Ingredient Detection and Recipe Recommendation System

Project Report

COMS 4995: Applied Computer Vision

Professor Austin Reiter

Kevin Li (kl3285)

Anuva Banwasi (ab5084)

Shivansh Srivastava (ss5945)

Adele Sinai (ams2559)

May 10, 2024

Table of Contents

Links to GitHub Repository and Datasets

Link to GitHub repository:

- https://github.com/kevinli7377/AppliedCVFinalProject

Link to project datasets on Google Drive:

- Extended-fruits-360 Dataset : https://drive.google.com/drive/folders/1MPE5mmMRz5GdFZicR0Ep4Cg9CF24c3yx?usp=sharing
- Test Batch 1: https://drive.google.com/drive/folders/1VqU8GrjYOhxsaUCuLiXNZh6iTUnG5mRA?usp=share_link
- Test Batch 2: https://drive.google.com/drive/folders/1meIHyuWd2kGOChNONTshH6FeOW9HFitU?usp=share_link

Links to original datasets:

- Vanilla fruits-360 Kaggle Dataset: https://www.kaggle.com/datasets/moltean/fruits
- Vanilla Epicurious Recipe Dataset: https://www.kaggle.com/datasets/pes12017000148/food-ingredients-and-recipe-dataset-with-images

Introduction

Project Background and Motivation

Many individuals can find it challenging to decide what to cook using the ingredients they already have at home. While online recipe databases are abundant, they lack context-awareness and cannot suggest recipes based on the real-time availability of ingredients. In response to this limitation, our project utilizes image models that segment and identify ingredients to suggest suitable recipes. We tested two different model architectures on a range of food item classes.

Purpose and Expected Outcomes

Our solution allows users to upload an image of ingredients on a counter/fridge shelf. The program is designed to:

- **Segment Ingredients:** Segment each ingredient visible in the image.
- **Identify Ingredients:** Accurately identify each segmented ingredient with predetermined labels.
- **Recipe Recommendation:** Based on the identified ingredients, the system outputs feasible recipes.

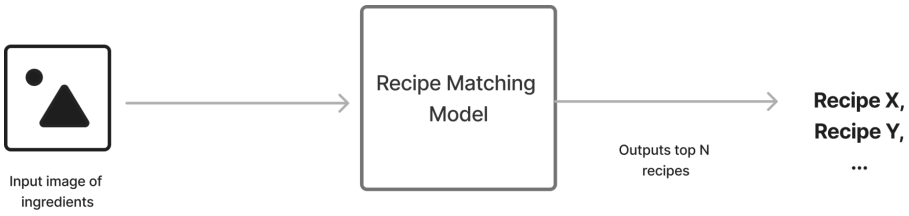


Figure 1: Diagram showing model input and output

Stakeholders

This system is designed with several key stakeholders in mind:

- **Consumers:** who seek convenient and efficient ways to utilize their existing groceries.
- **Curated Recipe Platforms:** which can integrate this technology to offer personalized recipe suggestions to their users
- **Supermarkets:** which could apply this technology for innovative solutions such as automated checkout processes and shopping recommendations.

Literature Review

We will briefly examine a few existing works on food segmentation and classification, as well as an overview of the models and algorithms we considered when building our detection and recipe matching procedures.

Delving into food segmentation, Ziyi Zhu and Ying Dai present a framework to segment ingredients utilizing feature maps of the CNN-based Single-Ingredient Classification Model trained on the dataset with image-level annotation [14]. This alternative annotation proves to be more manageable and less time consuming than annotating each image at the pixel level, which the authors point out as one of the motivations for this solution. To train their model, Zhu and Dai constructed a single-ingredient image dataset based on a standardized biological-based hierarchical ingredient structure. Then, single-ingredient classification model on this dataset, extracting feature maps for each image.

In terms of recipe recommendation models, Rodrigues, Fidalgo and Oliveira propose a recipe recommendation system in order to reduce food waste by utilizing ingredients people already have at home [10]. Their framework implements a web application that performs image recognition of food ingredients, and then produces a recipe recommendation. The user inputs an image of an ingredient and the app recommends recipes containing the recognized ingredient. They used ResNet-50 to perform image recognition and trained the model with a dataset that contains about 36 classes of vegetables and fruits. Through this training, the model reached 96% accuracy in classifying the dataset images.

To implement the recipe matching portion of our own model, we also referred to an article written by Jack Leitch, published on *Medium,* about using the Word2Vec and SciKit-Learn models for recipe matching [4]. The procedure outlined was to parse ingredients for each recipe by simplifying ingredient names, and encoding each recipe ingredient list using TF-IDF. Then, the Cosine Similarity function was applied to find similarity between ingredients for known recipes and input ingredients given by the user. The top recommended recipes are given according to the end score. Word2Vec was prioritized because it gives a distributional similarity between words, meaning it identifies which words tend to appear together in the same context. In this application, this identifies ingredients commonly used together. Leitch used the Continuous-Bag-of-Words (CBOW) version of Word2Vec.

For the classification component of our pipeline, we considered a few models in our preliminary research. In their paper “Deep Residual Learning for Image Recognition”, Kaiming He et. al. present a residual learning framework which aims to offer a solution to training complex deep networks [2]. More specifically, they reformulate the neural layers as learning residual functions. On their ImageNet dataset, they evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. Another classification model we researched were DeiT - or Data Efficient Image Transformer, and ViT - Vision Transformer. Dosovitskiy et. al. apply Transformer architecture without using CNNs [1]. As stated, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. Their proposed framework demonstrates that a reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks.

Similarly, we considered a few models for the segmentation component of our framework. The main models we looked at were SAM and GroundingDino, as well as Grounded SAM which is a combination of

the two. In their project “Segment Anything Model” (SAM), Kirillov et. al. provide a new model and dataset for image segmentation, and using their model were able to build a dataset with over 1 billion masks on 11M licensed and privacy respecting images [3]. The model is designed and trained to be promptable, so it can transfer zero-shot to new image distributions and tasks. Liu et. al. also present an open-set object detector, called Grounding DINO in their paper [6]. They combine Transformer-based detector DINO with grounded pre-training, which can detect arbitrary objects with human inputs such as category names or referring expressions. The key solution of open-set object detection is introducing language to a closed-set detector for open-set concept generalization. We were able to find a project on Github “Grounded-Segment-Anything” by Xiaoming Fang that combines both models, whose code repo is publicly available.

Team Contribution Breakdown

Team Member	Contribution
Kevin Li (kl3285)	● Classification model dataset (Extended-fruits-360) acquisition, curation, and processing ● Ingredient dataset parsing and embedding generation ● Implementation of matching algorithm with sentence transformers ● Final combined architecture testing
Anuva Banwasi (ab5084)	● Research on datasets for fruits and vegetables classification ● Fine-tuning CLIP model for detection of individual ingredients ● Implementation of segmentation (SAM and GroundingSAM) to extract masks of individual ingredients from images ● Designing additional test dataset combining ingredients from test images of Fruits-360
Shivansh Srivastava (ss5945)	● Preliminary research for object detection candidates ● Designing and conducting detection experiments ● Fine tuning and evaluating final classification model ● Combining chosen classification model with segmentation
Adele Sinai (ams2559)	● Research for training datasets for meat and dairy classification ● Designing and creating dataset for Test Batch 1 web scraping live scenes, augmenting scenes ● Enhancing segmentation and classification by iteratively debugging and refining test data ● Designing and creating dataset for Test Batch 2 augmenting scenes from validation set of Fruits-360

Methodology

Data Collection

Ingredients Dataset

Our image models were trained on an extended Kaggle Fruits-360 dataset, containing a total of 92,182 images and 143 classes. A list of all classes can be found in Appendix A. All images are stored on Google Drive and .json files for the training and validation datasets were produced for dataset loading.

The original Fruits-360 dataset contains 90,380 images of 131 types of fruits and vegetables. Each image contains one fruit or vegetable. The dataset is also pre-split into training and test sets.

For our system to work well with a wide range of scenarios, we believed that it would be necessary to consider other food items, not limited to fresh produce. However, we could not find good datasets of other common food items, including meats, dairy, and common pantry items. Other datasets including Freiburg Groceries contain multiple food instances in a single image, which would not be ideal for training.

To address this, we curated our own dataset of ingredients by extending the original Fruits-360 dataset with 12 additional common ingredient types, including beef, chicken, and eggs. Images for these additional classes were gathered from Google Images using a web scraper and downloaded from online grocery websites (i.e., WholeFoods and FreshDirect). All scraped images were filtered manually. Images containing watermarks, text, and non-whitebackground were edited or removed.

Recipes Dataset

Our system uses the ‘Food Ingredients and Recipes Dataset with Images’ available on Kaggle. The dataset contains a csv file and a folder of images for each recipe. A total of 13,582 recipes are available in the csv file. The ingredients list for each recipe were parsed and embedded. The results were stored as additional columns in an updated csv file. More details can be found in the ‘Model Development’ section below.

	Unnamed: 0	Title	Ingredients	Instructions	Image_Name
0	0	Miso-Butter Roast Chicken With Acorn Squash Pa...	['1 (3½–4-lb.) whole chicken', '2¾ tsp. kosher...	Pat chicken dry with paper towels, season all ...	miso-butter-roast-chicken-acorn-squash-panzanella
1	1	Crispy Salt and Pepper Potatoes	['2 large egg whites', '1 pound new potatoes (...	Preheat oven to 400°F and line a rimmed baking...	crispy-salt-and-pepper-potatoes-dan-kluger
2	2	Thanksgiving Mac and Cheese	['1 cup evaporated milk', '1 cup whole milk', ...	Place a rack in middle of oven; preheat to 400...	thanksgiving-mac-and-cheese-erick-williams
3	3	Italian Sausage and Bread Stuffing	['1 (¾- to 1-pound) round Italian loaf, cut in...	Preheat oven to 350°F with rack in middle. Gen...	italian-sausage-and-bread-stuffing-240559
4	4	Newton's Law	['1 teaspoon dark brown sugar', '1 teaspoon ho...	Stir together brown sugar and hot water in a c...	newtons-law-apple-bourbon-cocktail
...	...	...	...	...	...

Figure 2: Original csv file printed as a Pandas dataframe.

Scene Dataset

In order to test our final model architecture (i.e., combining segmentation, classification, and matching), we created two scene datasets, Test Batch 1 and Test Batch 2, which served as our two test batches. We have a total of 93 images in Test Batch 1 and 40 images in Test Batch 2.

In **Test Batch 1**, we prioritized creating scenes that contained “natural” images of ingredients. These consist of “live” scenes and “augmented” scenes. To assemble the “live” scene, the test scenes contain ingredients for a certain recipe placed together on a counter. Some of the scenes were also sourced from Google Images, social media sites, or other recipe websites. Due to the physical limitation of having all of the necessary ingredients for a certain recipe available in person or in one scene found online, we created “augmented” scenes to supplement the test dataset. For the augmented scenes, we found images of ingredients via Google Images and spliced them together via Google Doc to place them all on one color background to simulate a one-toned counter top. With the augmented scenes, we also prioritized each scene being as realistic as possible - with real pictures of ingredients. We experimented with using DALL-E to create realistic scenes of ingredients assembled on a countertop. However, we ended up using a total of 5 DALL-E produced images so as to only include realistic images.

In **Test Batch 2**, we created augmented test scenes by splicing individual images of ingredients from the validation portion of the Fruits-360 Kaggle dataset.

An issue we ran into while running the model on Test Batch 1 was that the model had trouble segmenting and detecting each individual ingredient in the test scenes. In analyzing the results with Test Batch 1, we acknowledge that there is a high variance in how each ingredient can appear due to differing shapes, coloring, and shadows of naturally occurring raw grocery ingredients. As such, we wanted to determine whether the model would have more success with detecting and classifying ingredients of a standardized shape and color per item class.

Since the model was not trained on images of ingredients from the validation set, we believe it is not a conflict to then create augmented test scenes from these individual ingredient images. We adhered to our earlier standard for augmented scenes in Test Batch 1, where the images of each ingredient would be spliced and assembled together onto a one-toned background using Google Docs or the Notability app.

Within each of the test batch datasets, each entry is a .jpeg file of the scene image, titled as the [ID-Number-of-Recipe].jpeg of the recipe entry it corresponds to in the Recipe Dataset.

Model Development

Model Architecture

Our approach consists of three main phases: 1) segmentation 2) classification and 3) recipe matching. Starting with an initial image consisting of many ingredients, we apply a segmentation model to extract masks of the individual ingredients. Then, we apply a fine-tuned ingredient classifier to determine each individual ingredient. We concatenate all of the detected ingredients into sentences and embed these with a sentence transformer. Finally, we utilize a matching algorithm to match the predicted ingredients to the most relevant recipes from the recipe database.

We considered many other potential model architectures before deciding on the above. For instance, one of our initial plans was to fine-tune CLIP to perform detection of multiple ingredients in a single image. However, we discovered that CLIP often was unable to recognize and often missed several ingredients in an image. Furthermore, CLIP is already an out of the box and should not necessitate fine-tuning. Ultimately, we decided to have a separate segmentation model and classification model. This ensures that we first collect the individual segments for each ingredient and then can run these through the classification model. For the classification phase, we fine-tuned and tested several classification models against each other and found that the DeiT model had the highest accuracy for ingredient classification.

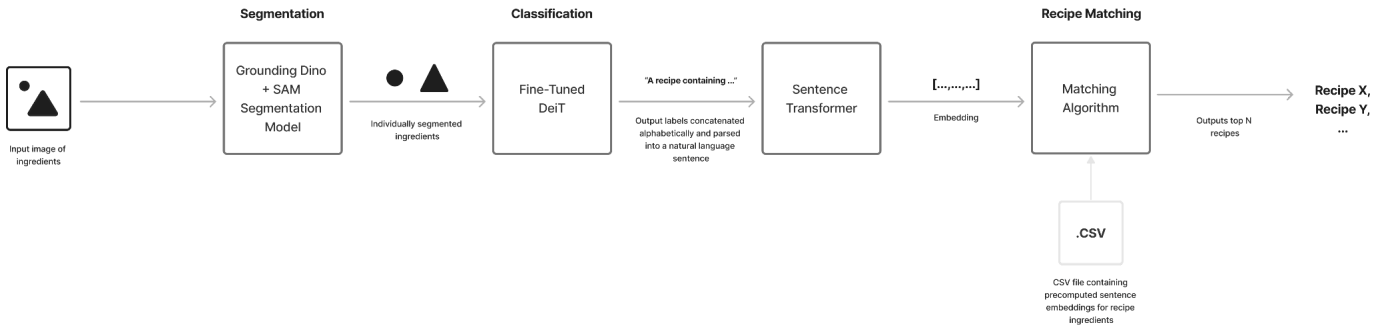


Figure 3: Final Architecture: Segmentation, Classification, Matching Pipeline

Segmentation Model

The goal of the segmentation model is to take an input image with many ingredients and output masks of each individual ingredient. To do this, we started with Segment Anything (SAM) from Meta. We implemented the model to extract masks of individual ingredients from an input image. Upon running the model on testing data, we discovered several weaknesses. First of all, we found that Vanilla SAM tends to segment the individual ingredients and create additional masks that we do not need. For instance, running the model on a jar of tomato sauce and it produces masks for the lid, bottle, etc. when we just want a single mask of the jar. Similarly, for ingredients inside bowls, the model outputs multiple masks of the bowl with ingredients as well as just the ingredients in the bowl themselves. Furthermore, we also discovered that the model tends to generate other unwanted masks like that of the background (ie. countertop, table, etc.). In order to address some of these issues, we sorted the masks by area and took the top 20-30 masks outputted from the model. However, we found that there were still too many instances of unnecessary masks that would most likely negatively impact the classification and matching results.

We began exploring alternative segmentation models capable of processing not just images but also textual or prompting inputs. Our search led us to Grounded Segment Anything (Grounded-SAM), a fusion of Grounding Dino and SAM. This zero-shot model is capable of being prompted with both text and images. We applied prompt engineering, experimenting with various prompting techniques for the model. Initially, we attempted to provide the model with a comprehensive list of ingredients from our dataset and instructed it to segment any ingredients present in the image. However, we observed that this approach caused the model to search for ingredients even when they were absent from the image. Ultimately, we determined that the most effective prompting input was to instruct the model to generate masks for all food items present in the image. This prompt struck a balance between generality and specificity,

guaranteeing that the model identifies masks for every ingredient visible. This approach addressed several challenges encountered with Vanilla SAM, such as preventing the background from being included as a mask as well as preventing duplicate masks.

Training Classification Model

All candidates for ingredient classification were fine-tuned on the original Fruits-360 dataset from Kaggle, which contained 67,692 training samples and 22,688 samples across 131 classes (Refer to Table 1 in Appendix A for the complete list of classes in this dataset). To ensure fair comparisons, each model was also fine-tuned with the same set of hyperparameters: a learning rate of 0.001 with no weight decay, an Adam optimizer with a momentum of 0.9, and a batch size of 64. Only the final classification layers for each model were changed to include an intermediary linear layer with ReLU activation, a dropout layer with a probability of 0.25 to aid in generalization, and another linear layer whose output size was equal to the number of classes.

Classification Model Experimentation

Several base models were considered for the ingredient classifier, including ResNet18, VGG16, ViT, and DeiT. Both pairs of CNNs and image transformers were previously trained on the ImageNet dataset, making them particularly useful for fine-tuning on this task. The ideal candidate would output strong accuracies and losses on both the training and validation sets throughout the experiment. This would be indicative of stable training and generalizability on unseen data.

Matching Algorithm

The final matching algorithm adopts a hybrid approach by utilizing the following techniques:

1. **Cosine similarity:** Perform cosine similarity on the embedding created for the query recipe sentence embedding and all dataset recipe sentence embeddings. This part of the algorithm computes scores of [0,1].
2. **Keyword matching:** Perform a brute force method on the top results returned from the initial cosine similarity test. Compute the total number of matches in ingredients (i.e, words) between query and recipe ingredients.

The final N results returned by the hybrid algorithm are ordered by averaging the sum of the cosine similarity score and number of matches with the query.

Note that additional testing and research was performed to find the best matching algorithm for the system, as mentioned in the literature review. While there are existing solutions for our problem, they utilize Word2Vec instead of Sentence Transformers. Word2Vec produces context-independent embeddings while sentence transformers capture contextual information and produce different embeddings for the same word when used in different contexts. We tested existing Word2Vec solutions that also utilize csv files containing different recipes. Unfortunately, they did not perform well, especially for our purposes as we are trying to develop a system that can match queries with recipes even when the query is missing a few ingredients. Ultimately, we decided to use sentence transformers to embed alphabetically ordered ingredients concatenated as single sentences (e.g., ‘A recipe containing apples, bananas, …’). We noticed that this provided the best results and could be used to narrow down the number of recipes to perform manual matching.

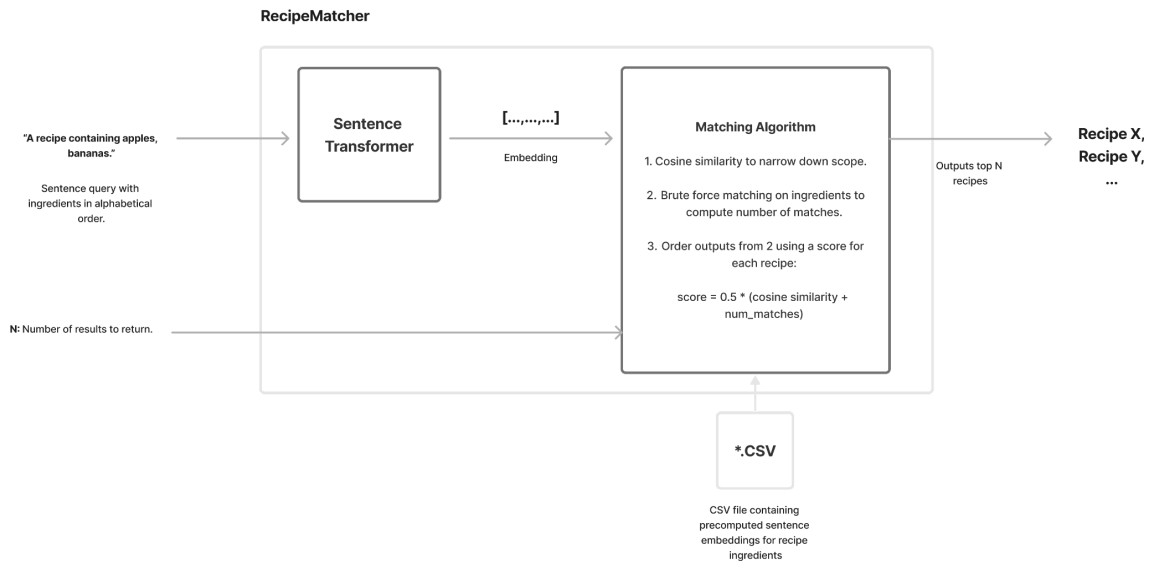


Figure 4: Matching Algorithm Architecture Diagram.

All ingredients lists in the original csv were parsed to remove as much unnecessary text (i.e., measurements and descriptive text) as possible. Regex and NTLK were used to create the parser. The updated csv file contains additional columns for the cleaned ingredients list, ingredient sentence, and the sentence embedding.

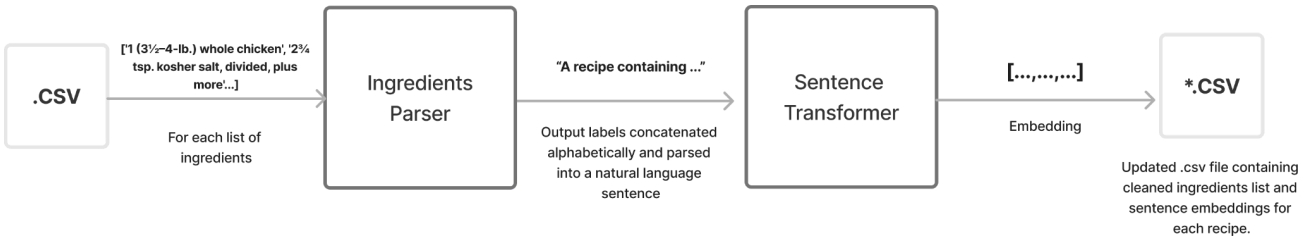


Figure 5: Recipe dataset preparation.

System Integration

Each step in the pipeline was managed manually, from input through processing to output. Segmentation outputs for each scene were saved as individual images in dedicated folders on Google Cloud Platform (GCP) using Jupyter notebook. These images were then manually processed with the fine-tuned DeiT model to generate labels for each scene. The outputs were stored as text files, which were later parsed and fed into the RecipeMatcher to retrieve the top N recipe suggestions for each scene image.

Testing and Validation

Classification Model

Using the hyperparameters outlined earlier, all four classification candidates were trained for 3 epochs on the base Fruits-360's training set. After each epoch, each model's accuracy and loss on the base Fruits-360's validation set were recorded in order to determine the best weights—whenever a lower validation loss was seen, the experiment pipeline saved the model's current state dictionary. At the end of training, each model's best state dictionary was loaded so that the best model could be evaluated on the validation set one last time.

The same methodology was also used for the final DeiT model on the extended Fruits-360 dataset.

Combined Model

The final architecture (i.e., combining segmentation, classification, and matching) was evaluated based on the following metric:

$$Accuracy = \frac{S_c}{S_t}$$

$S_c = Number\ of\ scenes\ with\ correctly\ matched\ recipes$

$S_t = Total\ number\ of\ scenes$

The criterion for a correctly matched recipe: A recipe is considered correctly matched if the pre-labeled recipe for the scene is within the top *N* returned recipes.

All scene images were segmented using the segmentation model and subsequently classified using the fine-tuned DeiT model. Finally, the generated list labels were processed by turning the list into a set to remove duplicates and passed into the RecipeMatcher.

Results

The following section presents the performance for 1) Classification model and 2) the final combined model architecture (i.e., combining segmentation, classification, and matching).

Classification Model Performance

While all four fine-tuned models yielded promising results, DeiT was ultimately chosen to move forward for classification on the expanded dataset. Based on the accuracy and loss curves on the validation set during training, the DeiT model was a clear front-runner. Although all candidates boasted relatively high accuracies on the validation set after training, the DeiT's data efficient architecture and reliance on the attention mechanism facilitated more stable training. As a result, the DeiT model had an easier time extracting key features from each ingredient, which led to improved generalization on the validation set.

A fresh instance of the DeiT model was then trained on the extended Fruits-360 dataset, which featured 12 new classes (Refer to Table 2 in Appendix A for the complete list of additional classes in this extended dataset). This extended dataset contained 69,167 training samples and 23,015 validation samples across 143 classes. The same set of hyperparameters were used. (Refer to Table 3 in Appendix for sample classifications on a random subset of 50 samples from the validation set using the best model).

Base Fruits-360 ResNet18 Training Metrics

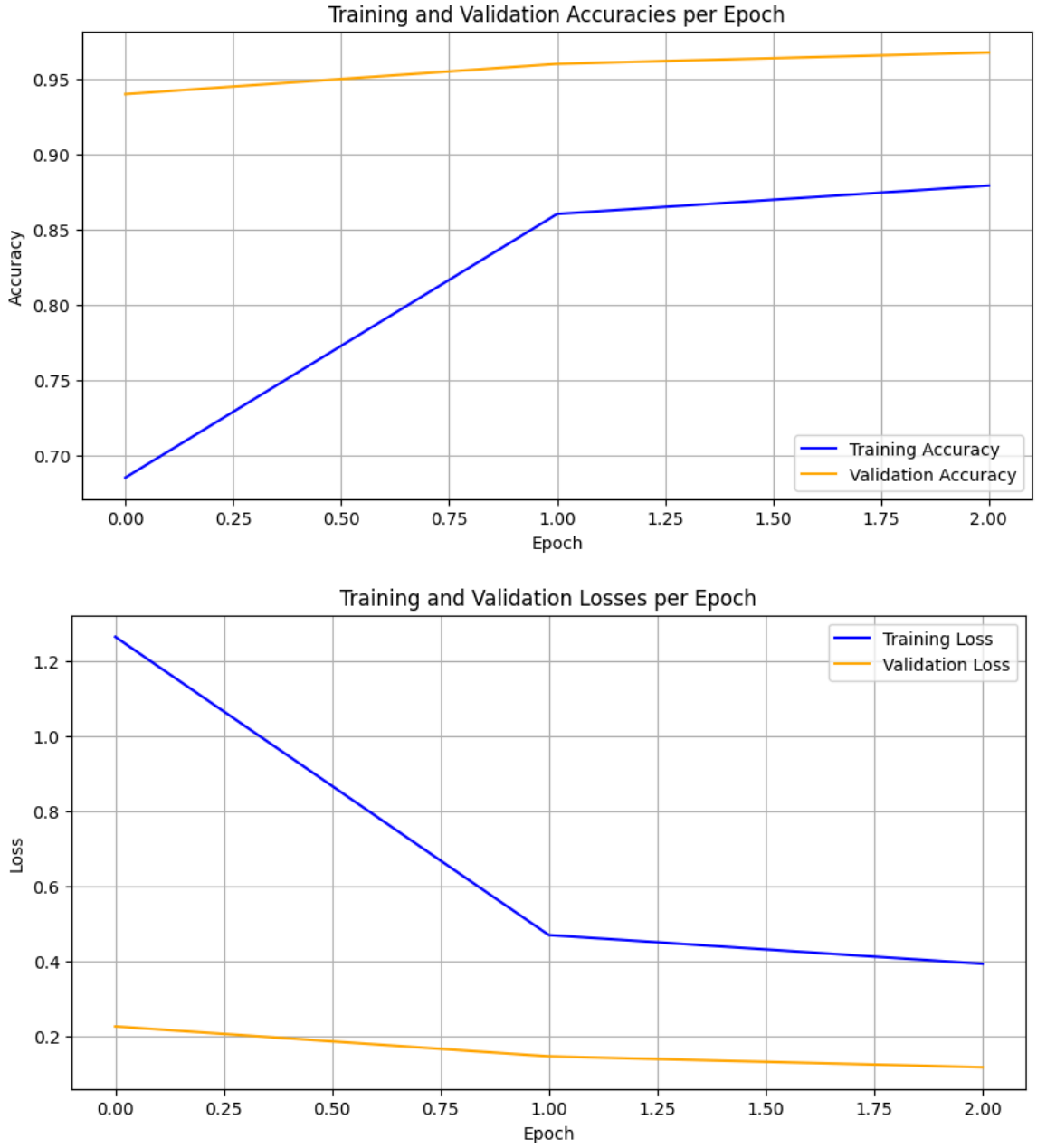


Figure 6: Training and evaluation accuracies (top) and losses (bottom) per epoch for base fruits-360 on Resnet18

Base Fruits-360 VGG16 Training Metrics

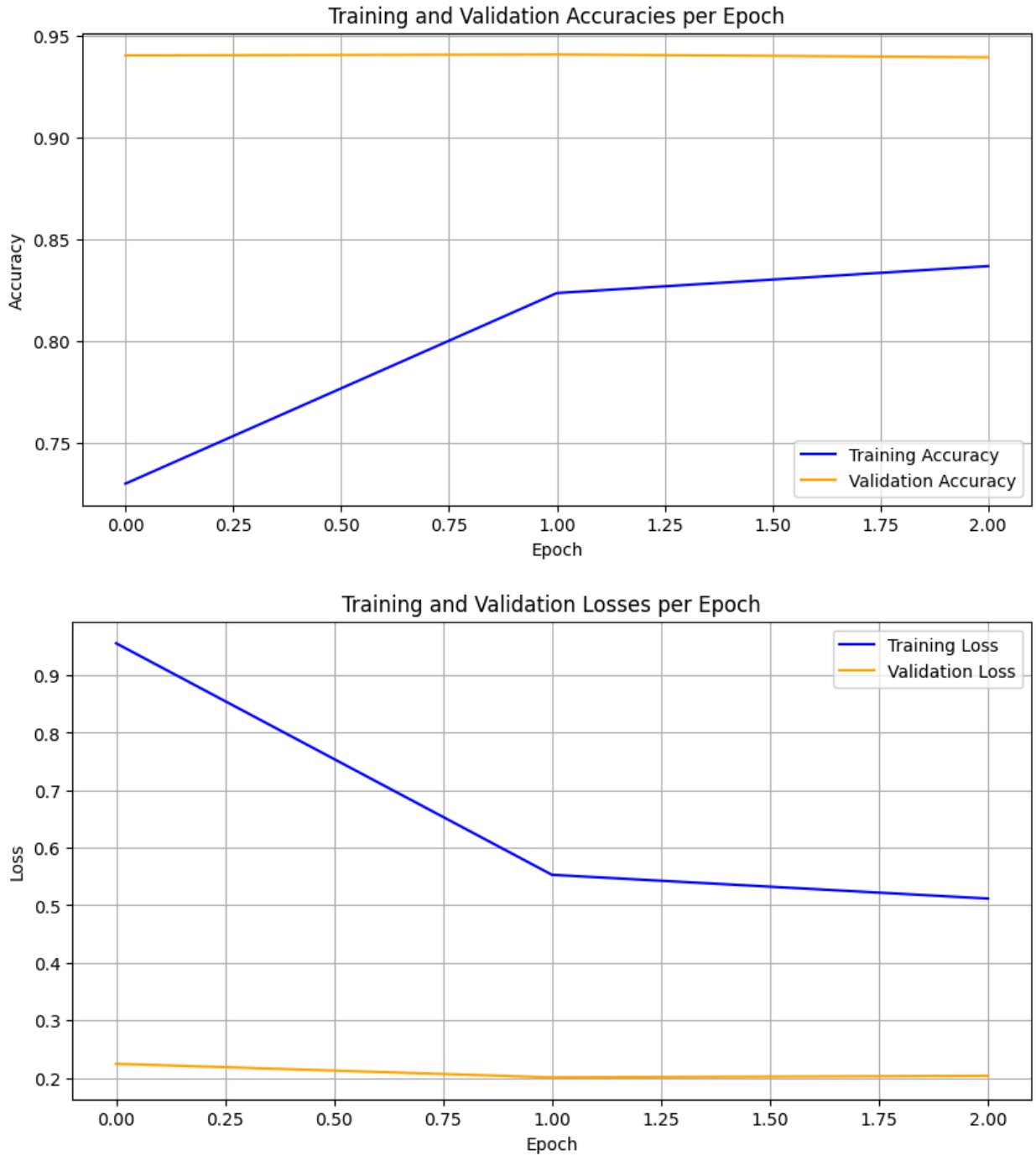


Figure 7: Training and evaluation accuracies (top) and losses (bottom) per epoch for base fruits-360 on VGG16.

Base Fruits-360 ViT Training Metrics

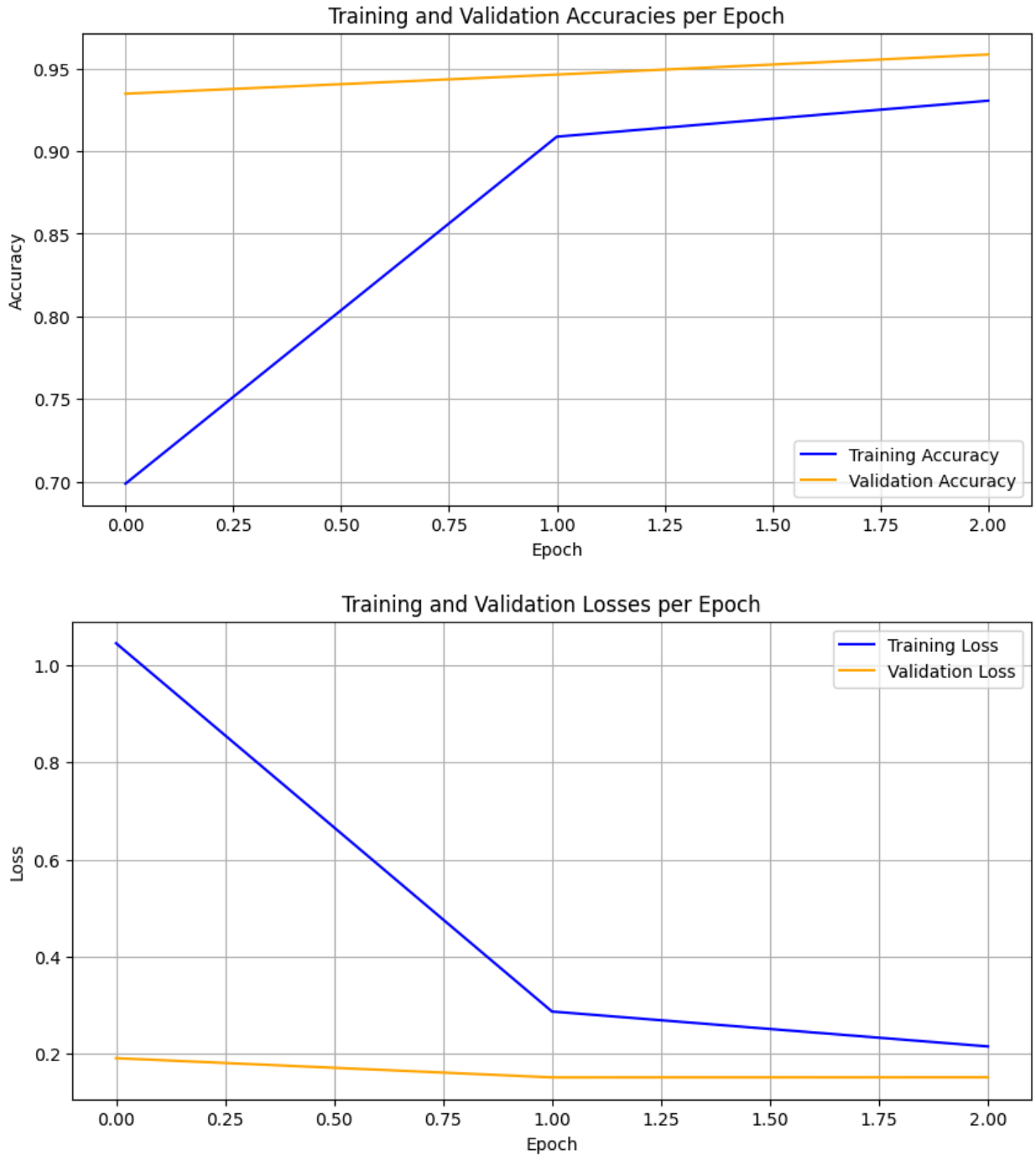


Figure 8: Training and evaluation accuracies (top) and losses (bottom) per epoch for base fruits-360 on ViT.

Base Fruits-360 DeiT Training Metrics

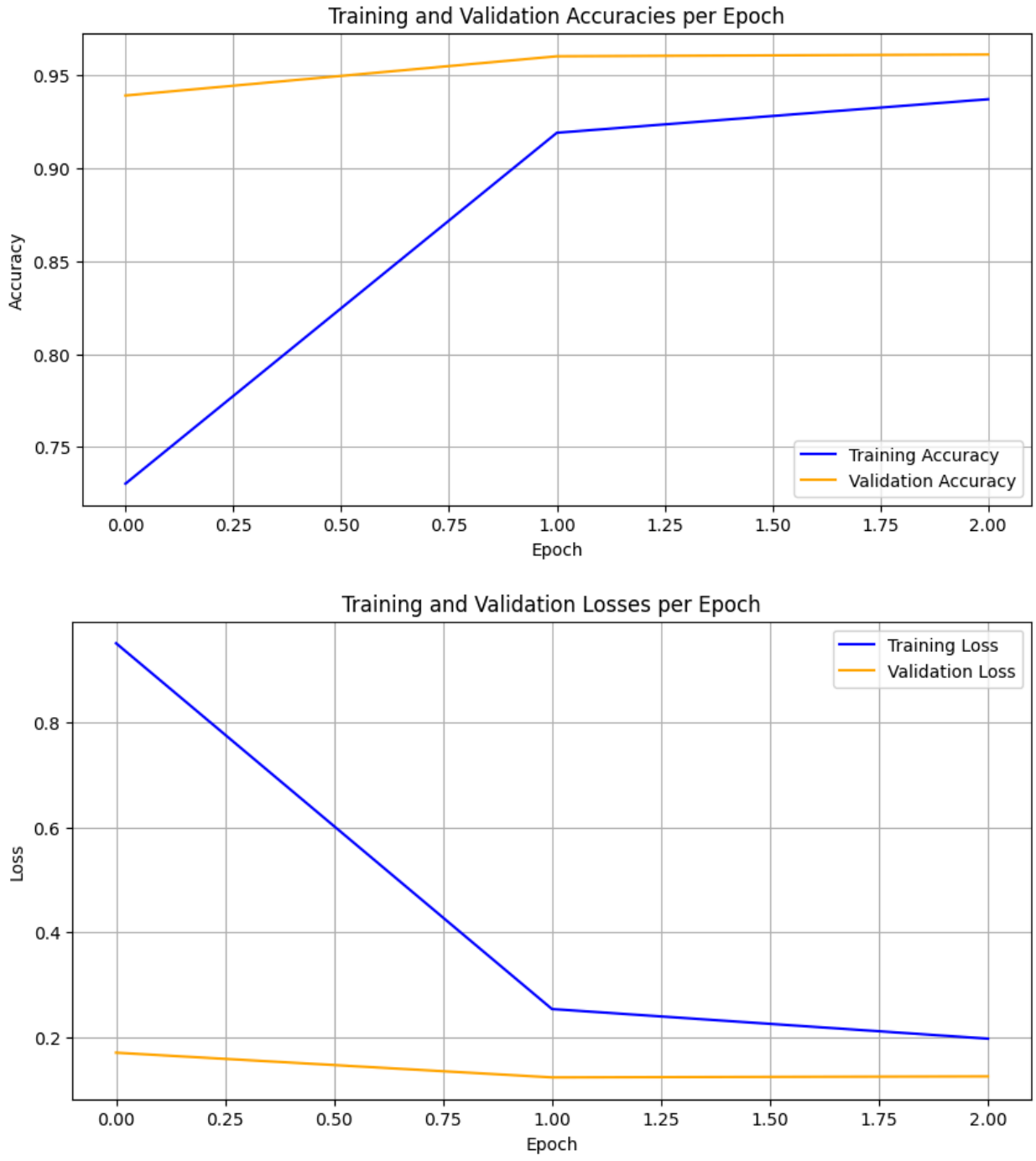


Figure 9: Training and evaluation accuracies (top) and losses (bottom) per epoch for base fruits-360 on DeiT.

Extended Fruits-360 DeiT Training Metrics

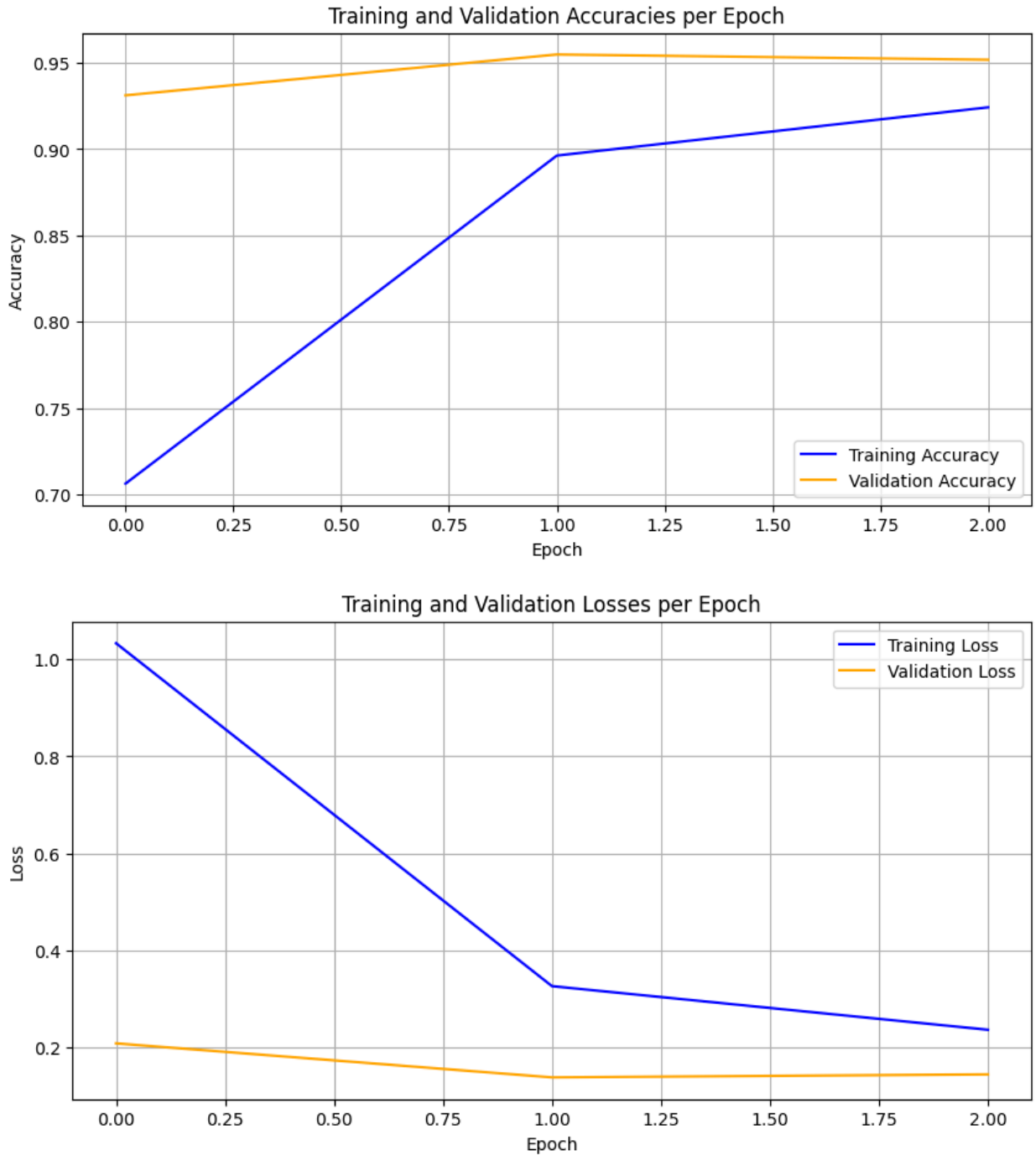


Figure 10: Training and evaluation accuracies (top) and losses (bottom) per epoch for extended-fruits-360 on DeiT.

Combined Model

The final model was tested using the evaluation metrics stated above. These are the results from running the entire pipeline (segmentation, classification, and matching) on the testing datasets.

Results are as follows:

Test Batch 1: Scenes that contained natural images of ingredients. Consist of live scenes (assembled in the “wild” on counter) and augmented scenes (from Google Photos, DALL-E).

Total number of scenes: 92

N (# of results to be returned by matching algorithm)	**Proportion of matches**	**Accuracy (%)**
10	2/92	2.17
100	6/92	6.52
500	17/92	18.48

Test Batch 2: Augmented test scenes created by splicing individual images of ingredients from the validation portion of the Fruits-360 Kaggle dataset.

Total number of scenes: 40

N (# of results to be returned by matching algorithm)	**Proportion of matches**	**Accuracy (%)**
10	16/40	40.00
100	30/40	75.00
500	35/40	87.50

**Note: A recipe is considered correctly matched recipe if the pre-labeled recipe for the scene is within the top N returned recipes.*

Discussion, Limitations, and Future Work

Model Performance and Generalizability

The final accuracy rates of the combined model differed greatly between the two test datasets containing images of different scenes with ingredients. For all values of N (i.e. the number of recipes the RecipeMatcher returns for a query), the model performed significantly better on Test 2, or test scenes containing augmented scenes created using unseen extended-fruits-360 test dataset images.

The significantly higher accuracy rates for the test scenes that use images from the validation extended-fruits-360 dataset suggest a lack of generalizability of the fine-tuned DeiT model. Although an evaluation of the fine-tuned DeiT model revealed high accuracy rates when tested on the validation extended-fruits-360 dataset, it fails to correctly classify many food items in Test 1. This discrepancy is likely because the ingredients in the test dataset differ in appearance from those in the training dataset. For example, the images of cauliflower in the training dataset contrast significantly with those in the test dataset. To address this challenge, a further study would be broadening the training dataset to encompass diverse images of ingredients resembling those encountered in real-world testing environments.

As discussed in the section on model architecture, we decided to have two separate phases for segmentation and classification. An advantage of this approach is that the segmentation phase ensures that all ingredients are individually segmented and run through the classification model. This ensures that no ingredients are missed in the process. However, one of the limitations is that this approach requires that the classification model is able to generalize well to an image of any ingredient. In our case, we found that the classification model is able to recognize images of ingredients similar to those from the training dataset. On the other hand, it did not generalize as well to images in Test Batch 1 of ingredients in the "wild." It may be that a model fine-tuned to perform both segmentation and classification on images in the "wild" would perhaps perform better. Further experiments are needed to understand how we can improve generalizability of classification.

Dataset Limitations

Based on its corresponding paper from 2018, the original Fruits-360 dataset was curated by capturing images of fruits from various angles under different lighting conditions. Augmentations such as rotation, flipping, scaling, and brightness adjustments were applied to increase the diversity of the dataset and improve potential model robustness. For future study, additional work could be done to ensure consistency between the training for the classification phase and the segmentation phase.
While the fine-tuned DeiT model achieved excellent results on the validation set during the classification experiments, it was likely unable to replicate those results on the segments because said segments did not undergo the same preprocessing. Note that this is separate from the preprocessing that we applied to the PyTorch DataLoader objects. For example, some augmentations like changes in brightness must have been applied prior to loading any data into the notebook. For future study, the exact augmentations could be applied to any new images before training the final classification model.

Finally, there may have been some data scarcity and class imbalance in the extended Fruits-360 dataset. Recall that the extended version of the dataset featured 12 additional labels across just 1,802 new images that weren't in the base Fruits-360 dataset. With more labels, it became harder to collect sufficient training data for each of the new classes. Certain classes also had significantly fewer samples than others, leading to the poor generalization that was observed. For future study, the data collection phase for the extended dataset could be mindful of this pitfall and mitigate any scarcity or imbalance.

Test Batches and Scene Creation

Test Batch 1
As previously mentioned in the discussion Scene Datasets Test Batch 1 and Test Batch 2, we faced physical and structural limitations in scene dataset collection. The initial issue we faced was a barrier to creating solely “live” test scenes due to not having all of the specific ingredients available to us for photographing. As such, we decided to create augmented scenes where we spliced “natural” images of ingredients onto one background, and to find real-life pictures of all ingredients assembled via searching on Google Images and on other online platforms. This workaround did not pose an issue for segmentation, and the model was able to segment each ingredient on the one-toned background of the augmented scene.

Test Batch 2
We faced another limitation when we ran the model on the “natural scenes” from Test Batch 1, which produced very low accuracy scores. In order to counter the fact that the model “as-is” did not generalize well to “natural” images of ingredients found via real-life pictures of these ingredients, we decided to create a second Test Batch 2.

Test Batch 2 aims to see whether the model can perform on test scenes where the appearance of each ingredient has less variation. In this way, we can observe whether the segmentation, classification, and matching components of the model work together and produce results without expecting the model to be able to generalize well on variations in ingredient appearance. To achieve this, we used ingredient images provided in the validation portion of the Fruits-360 Dataset, which were not used to train the model - but were only used in the validation stage. In this way, the model is not being tested on images it was trained on - but rather, we are eliminating variation in ingredient form for this portion. For future work, we would aim to expand the training of our current model to include more images of the same ingredient in different stages of ripeness, shadow, and form to account for naturally occurring variations in produce.

Recipe Dataset Ingredient List Matching

There is a discrepancy between the scenes we created for each recipe, and the ingredient list for the recipe from the Epicurious Recipe Database.

The official recipe ingredient lists from Epicurious contained certain ingredients - such as condiments, vegetables, meats, seasonings, etc - that were not present in our training dataset. As such, many of the scenes we created for each recipe ID did not contain the full list of ingredients specified for the recipe. This posed a problem with match for lower values of N matches, as multiple recipes could be matched to a scene that contains a smaller set of ingredients present in both recipes. As such, we had to increase the

number of N for matches when recommending recipes per scene, to account for the fact there are less differentiating ingredients per scene as we could only include the ingredients that are also in the training set.

For future work, we would expand our dataset to include more types of ingredients (classes) such as other types of vegetables, fruits, meats, condiments, etc, that are most common. To identify these items, we could perhaps research top commonly occurring / used food items per household in the United States, for example. Additionally, we would limit recipes from our recipe dataset thatincludes niche ingredients which are not included in our training dataset. This would eliminate the intrinsic problem of matching to recipes for ingredients which we cannot include in test scene data.

Conclusion

The group was able to successfully build the proposed pipeline by the end of the study. To recap, the pipeline first assumes that the input is an image of a scene with various food items. This image is passed into Grounding SAM in order to obtain the segments. Then, the individual segments are passed into the fine-tuned DeiT classifier to obtain the class indices. In order to get text representations of all the available ingredients found in the scene, these class indices are used for lookup in a dictionary that maps all class indices to their respective text representations. Finally, these text representations are passed into a sentence transformer so they can be used in conjunction with the recipe matching algorithm, which will ultimately output the desired number of appropriate recipes. Overall, while the system in this project is not exactly a single multimodal model, it still leverages segmentation, object classification, and natural language processing techniques individually in order to solve this problem.

However, despite these efforts, this problem proved to be more challenging than anticipated, and the results fell short of expectations. Future iterations could explore enhancements in segmentation accuracy, classifier fine-tuning, and recipe matching algorithms to improve performance. Since classification seemed to be the largest bottleneck in this project, future study could experiment with maintaining consistency when processing the scene segments, as well as fine-tuning other models. In addition to revisiting CLIP’s zero shot learning, future work could focus on the tradeoffs offered by other pretrained models, including EfficientNet, MobileNet, ResNeXt, DenseNet, Inception, and SqueezeNet among others. In terms of user experience, integrating feedback mechanisms for user validation and refining the model's understanding of ingredient combinations could enhance recipe relevance and usability. As a whole, these proposed adjustments aim to propel the system towards its intended goal of generating personalized and practical recipes efficiently.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

[2] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 27, 2016. 770–778. https://doi.org/10.1109/CVPR.2016.90

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment Anything.

[4] Jack Leitch. 2021. Building a Recipe Recommendation System. *Medium*. Retrieved May 8, 2024 from https://towardsdatascience.com/building-a-recipe-recommendation-system-297c229dda7b

[5] Yen-Chieh Lien, Hamed Zamani, and W. Croft. 2020. *Recipe Retrieval with Visual Query of Ingredients*. https://doi.org/10.1145/3397271.3401244

[6] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. 2023. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection.

[7] Horea Mureşan and Mihai Oltean. 2021. Fruit recognition from images using deep learning.

[8] Mark Needham. 2019. What's cooking? Part 4: Similarities. *Neo4j Developer Blog*. Retrieved May 8, 2024 from https://medium.com/neo4j/whats-cooking-part-4-similarities-d4443d89556a

[9] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. 2024. Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks.

[10] M. S. Rodrigues, F. Fidalgo, and Â. Oliveira. 2023. RecipeIS—Recipe Recommendation System Based on Recognition of Food Ingredients. *Applied Sciences* 13, (2023), 7880. https://doi.org/10.3390/app13137880

[11] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego,*

CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. . Retrieved from http://arxiv.org/abs/1409.1556

[12] The NYT Open Team. 2023. How The New York Times Cooking Team Makes Personalized Recipe Recommendations. *Medium*. Retrieved May 8, 2024 from https://open.nytimes.com/how-the-new-york-times-cooking-team-makes-personalized-recipe-recommendations-7b86df9b22ec

[13] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention.

[14] Zhu Z, Dai Y. A New CNN-Based Single-Ingredient Classification Model and Its Application in Food Image Segmentation. J Imaging. 2023 Sep 29;9(10):205. doi: 10.3390/jimaging9100205. PMID: 37888312; PMCID: PMC10607895.

[15] Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., and Zhang, L. 2023. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. arXiv preprint arXiv:2303.05499. https://arxiv.org/abs/2303.05499 .

Appendices

Appendix A: Dataset

Table 1: Original Kaggle Fruits-360 Dataset

Ingredient	Varieties/Notes
Apples	Crimson Snow, Golden, Golden-Red, Granny Smith, Pink Lady, Red, Red Delicious
Apricot	-
Avocado	Regular, Ripe
Banana	Yellow, Red, Lady Finger
Beetroot Red	-
Blueberry	-
Cactus fruit	-
Cantaloupe	2 varieties
Carambula	-
Cauliflower	-
Cherry	Different varieties, Rainier
Cherry Wax	Yellow, Red, Black
Chestnut	-
Clementine	-
Cocos	-

Corn	With husk
Cucumber	Ripened
Dates	-
Eggplant	-
Fig	-
Ginger Root	-
Granadilla	-
Grape	Blue, Pink, White (different varieties)
Grapefruit	Pink, White
Guava	-
Hazelnut	-
Huckleberry	-
Kiwi	-
Kaki	-
Kohlrabi	-
Kumsquats	-
Lemon	Normal, Meyer
Lime	-
Lychee	-
Mandarine	-

Mango	Green, Red
Mangostan	-
Maracuja	-
Melon Piel de Sapo	-
Mulberry	-
Nectarine	Regular, Flat
Nut	Forest, Pecan
Onion	Red, White
Orange	-
Papaya	-
Passion fruit	-
Peach	Different varieties
Pepino	-
Pear	Different varieties, Abate, Forelle, Kaiser, Monster, Red, Stone, Williams
Pepper	Red, Green, Orange, Yellow
Physalis	Normal, with husk
Pineapple	Normal, Mini
Pitahaya Red	-
Plum	Different varieties

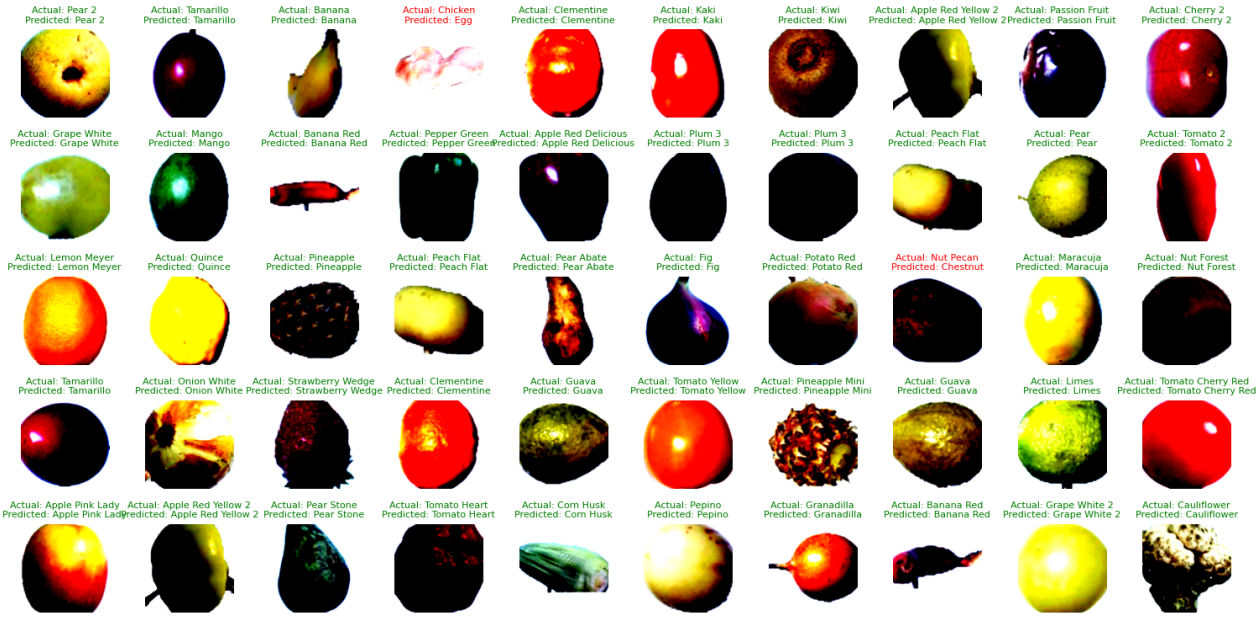
Pomegranate	-
Pomelo Sweetie	-
Potato	Red, Sweet, White
Quince	-
Rambutan	-
Raspberry	-
Redcurrant	-
Salak	-
Strawberry	Normal, Wedge
Tamarillo	-
Tangelo	-
Tomato	Different varieties, Maroon, Cherry Red, Yellow, not ripened, Heart
Walnut	-
Watermelon	-

Table 2: Common Kitchen Staples (Additional Classes)

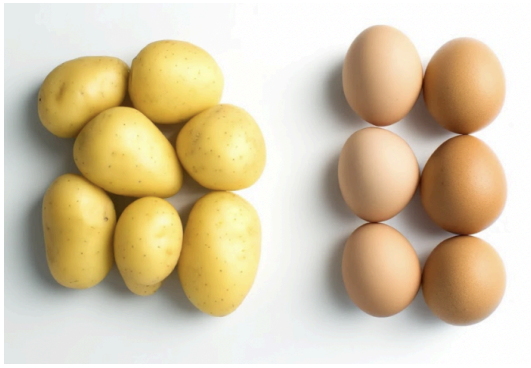
Ingredient	**Notes**
Butter	-

Beef	Images of different cuts, including steaks and ground beef.
Chicken	Images of different cuts, including whole chicken, chicken thighs, etc.
Egg	Images of individual eggs.
Eggs	Images of eggs in cartons.
Flour	Images of packaged flour.
Lamb	Images of different cuts.
Pork	Images of different cuts.
Pasta	Images of packaged pasta of a range of varieties.
Salmon	-
Milk	Images of cartons and differently packaged milk.
Rice	Images of packaged rice.

Table 3: Sample Images



Test Batch 1 Sample Scenes



Test Batch 2 Sample Scenes