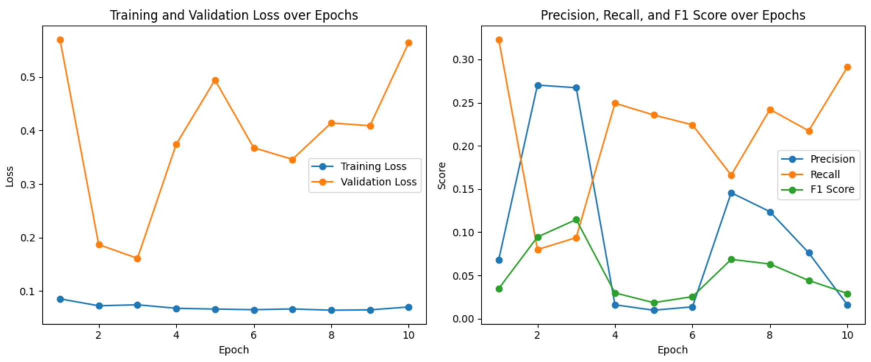


Figure 8: Left: Training and Validation Loss over 10 Epochs using 10% of Dataset. Right: Precision, Recall, and F1 Score over 10 Epochs using 10% of Dataset.

4.3 Proposed Improvement: Transformer Integration

To address the limitations observed with the current model, we propose integrating a transformer layer in place of the GRU for sequential modeling. Transformers offer a self-attention mechanism that enables the model to capture complex feature interactions and dependencies more effectively than traditional RNNs. By implementing self-attention, the model can focus on specific regions within the image, which is crucial for multi-label tasks where ingredients may be spread across distinct parts of the image. We plan to replace the GRU with a transformer encoder, which processes all features in parallel and can better capture the relationships between different ingredients.

Furthermore, we will introduce positional encoding to provide spatial information to the transformer, as transformers do not inherently handle sequence order. This encoding will help the model understand the arrangement of features in the image, which is critical for accurate ingredient detection. Overall, the transformer-based approach is expected to enhance contextual understanding, focus attention on relevant image regions, and improve model scalability.

4.4 Comparison to baseline and Feasibility

Our baseline models, SVM and Random Forest, currently outperform the CNN-RNN architecture, showing that traditional models handle simpler feature sets effectively at this preliminary stage. However, these models struggle with complex feature extraction and sequential dependencies, which limits their scalability.

In contrast, the CNN-RNN model, despite limited training, shows strong potential. With only 10% of the data and 10 epochs used in initial tests, we expect that full training, additional layers, and architectural updates like transformer layers would enable it to capture deeper patterns and interactions that traditional models cannot. Iterative improvements in architecture and hyperparameter tuning should ultimately allow the CNN-RNN to exceed baseline performance in both accuracy and generalization.

4.5 Next Steps

To address current limitations, we will refine the dataset for better balance, possibly by removing recipes with ingredients that appear less than 1,000 times, reducing sparsity and focusing on more common ingredients. We also plan to extend training to the full dataset, add layers, and further optimize the model given our limited GPU resources. These enhancements will help the CNN-RNN achieve meaningful gains, positioning it to outperform the baseline models.

5 Individual Contributions and Responsibilities

Over the course of the project, our team has worked together cohesively, guided by our team alignment charter and the communications methods we established. We used Discord extensively for discussions, asking questions, reacting to messages, providing updates, and document sharing, Google Drive for file storage and documentation, providing both an organized and central repository for our Colab workspace and written deliverables, as well as GitHub for maintaining version control on our code. These platforms facilitated good transparency, ensuring everyone was aware of any ongoing progress and could contribute and collaborate easily and we intend to continue using them over the remainder of our project.

In terms of progress, our group has successfully achieved several key milestones at this interim stage including completing processing of the majority of our data (for training, validation and testing purposes), as well as developing a reasonable baseline model and our deep model and producing some initial qualitative and quantitative results. Although our team experienced differing periods of high workload, by maintaining consistent weekly team meetings, working independently, then regrouping to provide updates and plan, everyone stayed informed and aligned on our task plan. We were also able to make adjustments as necessary to our internal due dates, for example, pushing our deadline for "Complete Full Dataset Cleaning" by a few days due to midterms, as we had planned for enough buffer time between tasks, which is something we will continue to plan for. Overall, our task division has been smooth, with everyone getting to develop their interests, learn and implement new models, and seeking help when needed. With our updated project plan detailed below, and our insights gained from this checkpoint stage, we feel our progress is on track to complete the project within the available time frame.

Table 5: Team Charter

Communication Quality	• **Everyone** was highly responsive in group discussions, answering questions, and reacting to messages to confirm reading. • **Anipreet** reached out to TAs and professors for clarification as needed and provided document comments. • **Arthur** read relevant papers and shared insights with the group. • **Felicia** consistently booked meeting rooms and project managed group meetings. • **Sid** monitored Piazza for updates and kept the group informed on messages.
Completed Accomplishments	• **Everyone** : Collaborated on dataset selection, cleaning processes, baseline and deep model choices, and data formatting for model input. • **Anipreet** : Developed baseline SVM model and evaluation metrics. • **Arthur** : Processed food dish images by cropping and normalizing data. • **Felicia** : Organized ingredient label data, standardizing names and categories. • **Sid** : Built the deep model using a CNN for dish images and a GRU for ingredient lists, and looking into using transformers.
Final Deliverable (Nov 29, 2024) Internal Deadline Nov 27th	• **Everyone** : Complete technical development tasks and finalize the deep learning model (Nov 18th); collaborate on revising drafts (Nov 24th). • **Anipreet** : Finish baseline model training and results (Nov 10th); assist in improving the deep model's computational efficiency, and write sections on the introduction, baseline model, and ethical considerations (Nov 25th). • **Arthur** : Develop web scraping for new test data; make improvements to training datasets (Nov 15th); write sections on background and related work, and evaluate the model on new data (Nov 25th). • **Felicia** : Assist in web scraping and cleaning new test data; make improvements to training datasets (Nov 15th); write sections on data processing, architecture, and discussion (Nov 25th). • **Sid** : Complete deep model training and results (Nov 10th); enhance the model's computational efficiency, and write sections on quantitative and qualitative results, project difficulty, and manage LaTeX conversion (Nov 25th).
Project Presentation (Nov 29, 2024) : Internal Deadline - Nov 25th	• **Everyone** : Brainstorm the flow of the demonstration video, create drafts iteratively, provide feedback for improvement, and finalize video submission (Nov 25th). • **Anipreet** : Record voiceover and edit the video for clarity and engagement (Nov 25th). • **Arthur** : Set up the framework for the video storyboard (Nov 4th). • **Felicia** : Develop the content for the storyboard to ensure coherence (Nov 11th). • **Sid** : Handle the recording of the video components (Nov 18th).

6 Links to Collab Notebook and Github page

Link to collab notebook
Link to GitHub
Link to Baseline Model

References

Epicurious – Recipes, Menu Ideas, Videos & Cooking Tips | Epicurious. URL https://www.epicurious.com/.

Food ingredients and recipes dataset with images. https://www.kaggle.com/datasets/pes12017000148/food-ingredients-and-recipe-dataset-with-images. Accessed: 2024-10-02.

Pinch of Yum. URL https://pinchofyum.com/.

Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images - mit. https://im2recipe.csail.mit.edu/. Accessed: 2024-10-02.

L Bossard and et al. Food-101 – mining discriminative components with random forests. In *Computer Vision – ECCV 2014*, pages 446–461, Cham, 2014.