# Project Final Report

Devin Chau
Shervan Shahparnia
Department of Computer Science, San Jose State University, San Jose, California

Abstract—Our project aims to address challenges in food deserts and promote healthier food choices by leveraging deep learning for food image classification and recipe generation. Using datasets like Food-11[2] and Food-MNIST[1], we developed a Convolutional Neural Network (CNN)-based model to classify food items. In the future, our goal is to utilize Language Models (LLMs) to generate culturally appropriate recipes that will satisfy the nutrition and dietary needs of those in under served communities. This system seeks to simplify calorie tracking and improve access to affordable, nutritious meals, particularly in under served communities. Despite computational constraints and dataset limitations, our model achieved promising accuracy, paving the way for scalable and impactful applications in food security.

## I. Progress on Data Collection and Preparation

THE project utilizes datasets from Kaggle, specifically Food-11 and food-MNIST. To evaluate the performance of our model before using Food-11, we decided to run the model on the food-MNIST dataset for setup purposes.

### A. Datasets

We used two publicly available datasets from Kaggle: Food-11 and food-MNIST. Below is a brief description of each dataset:

- Food-11: This dataset contains 16,643 images of food items grouped into 11 major categories, including bread, dairy products, desserts, eggs, and more. It is well-suited for multi-class classification tasks and serves as our primary dataset for training and testing.
- food-MNIST: A smaller dataset consisting of 5,000 images of food items.. This dataset was used as a preliminary test for model setup due to its simplicity and smaller size.

### B. Preprocessing

To ensure compatibility with our model, we performed the following preprocessing steps:

- Dataset Restructuring: Reformatted the Food-11 and food-MNIST datasets to match the required folder structure for seamless training, validation, and testing.
- Automation: Developed Bash/Zsh scripts to automate the organization of input datasets, significantly reducing manual effort and errors.
- Data Augmentation: Applied techniques such as rotation, flipping, and scaling to increase the diversity of the training set.
- Normalization: Scaled image pixel values to a range of 0-1 to ensure consistency and improve convergence during training.

## II. Model Architecture

Our project uses a Convolutional Neural Network (CNN) based on the ResNet-50 architecture. ResNet-50 is widely recognized for its deep residual learning framework, which makes it great for image classification tasks by effectively handling issues like vanishing gradients in deep networks. To make this model work for our specific task of classifying food images, we customized the fully connected layer by replacing it with an output layer designed specifically for our dataset. This adjustment allowed us to better classify images into specific food categories during training.

### A. Preliminary Testing with a Smaller Dataset

Before working with larger datasets, we tested our model using a smaller dataset of 5,000 food images [1]. This dataset was chosen because it was simple and smaller in size, therefore needing far less computational power to run tests. It allowed us to test our preprocessing pipeline and confirm the basic functionality of the ResNet-50 architecture without requiring significant computational power. By using this smaller dataset, we ensured that our model's structure and training methods were working correctly before moving on to larger, more complex datasets.

### B. Training and Dataset Integration

Once the preliminary tests were complete, we trained the customized ResNet-50 model using the Food-11 dataset [2]. This dataset is larger and more diverse, which helped the model learn to classify a wide range of food categories. To make the training process more effective, we used techniques like data augmentation and normalization, which helped make the model more robust. We also optimized key settings, such as the learning rate, batch size, and number of epochs, to achieve better classification accuracy.

### C. Inspiration and Code Adaptation

To build and train our model, we studied dozens of GitHub repositories and consulted many references on paperswithcode.com. However, three GitHub repositories stood out and had the structure that we based our initial code off of:

- Food Object Detection with PyTorch FasterRCNN [5]: This repository provided valuable insights into how to preprocess datasets and integrate models in PyTorch. Even though it focused on object detection, we learned techniques that were useful for our project.
- Food Recognition System with PyTorch [6]: This repository helped us understand how to set up a food recognition pipeline. By exploring its code, we learned about efficient training workflows and applied similar strategies to our ResNet-50 model.
- Food Classification with Deep Learning [7]: This repository gave us ideas on how to customize the fully connected layer in a CNN. Its approach to fine-tuning pretrained models for classification tasks directly influenced how we adapted ResNet-50 for our dataset.

### D. Application to Our Project

By analyzing these repositories, we gained the knowledge and tools needed to successfully build and train our model. Specifically:

- We used similar preprocessing techniques, like resizing and normalizing images, to prepare our dataset for training.
- We applied PyTorch's transfer learning features, as demonstrated in the repositories, to customize ResNet-50 for food classification.
- These resources also helped us troubleshoot issues during training and improve the model's performance.

Overall, ResNet-50's strong architecture combined with what we learned from the GitHub repositories enabled us to build a high-performing food classification model that was tailored to our datasets.

### E. Application to Our Project

By analyzing these repositories, we learned how to preprocess datasets, fine-tune pretrained models, and optimize training parameters. The code provided in these repositories served as a foundation for understanding and implementing key concepts in our project. Specifically:

- We adopted similar data preprocessing techniques, including resizing and normalization, to prepare our images for training.
- We leveraged PyTorch's transfer learning capabilities, as demonstrated in the repositories, to adapt ResNet-50 for food classification.
- These resources also helped us debug issues during training and improve the efficiency of our model.

The combination of ResNet-50's robust architecture and the knowledge gained from these repositories enabled us to create a high-performing food classification model customized for our datasets.

## III. Challenges We Faced

- Data Quality: The accuracy of the model depends on the quality and diversity of the training data.
- Diversity: Though we may have thousands of different kinds of foods in our dataset, we may struggle with items that are less popular as our datasets mostly comprise of common ingredients in the United States.
- Model Complexity: Convolutional Neural Networks (CNNs) can be complex and computationally demanding. The run times that we face take a great amount of time causing issues such as the reliance of using our computers instead of our laptops to do work. The computational power required. On top of that, we will require the use of a Large Language Model to allow users to interact with the model to understand and create recipes on their own.
- Technological Limits: There are hundreds of thousands of different ingredients, each requiring training, which would demand massive computational power.
- Model Failure: Failed to connect to a ChatBox to create recipes from the ingredients generated by the model.

## IV. Evaluation Metric

To ensure the reliability and accuracy of our model, we employed a comprehensive suite of evaluation metrics. These metrics shall gauge the predictive accuracy of the model and assess its practical applicability. The chosen metrics are designed to provide a multi-dimensional understanding of the model's performance. These metrics are as follows:

- Accuracy and Loss
- Confusion Matrix Analysis
- Recipe Diversity

The precision in recognizing certain ingredients is the backbone of our model for creating recipes for users. If the model fails to recognize ingredients and mislabels the item, it will create recipes based on inaccurate results, causing problems for the user.

Fig. 1. Training Accuracy History of Food-11 Dataset


## V. Graphs

- In Figure 1, we can see the training accuracy over time increase over the course of many epochs. We achieve roughly above an 70% accuracy for our model in training using the Food-11 Dataset.

Fig. 2. Training Data Confusion Matrix of Food-11 Dataset

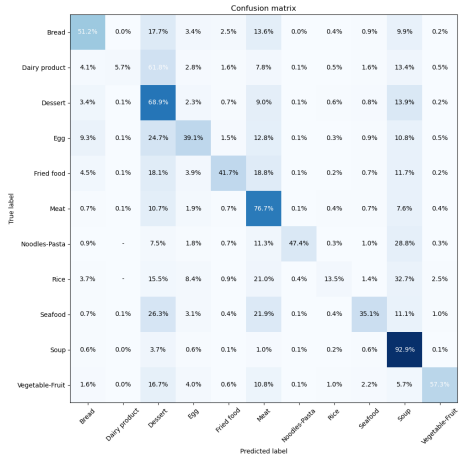



Fig. 3. Validation Data Confusion Matrix of Food-11 Dataset

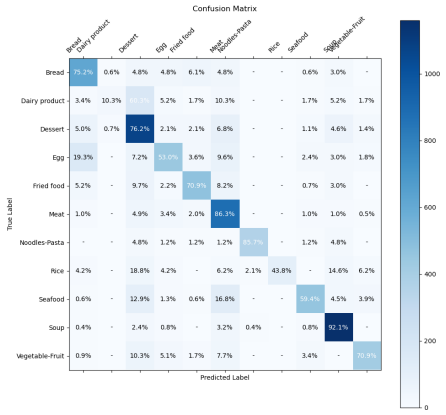
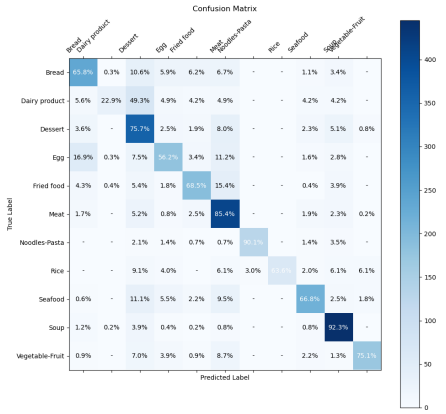



Fig. 4. Testing Data Confusion Matrix of Food-11 Dataset




- In Figure 2, we can see how the model predicts certain food groups and their percentage of occurrences. We can see how the model may predict why they may predict certain images as different food groups.
- In figure 3, we can see the validation confusion matrix of Food-11 Dataset where it helps us reduce over-fitting for our model.
- In Figure 4, we can see the testing data confusion matrix of Food-11 Dataset which is allowing us to see how well our model does during testing and recording its accuracy.
- In Figure 5, we can see the distribution of foods in the data set, each categorized by generalized terms, such as dairy or bread.
- In Figure 6, we see the items that are classified by our CNN model, with each image assigned to its respective category.

Fig. 5. Distribution of the 11 food items

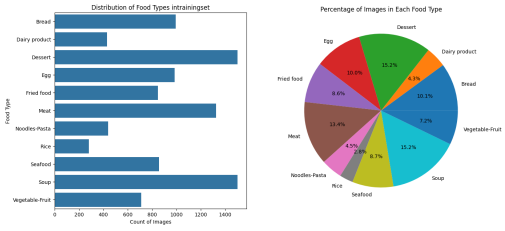
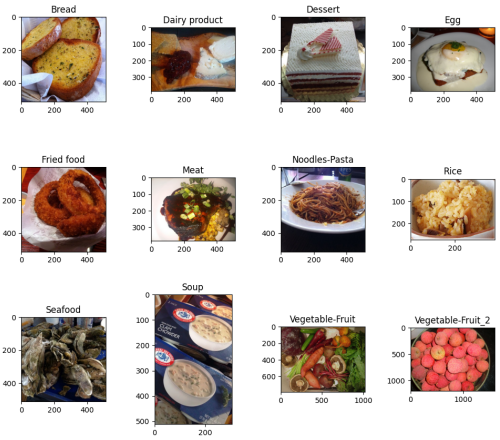



Fig. 6. Images of ingredients from the data




## VI. Future Adjustments

Scaling this model will become harder when more dishes and recipes become introduced as well as new ingredients as stated in the challenges. We initially studied AdaptFormer[3] until we learned that it tends to perform worse than a simple CNN model unless the dataset is substantially larger than the one we require. Therefore, we hope to use a more efficient model using a Swin Transformer[4] to improve image classification and vision. Though it may require extensive computational power, for our future adjustments, we aim to weigh the difference between the computational need and the computational benefits of the model. Improvement we can make for the future are:

- Tailor to user's experiences

  - Our model may repeat certain recipes if listed the exact same combinations, so we must find a way to avoid redundancy in recipes. We want to be able to store all the users' previous recipes to avoid repetitiveness, elevating our user's experience.
- Models for different culture
  - In the future, one expansion that would allow for widespread use, would be to incorporate different datasets. This would tailor to different cuisines that the user chooses from based on their ingredients. The reason why this becomes more difficult is that even with 11 food groups, it requires lots of computational power to do so. If we plan to switch to the Swin Transformer as stated before, it will require even more computational power in return for accuracy (although the accuracy improvement of the Swin Transformer on average is over 15% higher than that of a simple CNN-model). Instead of 11 food groups, the vast variety of food groups will grow to thousands when new ingredients are introduced in different parts of the world. It is not impossible, but it would leave to some difficulty.

## References

[1] Srohit. Food MNIST. GitHub repository, 2019. Available: https://github.com/srohit0/food_mnist.

[2] Trolukovich. Food-11 Image Dataset. Kaggle, 2019. Available: https://www.kaggle.com/datasets/trolukovich/food11-image-dataset.

[3] S. Chen, C. Ge, Z. Tong, J. Wang, Y. Song, J. Wang, and P. Luo. AdaptFormer: Adapting Vision Transformers for Scalable Visual Recognition. arXiv preprint, 2022. Available: https://arxiv.org/abs/2205.13535.

[4] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv preprint, 2021. Available: https://arxiv.org/abs/2103.14030.

[5] Kosletr. Food Object Detection with PyTorch Faster-RCNN. GitHub repository, 2024. Available: https://github.com/kosletr/Food-Object-Detection-Pytorch-FasterRCNN.

[6] Ivan Whaf. Food Recognition System with PyTorch. GitHub repository, 2024. Available: https://github.com/ivanwhaf/FRS-pytorch.

[7] Ismael Deka. Food Classification with Deep Learning. GitHub repository, 2024. Available: https://github.com/Ismael-Deka/Food-Classification-DL.