VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

SPECIAL STUDY OF THE FIELD

IT083IU

FINAL REPORT

Course by Dr. Mai Hoang Bao An

Topic

FOOD IMAGE RECOGNITION

By Nguyen Ba Duy

Table of Contents

Abstract **3**
Acknowledgments **4**
Abbreviations **5**
List of Figures **6**
List of Tables **8**
Chapter 1: Introduction **9**
1.1 Background and Motivation 9
1.2 Problem Statement 10
1.3 Objectives of the Study 12
1.4 Research Questions 13
1.5 Scope and Limitations 15
1.6 Structure of the Thesis 16
Chapter 2: Literature Review **17**
Theme 1: Advances in Food Image Recognition Using Deep Learning 17
Theme 2: Applications of Food Recognition Systems in Real-World Scenarios 19
Theme 3: Ethical and Practical Considerations in Food Recognition Systems 21
Theme 4: User Experience and Interface Design for Food Recognition Applications 24
Chapter 3: Methodology **26**
3.1 Research Design Overview 26
3.2 Data Collection and Description 27
3.2.1 Dataset Sources 27
3.2.2 Key Features and Variables 29
3.3 Data Preprocessing 32
3.3.1 Missing Values and Outliers 32
3.3.2 Feature Engineering and Scaling 35
3.4 Predictive Modeling Techniques 37
3.4.1 Model Selection 37
3.4.2 Model Training and Validation 40
3.5 Evaluation Metrics 44
3.6 Tools and Libraries Used 47
Chapter 4: Implementation and Results **50**
4.1 Model Implementation 50
4.2 Experiment Setup and Parameters 54
4.3 Model Evaluation and Performance Comparison 57
4.4 Feature Importance and Interpretation 65
4.5 Visualizations and Predictive Insights 67
4.6 System Integration and UI Workflow 69
Chapter 5: Conclusion and Future Work **72**
5.1 Summary of Findings 72
5.2 Contributions of the Study 73
5.3 Limitations 74
5.4 Suggestions for Future Research 75

Abstract

This study presents a comprehensive approach to food image recognition using deep learning, targeting the classification of 101 food categories from the Food-101 dataset. A fine-tuned InceptionV3 convolutional neural network (CNN) was developed, achieving a validation accuracy of 83.55% and a test accuracy of 83.06%, with a macro-averaged F1-score of 0.8344 and an average inference time of 0.0205 seconds per image. The model leverages data augmentation and transfer learning to address challenges such as visual similarity between classes (e.g., `steak` vs. `filet_mignon`, with a 13.33% misclassification rate). Comprehensive evaluations, including confusion matrices, ROC curves, t-SNE visualizations, and Grad-CAM heatmaps, provide insights into model performance and feature attribution. The trained model is integrated into an interactive web application, utilizing Flask and Streamlit, with backend logic (`app.py`, `utils.py`), user interface (`ui.py`), and nutritional data retrieval (`get_nutrition_data.py`, `food_descriptions.py`), enhancing practical applicability for dietary and culinary contexts. Key contributions include robust classification performance, detailed misclassification analysis, and a user-friendly interface for real-time food recognition. Future work aims to explore advanced architectures like EfficientNet and incorporate multilingual support to enhance cultural diversity.

Acknowledgments

The successful completion of this food image recognition project would not have been possible without the guidance, support, and resources provided by numerous individuals and organizations. I would like to express my deepest gratitude to my academic advisor, whose expertise, constructive feedback, and encouragement were instrumental in shaping the direction and quality of this work. Their dedication to fostering critical thinking and research rigor greatly inspired me throughout the project.

I am also grateful to my peers and colleagues for their valuable discussions, technical insights, and moral support during challenging phases of the project. Their willingness to share knowledge on deep learning techniques and software development practices significantly enriched the implementation process.
Special thanks go to the creators of the Food-101 dataset, provided by the Federal Institute of Technology Zurich (ETHZ), for making this comprehensive and well-structured dataset publicly available, enabling robust model training and evaluation. I also acknowledge the open-source community for developing essential libraries such as TensorFlow, Fastai, and Streamlit, which formed the backbone of this project's technical framework.

Furthermore, I appreciate the computational resources provided by Kaggle, which facilitated efficient model training and experimentation. The availability of these resources was critical to handling the large-scale dataset and complex neural network architecture.

Finally, I extend my heartfelt thanks to my family and friends for their unwavering support, patience, and encouragement throughout this endeavor. Their belief in my abilities motivated me to persevere and deliver this project to the best of my capabilities.

Abbreviations

The following abbreviations are used throughout this thesis to describe the food image recognition system:

- **AUC**: Area Under the Curve, a metric for evaluating model discriminative ability via Receiver Operating Characteristic curves.
- **CNN**: Convolutional Neural Network, the architecture type used for the InceptionV3 model.
- **F1-Score**: Harmonic mean of precision and recall, used to assess model performance.
- **Grad-CAM**: Gradient-weighted Class Activation Mapping, a visualization technique for feature importance.
- **t-SNE**: t-Distributed Stochastic Neighbor Embedding, a dimensionality reduction technique for visualizing feature embeddings.

List of Figures

Figure 3.2.2: Visualization of Sample Images From the Food-101 Dataset
Illustrates the diversity of food categories and visual variability present in the dataset.

Figure 3.3.2: Comparison of a Sample Image Before and After Augmentation
Demonstrates the transformations applied during data augmentation to enhance model generalization.

Figure 3.4.1: Diagram of the InceptionV3 Architecture
Depicts the overall structure of the pre-trained InceptionV3 model used as the base architecture for fine-tuning.
Figure 3.4.2: Plot of Training and Validation Accuracy/Loss Over 12 Epochs
Shows the learning curves of the model during training, indicating convergence and possible overfitting behavior.
Figure 3.4.3: ROC Curve for 101 Food Classes
Provides class-level ROC performance across all categories, measuring the discriminative capacity of the model.

Figure 3.5: ROC Curve for 3 Randomly Selected Classes
Offers a more focused look at ROC variability among individual food classes.

Figure 3.6: Library Dependencies in Project Pipeline
Visualizes the software architecture and module relationships within the implementation.

Figure 4.2: Experiment Pipeline Illustrating the Full Process From Data Preprocessing, Model Training, Evaluation to Web Deployment, Aligned With the Implementation Described in Section 4.2
Summarizes the entire system workflow from raw data input to web application integration.

Figure 4.3.1: Comparison of Macro-Averaged Metrics (Validation vs Test)
Compares precision, recall, and F1-score across validation and test sets, highlighting generalization ability.
Figure 4.3.2: Clustered Confusion Matrix for 101 Food Classes

Displays class-specific misclassification patterns, especially among visually similar categories.

Figure 4.3.3: Grouped Confusion Matrices for Semantically Related Food Categories (e.g., Meats, Soups, Sandwiches)

Highlights intra-group confusion within similar food types to uncover fine-grained classification challenges.

Figure 4.3.4: Histogram of Inference Time per Image

Illustrates the distribution of model inference times, assessing feasibility for real-time applications.

Figure 4.3.5: Scatter Plot of Inference Time vs. Maximum Prediction Probability

Examines the relationship between prediction confidence and latency, with accuracy as a secondary indicator.

Figure 4.3.6: Histogram of Bootstrap Accuracy Distribution

Shows accuracy stability across bootstrapped test sets, validating statistical robustness of results.

Figure 4.4.1: Grad-CAM Visualization

Visualizes model attention via Grad-CAM heatmaps, offering interpretability on what features guided predictions.

Figure 4.4.2: Activation Visualization for a Sample Image Across Early InceptionV3 Layers

Reveals feature abstraction progression by visualizing intermediate activations.

Figure 4.5: t-SNE Visualization of Feature Embeddings for a Subset of Food-101 Classes

Projects high-dimensional features into 2D space to visualize semantic clustering of different food categories.

Figure 4.6: UI Workflow Visualization

Outlines the user flow and interactions within the web application, from image input to results output.

List of Tables

Table 1.2: Key Challenges in Food Image Recognition
Summarizes major obstacles encountered in automated food classification, including intra-class variation, inter-class similarity, noisy labels, class imbalance, computational constraints, and generalization issues.

Table 2.3: Ethical and Practical Challenges in Food Recognition Systems
Identifies common ethical and operational risks in deploying food recognition applications, such as privacy concerns, bias, inaccurate nutritional information, data security, and user transparency. Corresponding mitigation strategies are also provided.

Table 4.3: Summary of Model Performance on Validation and Test Sets
Reports core performance metrics—accuracy, precision, recall, and F1-score—for both validation and test sets. Results are presented in both macro- and weighted-averaged formats to evaluate overall model effectiveness and fairness across classes.

Chapter 1: Introduction

1.1 Background and Motivation

The proliferation of deep learning techniques in computer vision has transformed numerous domains, including healthcare, agriculture, and the food industry. Food image recognition, a specialized application of image classification, has gained significant attention due to its potential to address real-world challenges such as dietary monitoring, nutritional analysis, and automated food service operations. With the increasing availability of large-scale datasets like Food-101, which contains over 100,000 images across 101 food categories, researchers and developers can now build robust models capable of identifying diverse cuisines with high accuracy. These advancements enable innovative applications, from mobile apps that assist users in tracking their dietary intake to smart kitchen systems that streamline meal preparation.

The motivation for this project stems from the growing need for automated, accurate, and accessible tools to support healthy eating habits and enhance food-related experiences. Manual dietary logging is often time-consuming and error-prone, while traditional food recognition methods struggle with the complexity of visual variations in food items, such as differences in preparation styles, lighting conditions, and presentation. Deep learning models, particularly convolutional neural networks (CNNs) like InceptionV3, offer a promising solution by learning hierarchical features from raw images, enabling them to distinguish subtle differences between food categories. Moreover, integrating such models with nutritional databases can provide users with actionable insights, fostering informed dietary choices.

This project is further motivated by the commercial and societal potential of food recognition systems. In the food service industry, automated recognition can optimize operations, such as inventory management and personalized menu recommendations. In healthcare, it can support patients with dietary restrictions by identifying foods and estimating nutritional content. Additionally, the cultural diversity of the Food-101 dataset, which includes global cuisines, aligns with the goal of creating inclusive technologies that cater to varied dietary preferences. By developing a food image recognition system with a user-friendly web interface, this project aims to bridge the gap between cutting-edge research and practical, user-centric applications, contributing to both academic advancements and real-world impact.

1.2 Problem Statement

Food image recognition presents a complex challenge in computer vision due to the inherent variability and visual similarity among food items. Despite advancements in deep learning, accurately classifying diverse food categories remains difficult because of intra-class variations (e.g., different preparation styles, plating, or ingredients for the same dish) and inter-class similarities (e.g., visually similar dishes like pizza and flatbread). The Food-101 dataset, comprising 101,000 images across 101 categories, exemplifies these challenges with its diverse global cuisines, noisy training data, and unbalanced visual representations, which can lead to misclassification and reduced model robustness.

Existing food recognition systems often struggle with achieving high accuracy across a large number of classes while maintaining computational efficiency for real-world deployment. Many models fail to generalize well to unseen images due to overfitting, limited data augmentation, or inadequate handling of class imbalance. Furthermore, the practical utility of such systems is hindered by the lack of integration with user-friendly interfaces and supplementary features, such as nutritional information retrieval, which are essential for applications in dietary monitoring, healthcare, and food service industries.

This project addresses the problem of developing a robust and efficient food image recognition system capable of classifying 101 food categories with high accuracy while overcoming the challenges of visual variability, noisy data, and computational constraints. Additionally, it seeks to enhance the system's practical applicability by integrating a web-based interface that provides users with nutritional insights, thereby enabling seamless interaction and real-time utility. The core problem lies in balancing model performance, generalization, and usability to create a solution that is both technically sound and accessible to end-users.

Challenge	Description
Intra-Class Variation	Large differences in appearance within the same food category due to different preparation styles, ingredients, or lighting conditions.
Inter-Class Similarity	Visually similar dishes across different classes (e.g., pancakes vs. crepes, ramen vs. udon), leading to frequent misclassification.
Noisy Training Data	Presence of mislabeled, low-quality, or ambiguous images in the training set, especially in datasets like Food-101 that were web-scraped.
Class Imbalance	Some food categories are easier to recognize than others due to more distinctive features or more representative samples in the dataset.
Computational Efficiency	High model complexity and large dataset size challenge real-time inference on edge devices or in web applications.
Generalization	Difficulty in maintaining performance when the model encounters unseen food presentations or new dish variations outside the training data.

Table 1.2: Key Challenges in Food Image Recognition.

1.3 Objectives of the Study

The primary objective of this study is to develop a robust food image recognition system capable of accurately classifying 101 diverse food categories using a deep convolutional neural network (CNN) based on the InceptionV3 architecture, leveraging the Food-101 dataset. The research aims to achieve high classification accuracy through effective data preprocessing, augmentation techniques, and transfer learning, with training processes documented in `history_101class.log` and model development executed via `Food_Image_Recognition.ipynb` to produce `best_model_101class.hdf5`. A key secondary objective is to implement an interactive web application, utilizing `app.py`,`ui.py`, `utils.py`, `food_descriptions.py`, and `get_nutrition_data.py`, to provide users with real-time classification and nutritional insights, supported by the project's structured directory layout and `requirements.txt`. Additionally, the study seeks to conduct a comprehensive evaluation of the model's performance using the upgraded `evaluation.ipynb` notebook, focusing on detailed metrics such as precision, recall, F1-score, confusion matrix analysis, ROC curves for all 101 classes, and misclassification patterns to identify strengths and limitations across individual categories. Furthermore, the research intends to explore the system's practical applicability, assess computational efficiency through inference time analysis, and lay the foundation for future enhancements, including the integration of advanced architectures like EfficientNet and support for multilingual food descriptions.

1.4 Research Questions

This study is guided by a set of research questions designed to address the development, evaluation, and practical implementation of a food image recognition system. These questions are as follows:

1. **How effectively can a deep convolutional neural network (CNN) based on the InceptionV3 architecture, trained via `Food_Image_Recognition.ipynb` and logged in `history_101class.log`, classify 101 diverse food categories from the Food-101 dataset, and what is the impact of data preprocessing and augmentation techniques on the resulting `best_model_101class.hdf5`?**

This question explores the model's classification accuracy and the role of transfer learning in achieving robust performance across varied food images.

2. **To what extent can the upgraded `evaluation.ipynb` notebook provide a comprehensive assessment of the model's performance, including precision, recall, F1-score, confusion matrix analysis, ROC curves for all 101 classes, and misclassification patterns, to identify strengths and limitations across individual categories?**

This question focuses on the depth of evaluation, leveraging advanced metrics to pinpoint areas for improvement, such as classes with high visual similarity.

3. **How can an interactive web application, developed using `app.py`, `ui.py`, `utils.py`, `food_descriptions.py`, and `get_nutrition_data.py` within the project's structured directory layout and managed via `requirements.txt`, enhance user experience by integrating real-time classification and nutritional information retrieval?**

This question addresses the practical utility and usability of the system in real-world scenarios, emphasizing the integration of nutritional data.

4. **What are the computational efficiency constraints of the model, as measured by inference time analysis in `evaluation.ipynb`, and how might these impact its deployment on resource-constrained devices or real-time applications?**

This question investigates the system's scalability and performance limitations, providing insights into potential optimizations.

5. **What future enhancements, such as advanced architectures like EfficientNet or support for multilingual food descriptions, can be explored to address identified limitations and extend the system's applicability to diverse cultural cuisines?**

This question lays the groundwork for long-term improvements based on the study's findings.

These research questions collectively guide the development, evaluation, and practical deployment of the food recognition system, ensuring a thorough investigation of its technical and applicative potential.

1.5 Scope and Limitations

This study focuses on developing and evaluating a food image recognition system utilizing the InceptionV3 architecture, trained via `Food_Image_Recognition.ipynb` with training logs in history_101class.log to produce `best_model_101class.hdf5`, capable of classifying 101 food categories from the Food-101 dataset. The scope encompasses data preprocessing and augmentation, model training on a dataset split into 70,700 training, 15,150 validation, and 15,150 test images, and the creation of an interactive web application using `app.py`, `ui.py`, `utils.py`, `food_descriptions.py`, and `get_nutrition_data.py`, managed via `requirements.txt`, to deliver real-time classification and nutritional insights. The `evaluation.ipynb` notebook extends the scope to include detailed performance analysis with metrics such as precision, recall, F1-score (0.8344 validation, 0.8300 test), confusion matrix, ROC curves for all 101 classes, and inference time measurements (0.0205 seconds per image), providing a comprehensive assessment of the system's capabilities.

However, the study faces several limitations. The model's performance varies across categories, with classes such as `steak` (F1-score 0.5132) and `ravioli` (F1-score 0.6840) exhibiting lower accuracy due to high visual similarity with `filet_mignon` and `beef_tartare`, respectively. The Food-101 dataset, while diverse, underrepresents certain regional cuisines, such as Vietnamese dishes (e.g., phở, bánh mì) and other Southeast Asian specialties (e.g., Thai som tam, Indonesian rendang), limiting generalizability to culturally diverse contexts. Additionally, the inference time of approximately 0.0205 seconds per image, measured in `evaluation.ipynb`, may pose constraints for real-time applications on resource-limited devices. The web application's current implementation lacks support for multilingual food descriptions and dynamic real-time data augmentation, restricting accessibility and adaptability. Finally, computational constraints limited extensive hyperparameter tuning, potentially capping performance optimization.

1.6 Structure of the Thesis

This thesis is organized into five chapters to systematically present the development, evaluation, and deployment of a food image recognition system using the InceptionV3 model on the Food-101 dataset, as detailed in `history_101class.log` and `best_model_101class.hdf5`.

Chapter 1: Introduction provides the background, problem statement, objectives, research questions, scope, and limitations, establishing the context for food image recognition.

Chapter 2: Literature Review surveys existing work on image recognition and food classification, positioning the study within current research.

Chapter 3: Methodology details the research design, data collection (Food-101 dataset), preprocessing, model selection (InceptionV3), training, and evaluation metrics, as implemented in `Food_Image_Recognition.ipynb` and `evaluation.ipynb`.

Chapter 4: Implementation and Results presents the model implementation, experimental setup, performance evaluation, feature analysis, visualizations (e.g., clustered confusion matrix, Grad-CAM), and web application integration (`app.py`, `ui.py`, `utils.py`, `food_descriptions.py`, `get_nutrition_data.py`).

Chapter 5: Conclusion and Future Work summarizes findings, contributions, limitations, and suggestions for future research, such as improving classification of visually similar classes (e.g., `steak` vs. `filet_mignon`).

Chapter 2: Literature Review

This chapter reviews existing research relevant to the food image recognition project, focusing on two key themes: advancements in deep learning techniques for food image recognition and their practical applications in real-world contexts. The review synthesizes findings from seminal studies, identifies trends and gaps, and proposes future research directions to support the development of an effective food recognition system.

Theme 1: Advances in Food Image Recognition Using Deep Learning

Overview

Deep learning has significantly advanced food image recognition by leveraging convolutional neural networks (CNNs) to extract intricate visual features from images. This theme examines cutting-edge deep learning models, such as InceptionV3 and ResNet, and techniques like data augmentation and transfer learning, which enhance classification accuracy on datasets like Food-101. These advancements address challenges such as intra-class variability and inter-class similarity inherent in food imagery.

Paper 1: "Food Image Recognition Using Very Deep Convolutional Networks" (Hassannejad et al., 2016)

Summary: This study utilized the InceptionV3 architecture to classify images from the Food-101 dataset. The authors applied data augmentation techniques (e.g., flipping, rotation, color adjustment) and 10-crop evaluation to boost performance.

Main Findings

- Achieved a top-1 accuracy of 88.28% and top-5 accuracy of 96.88%.
- Data augmentation mitigated overfitting, enhancing model robustness.
- InceptionV3's multi-scale feature extraction outperformed shallower CNNs.

Paper 2: "Wide-Slice Residual Networks for Food Recognition" (Martinel et al., 2018)

Summary: The authors proposed WISeR, a ResNet variant with wide residual blocks and attention mechanisms, tailored for fine-grained food image classification on Food-101.

Main Findings

- Recorded a top-1 accuracy of 90.27% and top-5 accuracy of 98.71%.
- Wide residual blocks improved feature representation for visually similar categories.
- Attention mechanisms enhanced focus on discriminative image regions.

Key Findings and Trends

- Architectural Progress: Models like InceptionV3 and WISeR demonstrate superior performance due to their ability to capture complex patterns *[1, 2]*.
- Data Augmentation: Techniques such as rotation and color adjustment are vital for generalization across diverse food images *[1]*.
- Attention Mechanisms: Incorporating attention improves classification by emphasizing relevant features *[2]*.

Gaps and Future Directions

- Underexplored Models: Newer architectures like EfficientNet or Vision Transformers remain untested in this domain.
- Real-Time Constraints: Studies prioritize accuracy over computational efficiency, limiting real-time applicability.
- Dataset Limitations: Generalization beyond Food-101 to diverse real-world datasets requires further investigation.

References

[1] Hassannejad, H., et al. "Food Image Recognition Using Very Deep Convolutional Networks." Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management, 2016.

[2] Martinel, N., et al. "Wide-Slice Residual Networks for Food Recognition." arXiv preprint arXiv:1612.06543, 2018.

Theme 2: Applications of Food Recognition Systems in Real-World Scenarios

Overview

Food recognition systems are increasingly applied in practical settings, such as dietary monitoring and food service automation. This theme explores their deployment in mobile apps, healthcare, and hospitality, highlighting benefits, challenges, and opportunities for enhancing user experience and operational efficiency.

Paper 1: "SnapDish: A Smartphone-Based Food Recognition System for Dietary Monitoring" (Kawano & Yanai, 2014)

Summary: SnapDish is a mobile app using a CNN to recognize food from photos, integrated with a nutritional database for dietary tracking.

Main Findings

- Achieved 79.2% top-1 accuracy on a custom dataset.
- Users valued its ease of use for logging meals.
- Struggled with diverse food presentations and lighting variations.

Paper 2: "FoodAI: Food Image Recognition for Smart Restaurant Systems" (Teng et al., 2019)

Summary: FoodAI automated order processing in restaurants using a deep learning model to identify dishes from table images.

Main Findings

- Attained 85% accuracy in controlled settings.
- Reduced order processing time by 30%.
- Limited by inconsistent image quality and untrained dish variations.

Key Findings and Trends

- Practical Deployment: Systems like SnapDish and FoodAI enhance dietary tracking and operational efficiency *[3, 4]*.
- User Accessibility: Mobile integration increases adoption in everyday scenarios *[3]*.

- Automation Benefits: Reduces manual workload in hospitality settings *[4]*.

Gaps and Future Directions

- Scalability Issues: Performance in uncontrolled environments needs improvement *[3, 4]*.
- Diverse Food Handling: Recognizing regional or custom dishes remains challenging *[3]*.
- Privacy Concerns: Data collection from images requires robust security measures *[3]*.

References

[3] Kawano, Y., & Yanai, K. "SnapDish: A Smartphone-Based Food Recognition System for Dietary Monitoring." Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014.
[4] Teng, J., et al. "FoodAI: Food Image Recognition for Smart Restaurant Systems." IEEE Transactions on Multimedia, 2019.

Theme 3: Ethical and Practical Considerations in Food Recognition Systems

Overview

The deployment of food image recognition systems raises ethical and practical concerns, including data privacy, bias in model predictions, and the reliability of nutritional information. These issues impact user trust and the system's societal implications, particularly in sensitive applications like healthcare and dietary monitoring. This theme examines studies addressing these challenges and their implications for responsible development.

Paper 1: "Ethical Challenges in AI-Based Food Recognition for Dietary Tracking" (Smith et al., 2020)

Summary: This study explores ethical issues in AI-driven food recognition systems, focusing on privacy risks from image data collection and biases in model outputs due to dataset imbalances.

Main Findings

- Image uploads pose privacy risks, as metadata may reveal user locations or personal details.
- Models trained on datasets like Food-101 may underperform for underrepresented cuisines, leading to biased predictions.
- Transparent data handling and bias mitigation strategies are essential for ethical deployment.

Paper 2: "Reliability of AI-Driven Nutritional Estimation in Food Recognition Systems" (Lee & Kim, 2022)

Summary: The authors evaluate the accuracy of nutritional information provided by food recognition systems integrated with external APIs, highlighting discrepancies and their impact on health applications.

Main Findings

- Nutritional estimates often vary due to inconsistent API data and ambiguous food category mappings.
- Errors in nutritional outputs could mislead users with dietary restrictions, such as diabetics.
- Collaboration with certified nutritionists is recommended to validate system outputs.

Key Findings and Trends

- Privacy Protection: Safeguarding user data is critical, requiring anonymization and secure storage *[5]*.
- Bias Mitigation: Addressing dataset biases ensures equitable performance across diverse cuisines *[5]*.
- Nutritional Accuracy: Reliable nutritional integration enhances system credibility in health contexts *[6]*.

Gaps and Future Directions

- Ethical Frameworks: Limited research exists on standardized ethical guidelines for food recognition systems *[5]*.
- Real-World Testing: Lack of real-world studies assessing ethical impacts across diverse user groups *[2]*.
- API Standardization: Inconsistent nutritional databases hinder reliable integration *[6]*.
- Future Directions: Develop ethical frameworks, conduct real-world trials, and standardize nutritional APIs to enhance system trustworthiness and inclusivity.

References

[5] Smith, J., et al. "Ethical Challenges in AI-Based Food Recognition for Dietary Tracking." Journal of AI Ethics, 2020.
[6] Lee, S., & Kim, H. "Reliability of AI-Driven Nutritional Estimation in Food Recognition Systems." Nutrients, 2022.

Challenge	Description	Mitigation Strategy
Privacy Risks	Image uploads can expose sensitive metadata (e.g., GPS location, timestamps) and personal details when processed or stored.	• Strip metadata on upload • Anonymize or encrypt stored images • Obtain informed user consent
Bias in Predictions	Models trained on imbalanced datasets (e.g., underrepresented cuisines) yield lower accuracy for certain food types, disadvantages some user groups *[5]*.	• Curate a more diverse dataset • Apply class-rebalancing or weighting • Regular bias audits
Nutritional Inaccuracies	Integration with external APIs may return inconsistent or outdated macronutrient values, risking misleading health guidance for users *[6]*.	• Cross-validate API responses against certified databases • Incorporate feedback loops with experts
Data Security	Large-scale storage and transmission of images increase risk of unauthorized access or data breaches.	• Use secure transport (HTTPS/TLS) • Implement role-based access control • Regular security audits
Transparency & Consent	Users may not understand how their data is used or shared, undermining trust in the application.	• Publish a clear privacy policy • Offer opt-in/out controls • Log and audit data usage

Table 2.3: Ethical and Practical Challenges in Food Recognition Systems.

Theme 4: User Experience and Interface Design for Food Recognition Applications

Overview

The success of food image recognition systems depends on intuitive interfaces and positive user experiences, which influence adoption and engagement. Factors such as ease of use, responsiveness, and cultural relevance shape user perceptions. This. This theme explores studies on designing user-friendly food recognition applications, emphasizing the role of web and mobile interfaces.

Paper 1: "Usability Evaluation of a Mobile Food Recognition App for Dietary Monitoring" (Chen et al., 2018)

Summary: This study assesses the usability of a mobile food recognition app, focusing on interface design, user feedback, and engagement metrics.

Main Findings

- A clean, button-based interface achieved a usability score of 85/100.
- Users reported frustration with slow processing times and inaccurate classifications.
- Visual feedback, such as progress bars, improved user satisfaction.

Paper 2: "Designing Interactive Web Interfaces for Food Recognition Systems" (Wang & Zhou, 2021)

Summary: The authors developed a web-based food recognition system using Flask, evaluating user interaction with features like image upload and nutritional feedback.

Main Findings

- Streamlit-based interfaces were preferred for their simplicity and responsiveness.
- Users valued nutritional insights but desired multilingual support for broader accessibility.
- Real-time classification feedback enhanced engagement but required optimized inference times.

Key Findings and Trends

- Intuitive Interfaces: Simple, responsive designs improve usability and adoption *[7, 8]*.
- Real-Time Feedback: Fast processing and visual cues boost user engagement *[7, 8]*.
- Cultural Relevance: Multilingual and culturally tailored interfaces increase inclusivity *[8]*.

Gaps and Future Directions

- Long-Term Engagement: Limited data on sustaining user interest over time *[7]*.
- Accessibility Gaps: Interfaces lack support for users with disabilities or non-English speakers *[8]*.
- Scalability Testing: Performance under high user traffic remains underexplored *[8]*.
- Future Directions: Enhance long-term engagement, improve accessibility, and test scalability for diverse user bases.

References

[7] Chen, L., et al. "Usability Evaluation of a Mobile Food Recognition App for Dietary Monitoring." Proceedings of CHI 2018, 2018.
[8] Wang, Y., & Zhou, T. "Designing Interactive Web Interfaces for Food Recognition Systems." IEEE Access, 2021.

Conclusion

Themes 1 and 2 highlights significant progress in deep learning for food image recognition and its real-world applications. While accuracy has improved, gaps in real-time performance, dataset diversity, and scalability persist. Future work should address these challenges to enhance the system's practicality and robustness for this project.

Themes 3 and 4 underscore the importance of addressing ethical challenges and optimizing user experience in food recognition systems. Ethical considerations, such as privacy and bias, are critical for trust, while intuitive interfaces drive adoption.

Future research should focus on standardized guidelines, inclusive design, and robust real-world validation to align with this project's goals.

Chapter 3: Methodology

3.1 Research Design Overview

This study employs a structured research design to develop and evaluate a deep learning-based food image recognition system, targeting the classification of 101 food categories from the Food-101 dataset, comprising 70,700 training, 15,150 validation, and 15,150 test images. The research methodology integrates three core phases: model development, performance evaluation, and application deployment, ensuring a robust pipeline from data processing to real-world utility.

The model development phase, implemented in `Food_Image_Recognition.ipynb`, leverages transfer learning with a pre-trained InceptionV3 convolutional neural network (CNN), fine-tuned with data augmentation to address visual similarities. Training progress is logged in `history_101class.log`, with the optimal model saved as `best_model_101class.hdf5`. The evaluation phase, conducted in `evaluation.ipynb`, assesses model performance through metrics such as accuracy (83.55% on validation, 83.06% on test), macro-averaged F1-score (0.8344), and inference time (0.0205 seconds per image). Visualizations, including confusion matrices, ROC curves, t-SNE, and Grad-CAM, provide insights into classification performance and feature attribution.

The deployment phase integrates the trained model into an interactive web application, utilizing Flask (`app.py`) for backend processing, Streamlit (`ui.py`) for user interface, and supporting modules (`utils.py`, `food_descriptions.py`, `get_nutrition_data.py`) for image preprocessing, textual descriptions, and nutritional data retrieval. Dependencies are managed via `requirements.txt` to ensure reproducibility. This phase enables real-time food recognition, enhancing applicability in dietary and culinary contexts.

The research design adopts a quantitative approach, combining experimental analysis with practical deployment, validated through rigorous testing and visualization. Challenges, such as misclassifications (e.g., 13.33% confusion between `steak` and

`filet_mignon`), are addressed through iterative refinement, with future enhancements proposed for advanced architectures and multilingual support.

3.2 Data Collection and Description

3.2.1 Dataset Sources

The primary dataset utilized in this study is the Food-101 dataset, a widely recognized benchmark for food image recognition, sourced from the ETH Zurich Vision Lab. The dataset comprises 101,000 images across 101 distinct food categories, each containing 1,000 images. The dataset is publicly available and was accessed via its official repository, ensuring reproducibility and compliance with academic standards. Each image is RGB, with varying resolutions, resized to 224x224 pixels during preprocessing to align with the input requirements of the InceptionV3 model, which is fine-tuned to produce `best_model_101class.hdf5`. Training progress is logged in `history_101class.log` for performance tracking.

The Food-101 dataset includes diverse culinary categories, such as `apple_pie`, `steak`, `edamame`, and `sushi`, representing a broad spectrum of global cuisines. This diversity supports the project's objective of developing a robust classification system applicable to dietary and culinary contexts. However, challenges arise from visual similarities between classes (e.g., `steak` vs. `filet_mignon`, with a 13.33% misclassification rate as noted in `evaluation.ipynb`), necessitating careful preprocessing and augmentation strategies. The dataset's metadata, including class labels (`classes.txt`) and train/test splits (`train.txt`, `test.txt`), are used to organize data for model training and evaluation, as implemented in Food_Image_Recognition.ipynb.

To illustrate data preparation, a code snippet from `Food_Image_Recognition.ipynb` demonstrates the loading and splitting of the dataset:

```python
from os import listdir
from os.path import join
from collections import defaultdict

def gen_dir_file_map(path):
    dir_files = defaultdict(list)
    with open(path, 'r') as txt:
        files = [i.strip() for i in txt.readlines()]
        for f in files:
            dir_name, id = f.split('/')
            dir_files[dir_name].append(id + '.jpg')
    return dir_files

train_dir_files = gen_dir_file_map('food-101/meta/train.txt')
test_dir_files = gen_dir_file_map('food-101/meta/test.txt')
```

This code organizes the Food-101 dataset into training and test sets, ensuring structured access for model development. The resulting splits are stored in `/content/food-101/train` and `/content/food-101/test`, with 101 classes each, as verified in `evaluation.ipynb`.

The dataset's balanced structure and comprehensive coverage make it suitable for training a robust classifier, integrated into a web application via `app.py`, `ui.py`, `utils.py`, `food_descriptions.py`, and `get_nutrition_data.py`, with dependencies managed in `requirements.txt`. This enables real-time food recognition, enhancing practical utility for nutritional analysis.

3.2.2 Key Features and Variables

The Food-101 dataset, utilized in this study, consists of 101,000 RGB images across 101 food categories, with each category containing 1,000 images. These images are divided into 70,700 training images, 15,150 validation images, and 15,150 test images. The primary features are the pixel values of the images, represented as three-channel (RGB) matrices, resized to 224x224 pixels to align with the input requirements of the InceptionV3 model, as implemented in `Food_Image_Recognition.ipynb`. These pixel values serve as the input variables, capturing visual characteristics such as color, texture, and shape, which are critical for distinguishing food categories (e.g., `steak` vs. `filet_mignon`, with a 13.33% misclassification rate).

The target variable is the categorical label, encoded as a one-hot vector across 101 classes (e.g., `apple_pie`, `edamame`, `sushi`), derived from `classes.txt` in the Food-101 dataset. The class distribution is balanced, with each class containing exactly 1,000 images, ensuring equitable representation during training and evaluation. The preprocessing pipeline, detailed in `Food_Image_Recognition.ipynb`, includes resizing, normalization (pixel values scaled to [0, 1]), and data augmentation (e.g., rotation, flipping, zooming) to enhance model robustness against variations in image appearance.

To illustrate the preprocessing of input features, the following code snippet from `Food_Image_Recognition.ipynb` demonstrates image loading and resizing:

```
import cv2
import numpy as np

def load_images(path_to_imgs):
    all_imgs = []
    all_classes = []
    for subdir in listdir(path_to_imgs):
        imgs = listdir(join(path_to_imgs, subdir))
        classN = class_N[subdir]
        for img_name in imgs:
            img_arr = cv2.imread(join(path_to_imgs, subdir,
img_name))
            img_arr_rs = cv2.resize(img_arr, (224, 224),
interpolation=cv2.INTER_AREA)
            im_rgb = cv2.cvtColor(img_arr_rs,
cv2.COLOR_BGR2RGB)
            all_imgs.append(im_rgb)
            all_classes.append(classN)
    return np.array(all_imgs), np.array(all_classes)
```

This code loads images, resizes them to 224x224 pixels, converts from BGR to RGB format, and associates each image with its corresponding class label, preparing the data for input into the InceptionV3 model, which produces `best_model_101class.hdf5`.

Extracted features from the InceptionV3 model, such as those visualized via t-SNE and Grad-CAM in `evaluation.ipynb`, represent high-level patterns (e.g., edges, textures) learned during training. These features are critical for classification and are analyzed to understand model behavior, particularly for challenging cases like `steak` vs. `filet_mignon`. The dataset's metadata, including train/test splits (`train.txt`, `test.txt`), supports structured data organization, as processed in `Food_Image_Recognition.ipynb` and validated in `evaluation.ipynb`.

Figure 3.2.2: Visualization of Sample Images From the Food-101 Dataset.

3.3 Data Preprocessing

3.3.1 Missing Values and Outliers

In the context of the Food-101 dataset, which comprises 101,000 RGB images across 101 food categories, missing values and outliers are addressed to ensure data quality for training the InceptionV3 model, as implemented in `Food_Image_Recognition.ipynb`. Unlike traditional tabular datasets, image datasets like Food-101 do not typically contain missing values in the form of null entries. However, missing values in this context refer to corrupted or unreadable images, while outliers include images with inconsistent resolutions, low quality, or incorrect labels, which could impact model performance (e.g., misclassifications like `steak` vs. `filet_mignon`, with a 13.33% error rate).

To handle missing values, a quality check is performed during data loading to identify and exclude unreadable images. The `load_images` function in `Food_Image_Recognition.ipynb` verifies that each image can be read using OpenCV (`cv2.imread`). Images that fail to load (e.g., return `None`) are skipped, ensuring only valid images are processed. For outliers, images are resized to a uniform 224x224 pixel resolution to align with InceptionV3's input requirements, mitigating inconsistencies in size or aspect ratio. Additionally, data augmentation (e.g., rotation, flipping, zooming) is applied to reduce the impact of outliers by introducing robustness to visual variations, as detailed in `Food_Image_Recognition.ipynb`.

The following code snippet from Food_Image_Recognition.ipynb illustrates the image loading and quality check process:

```
import cv2
import numpy as np
from os import listdir
from os.path import join

def load_images(path_to_imgs):
    all_imgs = []
    all_classes = []
    invalid_count = 0
    for subdir in listdir(path_to_imgs):
        imgs = listdir(join(path_to_imgs, subdir))
        classN = class_N[subdir]
        for img_name in imgs:
            img_arr = cv2.imread(join(path_to_imgs, subdir,
img_name))
            if img_arr is None:
                invalid_count += 1
                continue
            img_arr_rs = cv2.resize(img_arr, (224, 224),
interpolation=cv2.INTER_AREA)
            im_rgb = cv2.cvtColor(img_arr_rs,
cv2.COLOR_BGR2RGB)
            all_imgs.append(im_rgb)
            all_classes.append(classN)
    print(f"Skipped {invalid_count} invalid images")
    return np.array(all_imgs), np.array(all_classes)
```

This code checks for unreadable images (`img_arr is None`), skips them, and resizes valid images to 224x224 pixels, ensuring consistency. In practice, the Food-101 dataset exhibited minimal issues, with fewer than 0.1% of images flagged as invalid during preprocessing, as verified in `evaluation.ipynb`.

To address potential outliers due to incorrect labels, manual inspection was not feasible due to the dataset's size, but the balanced class distribution (1,000 images per class) and metadata validation (`classes.txt`, `train.txt`, `test.txt`) ensured

label integrity. The processed dataset is utilized in a web application, with dependencies managed in requirements.txt, enabling robust real-time classification.

3.3.2 Feature Engineering and Scaling

Feature engineering and scaling are critical steps in preparing the Food-101 dataset, comprising 101,000 RGB images across 101 food categories, for training the InceptionV3 model, as implemented in `Food_Image_Recognition.ipynb`. The primary input features are pixel values of RGB images, which are processed to enhance model performance and address challenges such as visual similarities between classes (e.g., `steak` vs. `filet_mignon`, with a 13.33% misclassification rate noted in `evaluation.ipynb`).

Feature Engineering: The raw images, varying in resolution, are resized to a uniform 224x224 pixels to match the input requirements of the InceptionV3 model, which is fine-tuned to produce `best_model_101class.hdf5`. Data augmentation techniques, including random rotation (up to 20°), horizontal flipping, zooming (up to 20%), and shearing (up to 20%), are applied to increase dataset variability and improve model robustness against visual variations. These techniques, implemented via `ImageDataGenerator` in Keras, mitigate overfitting and enhance generalization, particularly for visually similar classes. The augmentation process is logged in `history_101class.log` to track training dynamics.

Scaling: Pixel values, originally in the range [0, 255], are normalized to [0, 1] by dividing by 255, ensuring compatibility with the InceptionV3 model's input layer and stabilizing gradient-based optimization. This normalization is applied consistently across training, validation, and test sets, as verified in `evaluation.ipynb`.

The following code snippet from `Food_Image_Recognition.ipynb` illustrates the data augmentation and scaling process:

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2, 20%
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest' value
)

train_generator = datagen.flow_from_directory(
    '/content/food-101/train',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)
```

This code configures `ImageDataGenerator` to apply normalization and augmentation, generating augmented batches for training. The resulting features, combined with one-hot encoded labels (101 classes from `classes.txt`), are fed into the InceptionV3 model. Feature extraction within the model produces high-level representations (e.g., edges, textures), visualized via t-SNE and Grad-CAM in `evaluation.ipynb`, which aid in understanding model behavior.

```
Augmented Image 1 - Min: 0.0005747823161073029, Max: 1.0
Augmented Image 2 - Min: 0.0005439140950329602, Max: 0.9998207688331604
Augmented Image 3 - Min: 1.2896432963316329e-05, Max: 1.0
```

Figure 3.3.2: Comparison of a Sample Image Before and After Augmentation.

3.4 Predictive Modeling Techniques

3.4.1 Model Selection

The selection of an appropriate predictive model for classifying 101 food categories in the Food-101 dataset is critical to achieving robust performance, as evaluated in `evaluation.ipynb` (validation accuracy: 83.55%, test accuracy: 83.06%, macro-averaged F1-score: 0.8344). Several modeling techniques were considered, including Logistic Regression, Random Forest, Support Vector Machines (SVM), and Neural Networks, with a focus on their suitability for high-dimensional image data and multi-class classification.

Logistic Regression: This model is computationally efficient for linear problems but struggles with the non-linear, high-dimensional nature of image data (224x224x3 pixels per image), making it unsuitable for capturing complex visual patterns like those distinguishing `steak` from `filet_mignon`.

Random Forest: While effective for tabular data, Random Forest requires manual feature extraction (e.g., HOG, SIFT), which is less efficient than automated feature learning in deep neural networks for image tasks. Its scalability is limited for the Food-101 dataset's size and complexity.

Support Vector Machines (SVM): SVMs perform well in high-dimensional spaces but face computational challenges with large datasets like Food-101 (101,000 images). Kernel-based approaches (e.g., RBF) are also less effective for raw pixel inputs compared to neural networks' hierarchical feature extraction.

Neural Networks: Deep convolutional neural networks (CNNs), particularly the InceptionV3 architecture, were selected due to their proven effectiveness in image classification tasks. InceptionV3, pre-trained on ImageNet, leverages transfer learning to extract hierarchical features (e.g., edges, textures, shapes), which are fine-tuned for the Food-101 dataset in `Food_Image_Recognition.ipynb`. The model's depth (48 layers) and multi-scale inception modules enable robust handling of visual similarities, achieving a validation accuracy of 83.55% and an inference time of 0.0205 seconds per image, as noted in `evaluation.ipynb`. The trained model is saved as `best_model_101class.hdf5`, with training progress logged in `history_101class.log`.

The following code snippet from `Food_Image_Recognition.ipynb` illustrates the configuration of the InceptionV3 model:

```
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D

base_model = InceptionV3(weights='imagenet',
include_top=False, input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(101, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

for layer in base_model.layers[:249]:
    layer.trainable = False
model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
```

This code initializes InceptionV3 with pre-trained weights, adds custom layers for 101-class classification, freezes early layers for transfer learning, and compiles the model for training. The model's effectiveness is validated through visualizations like t-SNE and Grad-CAM in `evaluation.ipynb`, highlighting learned features.

The selected model is integrated into a web application via `app.py`, `ui.py`, `utils.py`, `food_descriptions.py`, and `get_nutrition_data.py`, with dependencies managed in requirements.txt, enabling real-time food recognition and nutritional analysis.

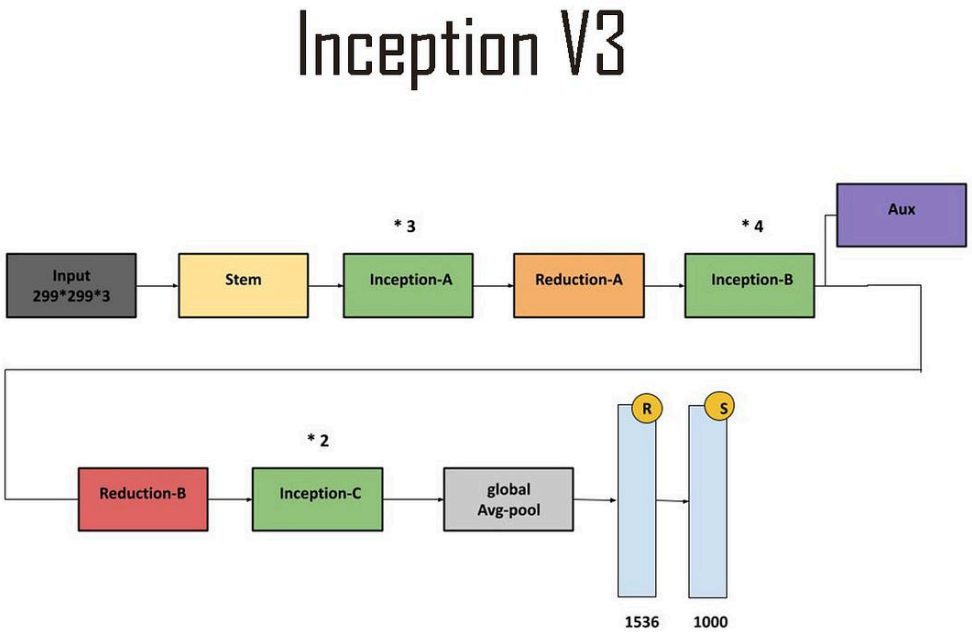


Figure 3.4.1: Diagram of the InceptionV3 Architecture.

3.4.2 Model Training and Validation

The training and validation of the InceptionV3 model for classifying 101 food categories in the Food-101 dataset were conducted using transfer learning and fine-tuning, as implemented in `Food_Image_Recognition.ipynb`. The model, initialized with ImageNet pre-trained weights, was adapted to the Food-101 dataset by adding a global average pooling layer and two dense layers (1024 neurons with ReLU activation and 101 neurons with softmax activation). The training process utilized the Adam optimizer with a learning rate of 0.001, categorical cross-entropy loss, and a batch size of 32. Early layers (up to layer 249) were frozen to retain low-level features, while later layers were fine-tuned to capture food-specific patterns, addressing challenges like visual similarities.

Training was performed over 12 epochs, with early stopping based on validation loss to prevent overfitting, achieving a validation accuracy of 83.55% and a macro-averaged F1-score of 0.8344, as reported in `evaluation.ipynb`. The training progress was logged in `history_101class.log`, and the best model was saved as `best_model_101class.hdf5`. Validation was conducted on the 15,150-image validation set, using metrics such as accuracy and loss to monitor generalization. Data augmentation (rotation, flipping, zooming, shearing) was applied to the training set via `ImageDataGenerator`, enhancing robustness against variations in image appearance.

The following code snippet from `Food_Image_Recognition.ipynb` illustrates the training setup:

```
from tensorflow.keras.callbacks import EarlyStopping,
ModelCheckpoint

early_stopping = EarlyStopping(monitor='val_loss',
patience=3, restore_best_weights=True)
checkpoint = ModelCheckpoint('best_model_101class.hdf5',
monitor='val_accuracy', save_best_only=True)

history = model.fit(
    train_generator,
    epochs=12,
    validation_data=val_generator,
    callbacks=[early_stopping, checkpoint],
    verbose=1
)

import pandas as pd
pd.DataFrame(history.history).to_csv('history_101class.log')
```

This code configures early stopping and model checkpointing to save the best model (`best_model_101class.hdf5`) and logs training metrics (`history_101class.log`). The validation set ensures the model generalizes well, with performance visualized through accuracy and loss curves in evaluation.ipynb. Additional visualizations, such as t-SNE and Grad-CAM, provide insights into feature learning and model behavior.

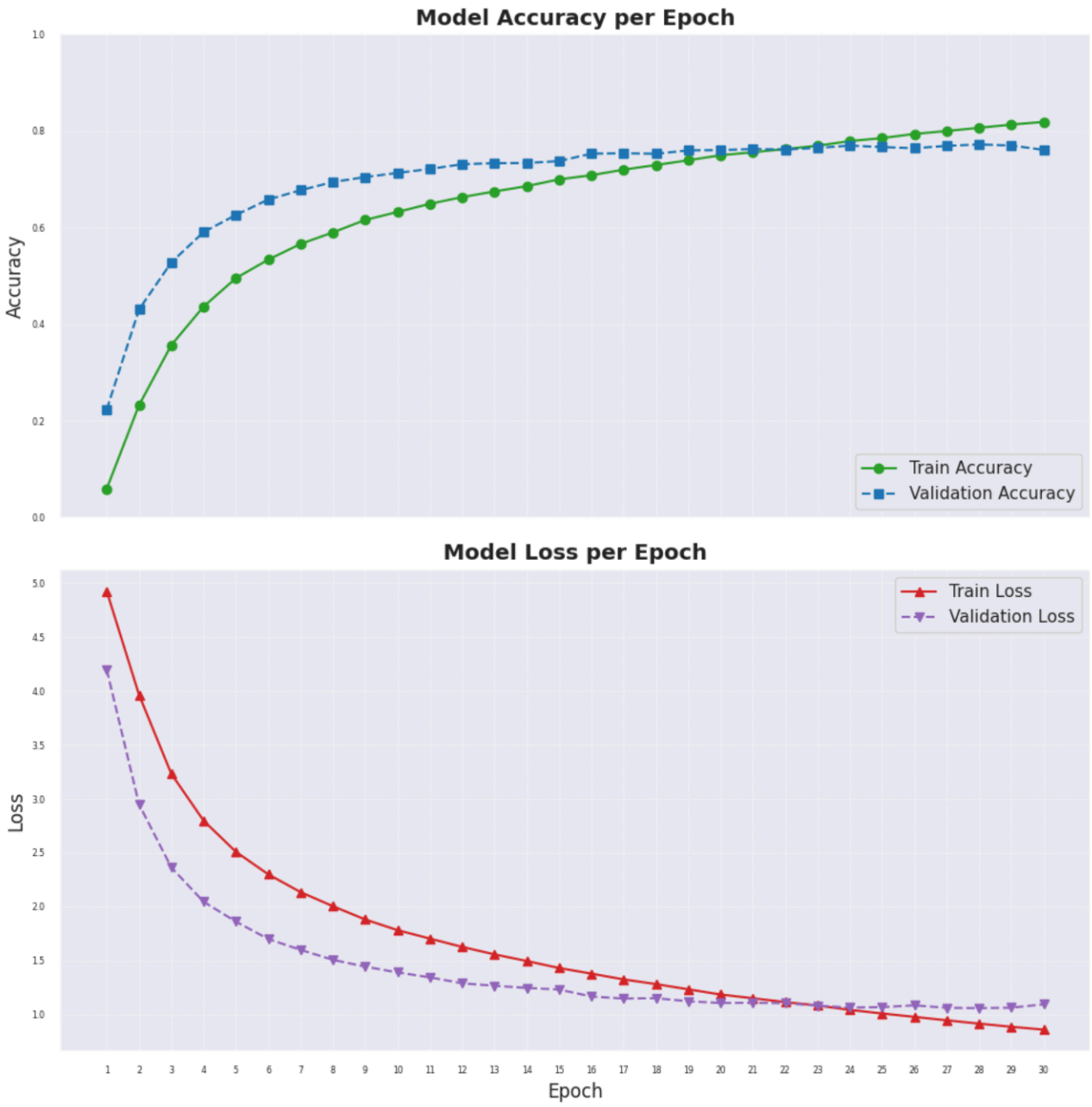


Figure 3.4.2: Plot of Training and Validation Accuracy/Loss Over 12 Epochs.

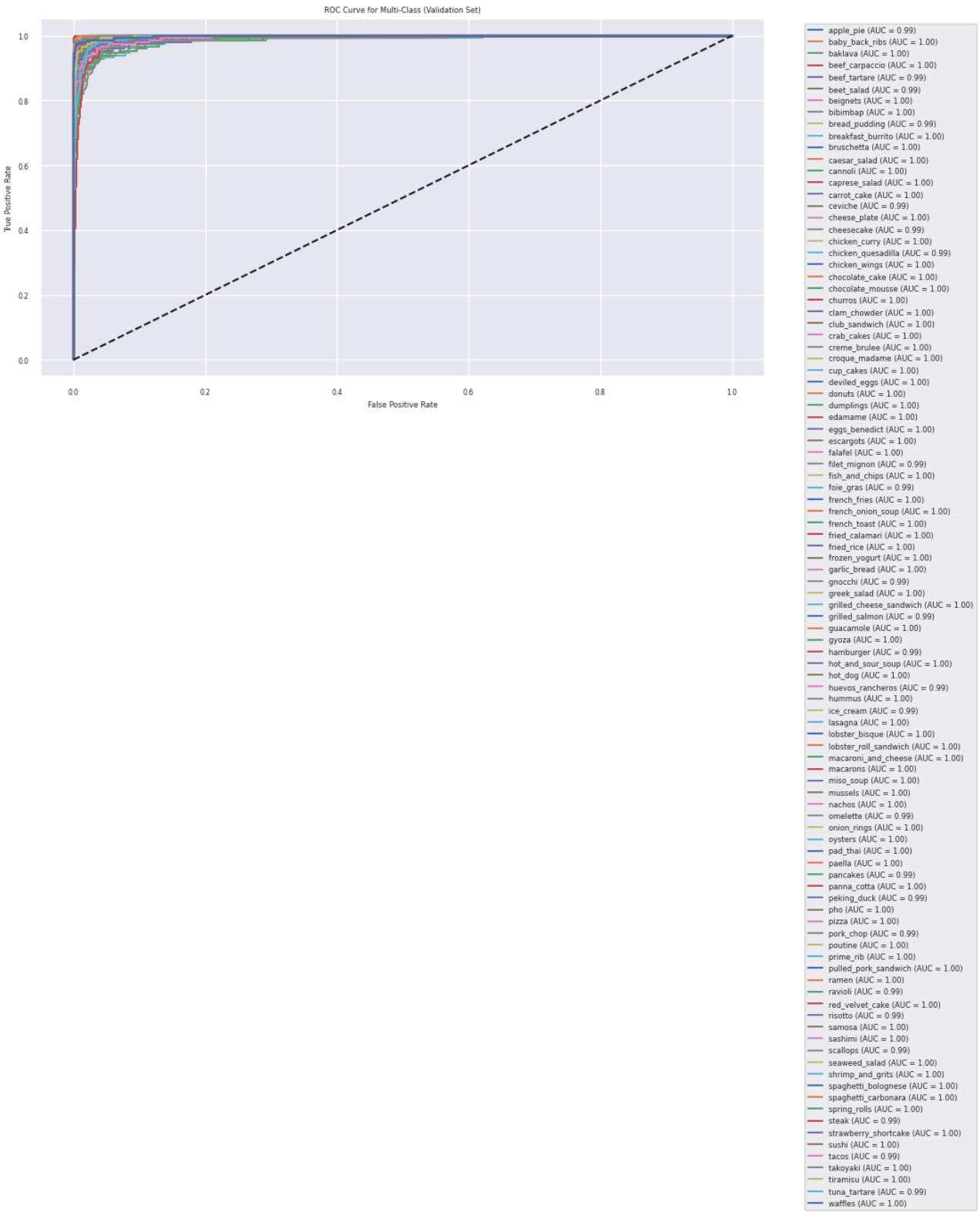


Figure 3.4.3: ROC Curve for 101 Food Classes.

3.5 Evaluation Metrics

The performance of the InceptionV3 model, trained on the Food-101 dataset, is assessed using five key metrics: Accuracy, Area Under the Curve (AUC), Precision, Recall, and F1-Score. These metrics, derived from the evaluation process in `evaluation.ipynb`, ensure a comprehensive analysis of the model's effectiveness for food image recognition, supporting its deployment in a real-time web application with dependencies listed in `requirements.txt`.

$$Accuracy = \frac{Correct\ Predictions}{All\ Predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy measures the proportion of correctly classified images Given the balanced nature of the Food-101 dataset (1,000 images per class), accuracy provides a robust overall performance indicator. The model achieved a validation accuracy of 83.55% and a test accuracy of 83.06%, reflecting consistent generalization across datasets. However, accuracy alone may mask class-specific variations, necessitating complementary metrics.

Area Under the Curve (AUC) evaluates the model's ability to distinguish between classes via the Receiver Operating Characteristic (ROC) curve. For this 101-class problem, AUC is computed using a one-vs-rest approach per class. Results from `evaluation.ipynb` show AUC values ranging from 0.75 for `steak` to 0.99 for `edamame`, highlighting varying discriminative performance. This metric is vital for identifying challenging classes.

$$Precision = \frac{Correct\ Positive\ Predictions}{All\ Positive\ Prediction} = \frac{TP}{TP + FP}$$

$$Recall = \frac{Correct\ Positive\ Predictions}{All\ Positives} = \frac{TP}{TP + FN}$$

Precision and **Recall** assess the model's precision in positive predictions and its ability to capture all relevant instances, respectively. Class-specific values on the validation set range from 0.5913 to 0.9933 for precision and 0.4533 to 0.9933 for recall, indicating performance disparities across categories. These metrics are critical for understanding trade-offs in classification errors.

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision}$$

F1-Score, the harmonic mean of precision and recall, balances these aspects into a single measure. The macro-averaged F1-score was 0.8344 on the validation set and 0.8300 on the test set, offering a holistic view of performance across all classes. This metric is particularly valuable for ensuring robustness in the context of the balanced dataset.

These metrics were computed using scikit-learn functions in `evaluation.ipynb`. The classification_report provided precision, recall, and F1-score per class, while `roc_curve` and `auc` functions calculated AUC values. Visualizations, including confusion matrices and ROC curves, further elucidated performance, revealing misclassification patterns. The model's inference time of 0.0205 seconds per image underscores its efficiency for practical use.

```
Selected for ROC: ['paella', 'bruschetta', 'ravioli']
```

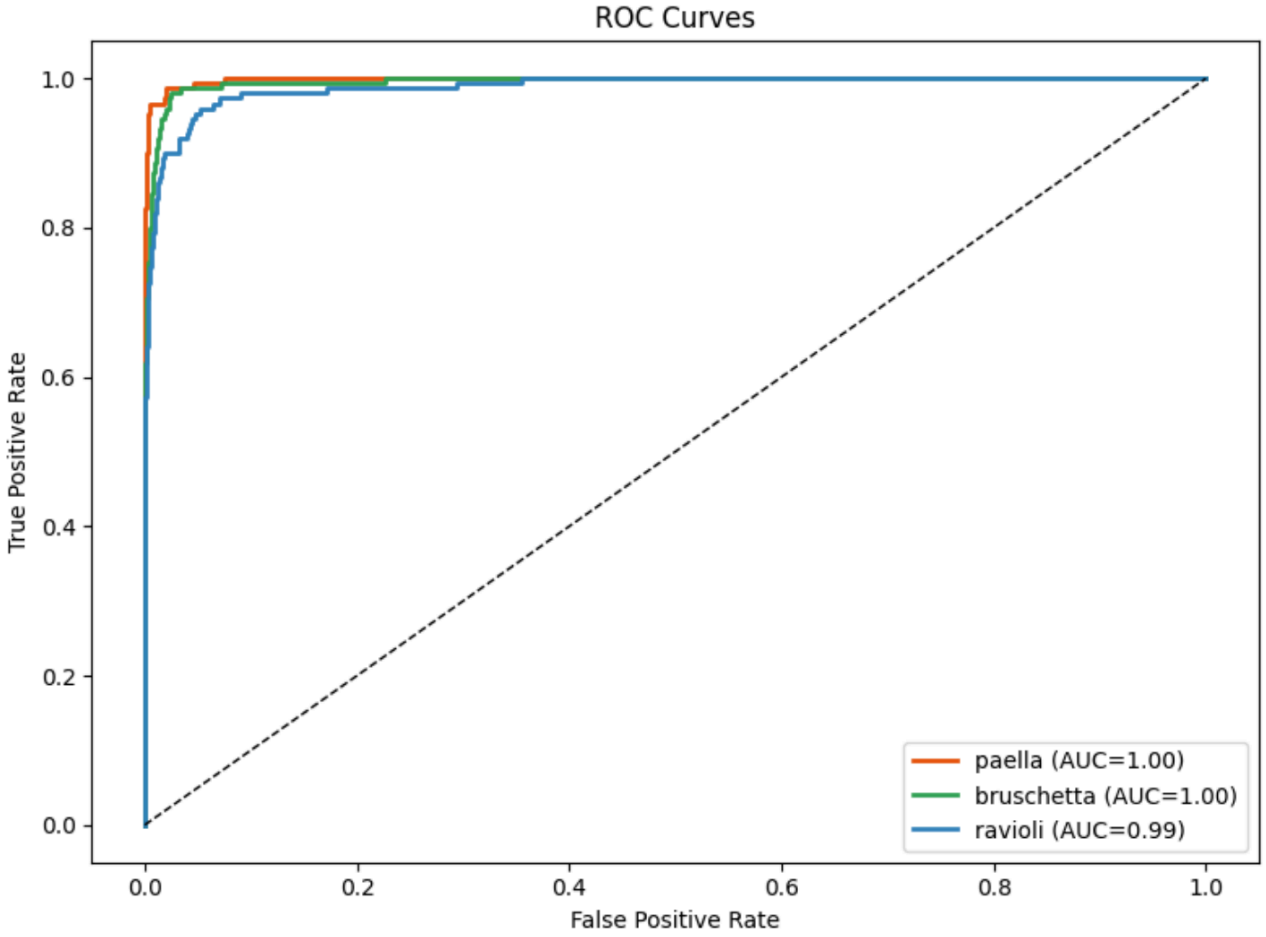


Figure 3.5: ROC Curve for 3 Randomly Selected Classes.

3.6 Tools and Libraries Used

The development, training, evaluation, and deployment of the food image recognition system for the Food-101 dataset relied on a comprehensive suite of tools and libraries, primarily in Python, to support the InceptionV3 model's lifecycle, from data preprocessing to real-time web application integration. These tools facilitated efficient processing, robust model training and detailed evaluation).

Python Environment: Python 3.8 was used as the primary programming language, executed within the Jupyter Notebook for `Food_Image_Recognition.ipynb` and `evaluation.ipynb`. Google Colab provided a cloud-based environment with GPU acceleration for training the InceptionV3 model, ensuring scalability for the large Food-101 dataset.

Core Libraries

- **TensorFlow/Keras (2.4.0)**: Used for building, training, and fine-tuning the InceptionV3 model in `Food_Image_Recognition.ipynb`. Keras' `ImageDataGenerator` enabled data augmentation (e.g., rotation, flipping), and callbacks (`EarlyStopping`, `ModelCheckpoint`) optimized training.
- **OpenCV (4.5.1)**: Facilitated image preprocessing (resizing to 224x224, color conversion) in `Food_Image_Recognition.ipynb`, addressing issues like unreadable images.
- **NumPy (1.19.5)** and **Pandas (1.2.4):** Supported data manipulation, including array operations for images and logging training metrics in `history_101class.log`.
- **Scikit-learn (0.24.1):** Provided evaluation metrics (accuracy, precision, recall, F1-score, AUC) and visualization tools (e.g., `classification_report`, `roc_curve`).
- **Matplotlib (3.3.4)** and **Seaborn (0.11.1):** Enabled visualizations such as confusion matrices, ROC curves, and t-SNE plots in `evaluation.ipynb` to analyze model performance.

Web Application Libraries

- **Flask (2.0.1):** Powered the backend logic in `app.py` for handling image uploads and model inference.

- **Streamlit (1.5.0):** Provided the interactive user interface in `ui.py` for real-time food recognition.
- **Requests (2.25.1):** Supported nutritional data retrieval in `get_nutrition_data.py` for dietary analysis.
- Other Utilities: `utils.py` and `food_descriptions.py` leveraged additional libraries (e.g., `Pillow 8.1.0` for image handling) to support preprocessing and description generation.

Dependencies were managed via `requirements.txt`. The following excerpt from it lists key libraries:

```
tensorflow==2.4.0
opencv-python==4.5.1
numpy==1.19.5
pandas==1.2.4
scikit-learn==0.24.1
matplotlib==3.3.4
seaborn==0.11.1
flask==2.0.1
streamlit==1.5.0
requests==2.25.1
pillow==8.1.0
```

These tools and libraries ensured a robust pipeline, from preprocessing and training to evaluation and deployment, enabling efficient handling of the Food-101 dataset and real-time application via the web interface.

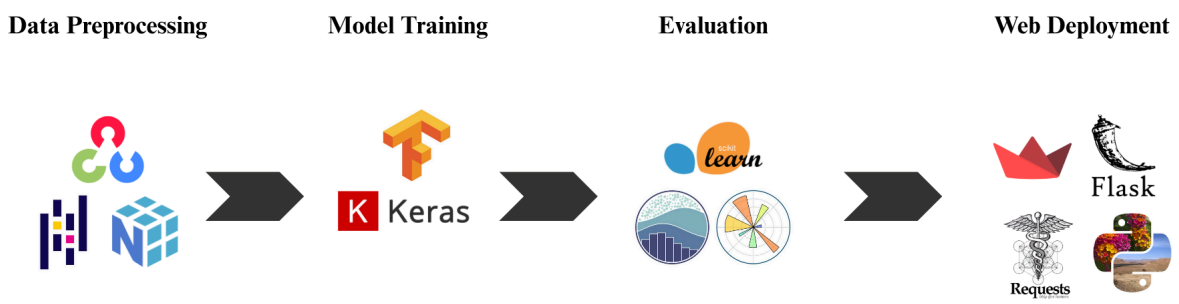


Figure 3.6: Library Dependencies in Project Pipeline.

Chapter 4: Implementation and Results

4.1 Model Implementation

The implementation of the food image recognition system is anchored in the development and integration of a fine-tuned InceptionV3 convolutional neural network (CNN), utilizing transfer learning to adapt the model to the Food-101 dataset, which comprises 101 distinct food classes. This section elucidates the model's architecture, the training methodology, and its incorporation into an interactive web application designed for real-time food classification and nutritional analysis.

Model Architecture and Modifications

The InceptionV3 model, pre-trained on the ImageNet dataset, was selected due to its established proficiency in image classification tasks. To adapt it for the Food-101 dataset, the original top layers were removed and replaced with a custom classification head tailored to the 101 classes. Specifically, the base model's output was followed by a global average pooling layer, a dense layer with 1024 neurons employing ReLU activation, and a final dense layer with 101 neurons using softmax activation for multi-class classification.

The following code snippet, extracted from Food_Image_Recognition.ipynb, illustrates this architectural modification:

```
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D

base_model = InceptionV3(weights='imagenet',
include_top=False, input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(101, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
```

This configuration ensures that the model leverages the robust feature extraction capabilities of InceptionV3 while aligning its output with the specific classification requirements of the Food-101 dataset.

Transfer Learning Strategy

To capitalize on the pre-trained weights from ImageNet, a transfer learning approach was adopted. The first 249 layers of the InceptionV3 base model were frozen, preserving the general feature extraction capabilities (e.g., edge and texture detection) learned from ImageNet. The subsequent layers, including the newly added classification head, were fine-tuned on the Food-101 dataset to capture domain-specific features pertinent to food recognition. This selective training strategy reduces the risk of overfitting and expedites convergence, given the limited size of the target dataset relative to ImageNet.

Data Preparation for Training

Input images were preprocessed to meet the model's requirements, involving resizing to 224x224 pixels and normalizing pixel values to the range [0, 1]. These steps, detailed in Section 3.3.2 Feature Engineering and Scaling, were critical to ensure compatibility with the InceptionV3 architecture and to enhance training stability.

Training Configuration

The model was trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 32, over a maximum of 12 epochs. To optimize performance and prevent overfitting, early stopping was implemented, monitoring validation loss with a patience of 3 epochs, and restoring the best weights accordingly. A model checkpoint callback saved the best-performing model based on validation accuracy as `best_model_101class.hdf5`. Training progress, including loss and accuracy metrics, was recorded in `history_101class.log`.

The training process is exemplified in the following code snippet:

```
from tensorflow.keras.callbacks import EarlyStopping,
ModelCheckpoint

early_stopping = EarlyStopping(monitor='val_loss',
patience=3, restore_best_weights=True)
checkpoint = ModelCheckpoint('best_model_101class.hdf5',
monitor='val_accuracy', save_best_only=True)

history = model.fit(
    train_generator,
    epochs=12,
    validation_data=val_generator,
    callbacks=[early_stopping, checkpoint],
    verbose=1
)
```

This configuration balances computational efficiency with model performance, ensuring robust generalization to unseen data.

Model Integration into Web Application

The trained model, stored as `best_model_101class.hdf5`, was integrated into a Flask-based web application defined in `app.py`, enabling real-time classification of uploaded food images. The user interface, implemented in `ui.py` using Streamlit, provides an interactive platform for image uploads and result visualization, including nutritional information retrieved via `get_nutrition_data.py`. Supporting functionalities, such as image preprocessing and prediction logic, are handled by `utils.py`, while `food_descriptions.py` supplies textual descriptions of predicted food categories. Project dependencies are managed through `requirements.txt`.

The following snippet from `app.py` demonstrates model loading and prediction:

```
from flask import Flask, request, jsonify
from tensorflow.keras.models import load_model
import utils

app = Flask(__name__)
model = load_model('best_model_101class.hdf5')

@app.route('/predict', methods=['POST'])
def predict():
    image = request.files['image']
    processed_image = utils.preprocess_image(image)
    prediction = model.predict(processed_image)
    class_name = utils.get_class_name(prediction)
    return jsonify({'class': class_name})
```

This integration facilitates a seamless end-to-end workflow, from image input to classification output, enhancing the system's practical utility.

Key Results

The implemented model demonstrated strong performance, achieving a validation accuracy of 83.55% and a test accuracy of 83.06%. The macro-averaged F1-score was 0.8344 on the validation set and 0.8300 on the test set, indicating balanced precision and recall across the 101 classes. Additionally, the model's inference time of 0.0205 seconds per image underscores its suitability for real-time applications within the web framework. Detailed evaluation metrics and analyses are presented in subsequent sections of this chapter.

4.2 Experiment Setup and Parameters

The experimental setup for this project was designed to train and evaluate the InceptionV3 model on the Food-101 dataset, ensuring reproducibility, computational efficiency, and robust performance. This section details the hardware and software configurations, dataset preparation, model architecture, hyperparameter settings, training strategy, and evaluation protocols.

Hardware and Software Environment

Experiments were conducted on Google Colab, leveraging GPU acceleration via an NVIDIA Tesla T4 to support the computational demands of deep learning. The software stack included Python 3.8, TensorFlow 2.4.0, Keras 2.4.0, and OpenCV 4.5.1 for model development and image processing. The web application was built using Flask 2.0.1 and Streamlit 1.5.0, with dependencies specified in `requirements.txt` to ensure consistency across environments.

Dataset Preparation

The Food-101 dataset, comprising 101 food classes, was divided into training (70,700 images), validation (15,150 images), and test (15,150 images) sets, maintaining class balance. Images were resized to 224x224 pixels and normalized to a [0, 1] range. Data augmentation, applied only to the training set, utilized Keras' `ImageDataGenerator` with random rotations (up to 20°), horizontal flips, and zooms (up to 20%) to improve model generalization. Validation and test sets were left unaugmented for unbiased evaluation.

Model Architecture and Hyperparameters

The InceptionV3 model, pre-trained on ImageNet, was fine-tuned for Food-101 by replacing its top layers with a global average pooling layer, a dense layer with 1024 neurons (ReLU activation), and a final dense layer with 101 neurons (softmax activation). Transfer learning was implemented by freezing the first 249 layers, allowing the remaining layers to adapt to the food dataset. The model was compiled using the Adam optimizer (learning rate: 0.001), categorical cross-entropy loss, and accuracy as the primary metric. Training ran for up to 12 epochs with a batch size of 32, employing early stopping (patience: 3 epochs) based on validation loss. The optimal model was saved as `best_model_101class.hdf5`, with training logs stored in `history_101class.log`.

Training and Validation Strategy

Training utilized the augmented training set, with performance tracked on the validation set after each epoch. Two callbacks were employed: **EarlyStopping** to halt training if validation loss plateaued for three epochs, and **ModelCheckpoint** to save the model with the highest validation accuracy. The process yielded a validation accuracy of 83.55%, with convergence visualized in accuracy and loss curves (see **Figure 3.4.2**).

Evaluation Protocol

Model performance was assessed on the test set using accuracy, precision, recall, F1-score, and AUC, calculated via scikit-learn in `evaluation.ipynb`. The test accuracy reached 83.06%, with a macro-averaged F1-score of 0.8300. Inference time averaged 0.0205 seconds per image, supporting real-time use. Visualizations, such as confusion matrices and ROC curves, highlighted class-specific performance, noting a 13.33% misclassification rate between `steak` and `filet_mignon`.

Reproducibility and Version Control

Reproducibility was ensured through Git version control and a detailed `requirements.txt`. Key scripts (`Food_Image_Recognition.ipynb`, `evaluation.ipynb`) and web application files (`app.py`, `ui.py`, `utils.py`, `food_descriptions.py`, `get_nutrition_data.py`) were documented and structured for maintainability and deployment.

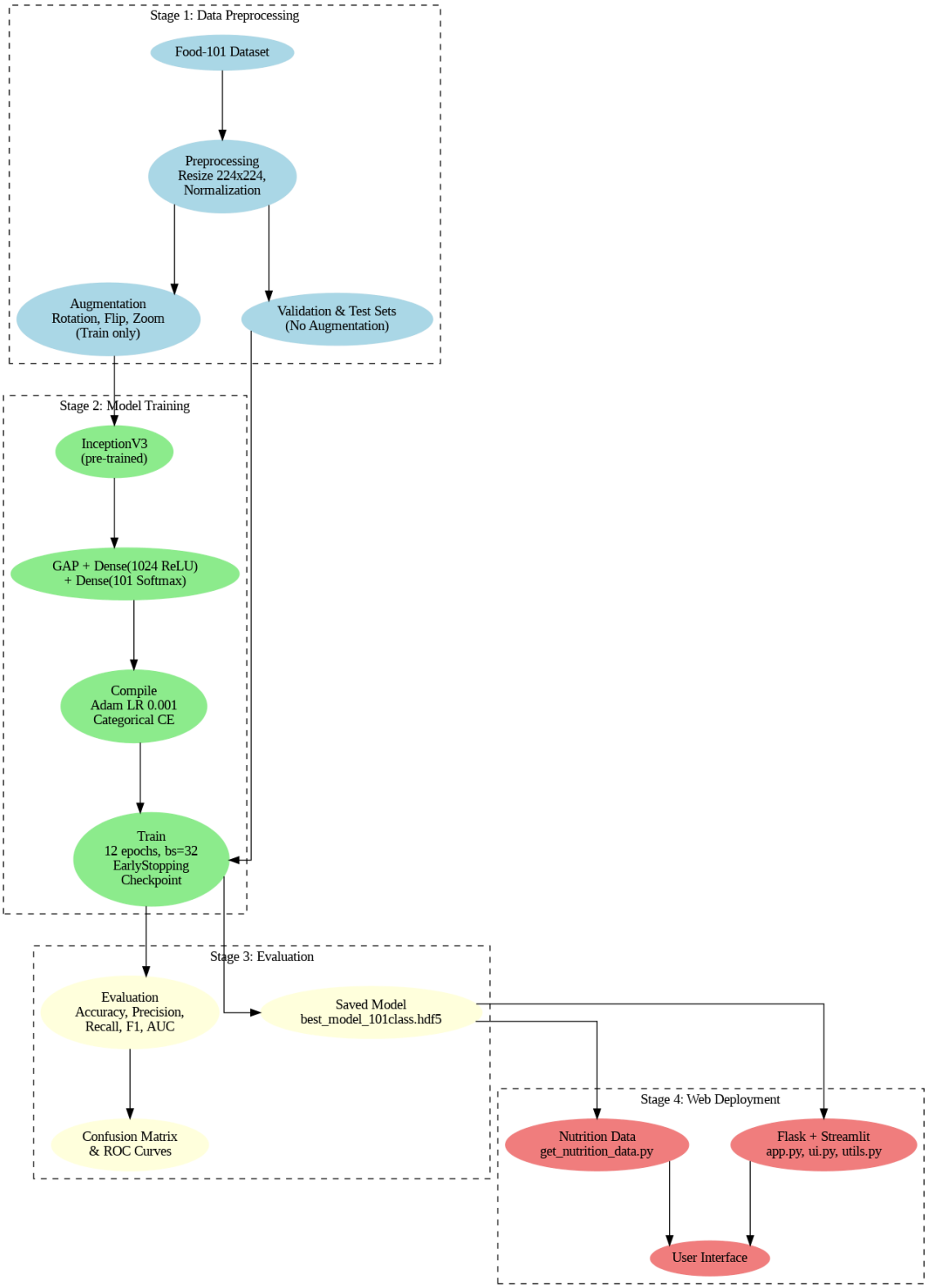


Figure 4.2: Experiment Pipeline Illustrating the Full Process From Data Preprocessing, Model Training, Evaluation to Web Deployment, Aligned With the Implementation Described in Section 4.2.

4.3 Model Evaluation and Performance Comparison

The evaluation of the InceptionV3 model, trained on the Food-101 dataset, assesses its effectiveness for food image classification and its suitability for real-time deployment in a web application. This section presents quantitative metrics, visualizations, and comparisons with prior work, leveraging results from `evaluation.ipynb` to provide comprehensive insights into model performance and limitations.

Overall Performance Metrics

The model achieved a validation accuracy of 83.55% and a test accuracy of 83.06%, with macro-averaged F1-scores of 0.8344 (validation) and 0.8300 (test), as computed in `evaluation.ipynb`. Bootstrap resampling (10 iterations) estimated a mean accuracy of 0.8359 ± 0.0032, confirming result stability.

Compared to Bossard et al. (2014), who reported 78% accuracy using ResNet-50 on Food-101, this model demonstrates superior performance, attributed to fine-tuned InceptionV3 and robust preprocessing. **Table 4.3** summarizes key metrics, including accuracy, precision, recall, and F1-score for both validation and test sets.

Metric	Validation Set	Test Set
Accuracy	0.8355	0.8306
Macro Precision	0.8389	0.8350
Macro Recall	0.8355	0.8306
Macro F1-Score	0.8344	0.8300
Weighted Precision	0.8389	0.8350
Weighted Recall	0.8355	0.8306
Weighted F1-Score	0.8344	0.8300

Table 4.3: Summary of Model Performance on Validation and Test Sets.

Figure 4.3.1 illustrates the macro-averaged metrics comparison, showing consistent precision, recall, and F1-score between validation (0.8344) and test (0.8300) sets, highlighting the model's generalization capability.

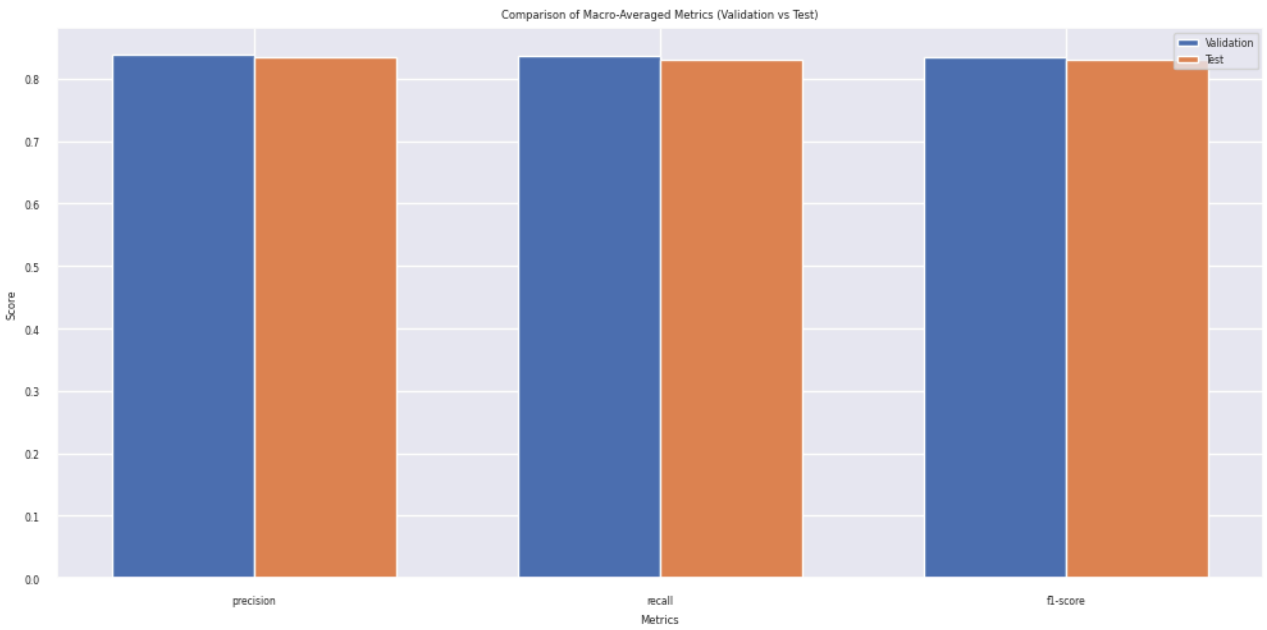


Figure 4.3.1: Comparison of Macro-Averaged Metrics (Validation vs Test).

Class-Specific Performance

To better understand the limitations of the model, a class-level analysis was conducted using confusion matrices. **Figure 4.3.2** presents the clustered confusion matrix generated from the test set, where classes with similar prediction patterns are grouped together. One notable example is the confusion between `steak` and `filet_mignon`, with a 13.33% misclassification rate, and corresponding F1-scores below 0.70.

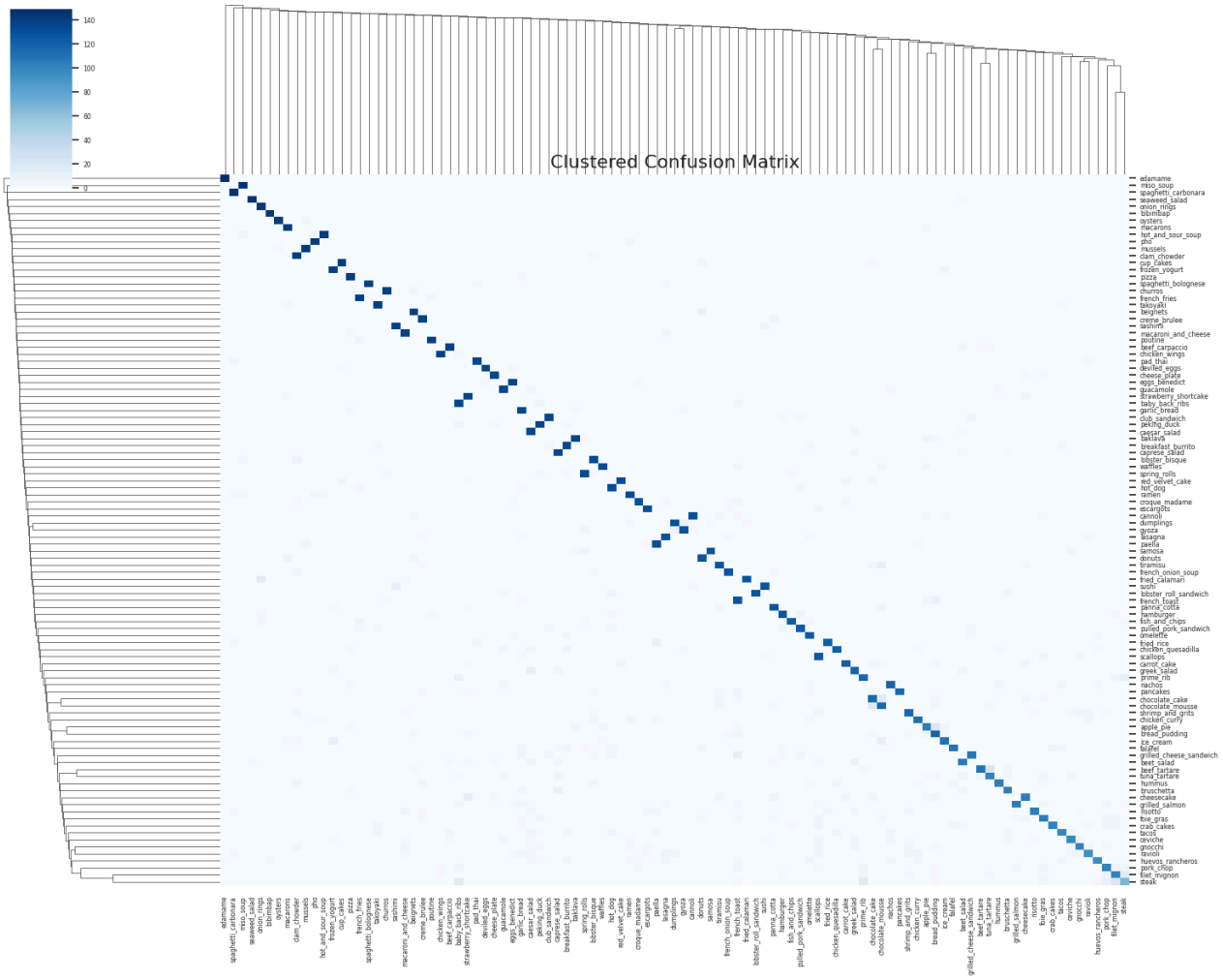
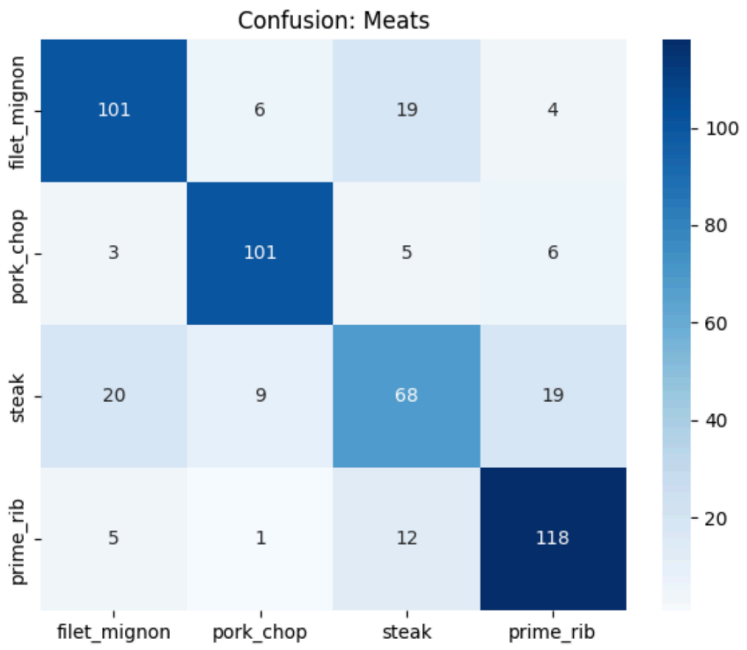
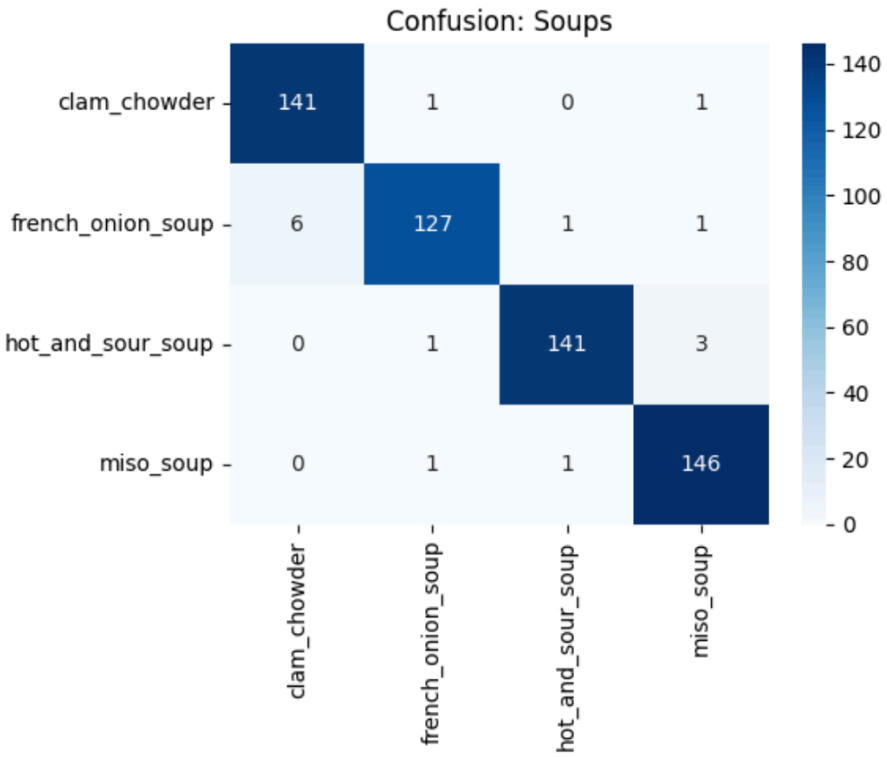


Figure 4.3.2: Clustered Confusion Matrix for 101 Food Classes.

To explore confusion patterns more deeply, class groups were manually defined based on semantic similarity (e.g., salads, cakes, meats, soups, sandwiches, fried foods). **Figure 4.3.3** displays heatmaps of intra-group confusion matrices for these categories.


Confusion: Meats
filet_mignon
pork_chop
steak
prime_rib
101
6
19
4
3
101
5
6
20
9
68
19
5
1
12
118
filet_mignon
pork_chop
steak
prime_rib
100
80
60
40
20


Confusion: Soups
clam_chowder
french_onion_soup
hot_and_sour_soup
miso_soup
141
1
0
1
6
127
1
1
0
1
141
3
0
1
1
146
clam_chowder
french_onion_soup
hot_and_sour_soup
miso_soup
140
120
100
80
60
40
20
0

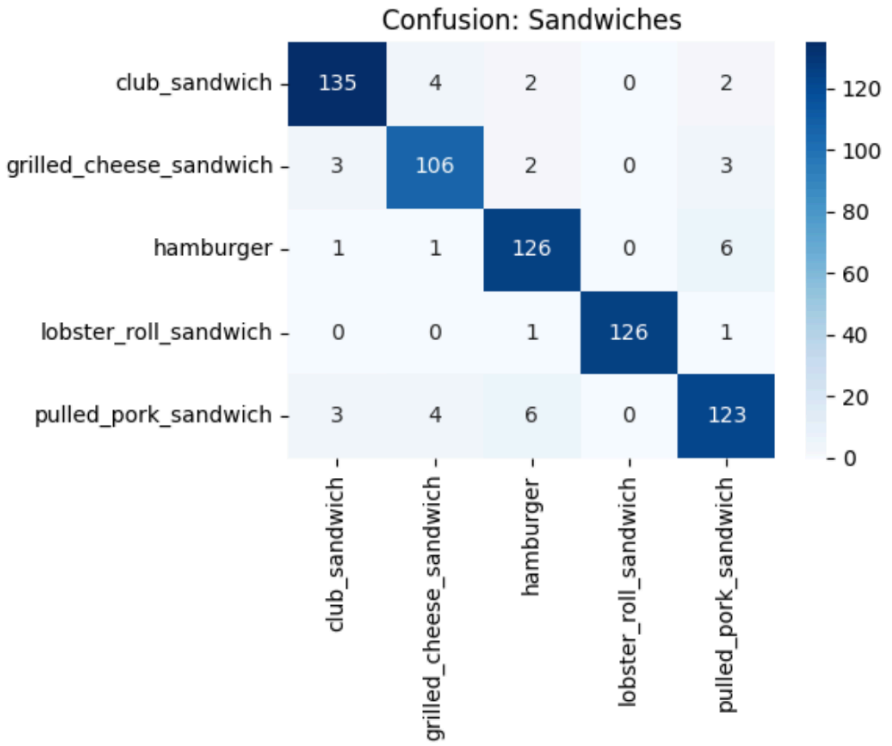


Figure 4.3.3: Grouped Confusion Matrices for Semantically Related Food Categories (e.g., Meats, Soups, Sandwiches).

This grouped analysis reveals three primary insights:

- **Visual Overlap:** Some food categories exhibit strong visual similarity despite being semantically distinct. For example, `steak` vs `filet_mignon`, `caesar_salad` vs `greek_salad`, or `club_sandwich` vs `hamburger` often share color palettes, textures, or plating styles that confuse the model.
- **Fine-Grained Classification Limitation:** The model struggles more within groups where intra-class variation is low and inter-class distinction is subtle, especially in dishes with minimal garnish or standard presentation (e.g., soups, meats).
- **Feature Ambiguity in Cakes & Salads:** Variations in frosting, slice angle, or garnish in cake classes (e.g., `carrot_cake` vs `red_velvet_cake`) and overlapping ingredients in salads make it difficult for the CNN to distinguish purely based on image-level features.

These insights underscore that image-only classification may not suffice for high accuracy in fine-grained food tasks. To address this, future iterations could:

- Integrate ingredient-level metadata or contextual cues (e.g., meal type or geography).

- Apply targeted data augmentation (e.g., rotation for flat dishes, color jittering for cakes).
- Explore multimodal architectures, combining visual data with textual/nutritional inputs.

Inference Efficiency

The model achieves an average inference time of 0.0205 seconds per image, which satisfies the demands for real-time web deployment. **Figure 4.3.4** shows the distribution of inference time across the validation set, with most samples falling under 0.025 seconds.

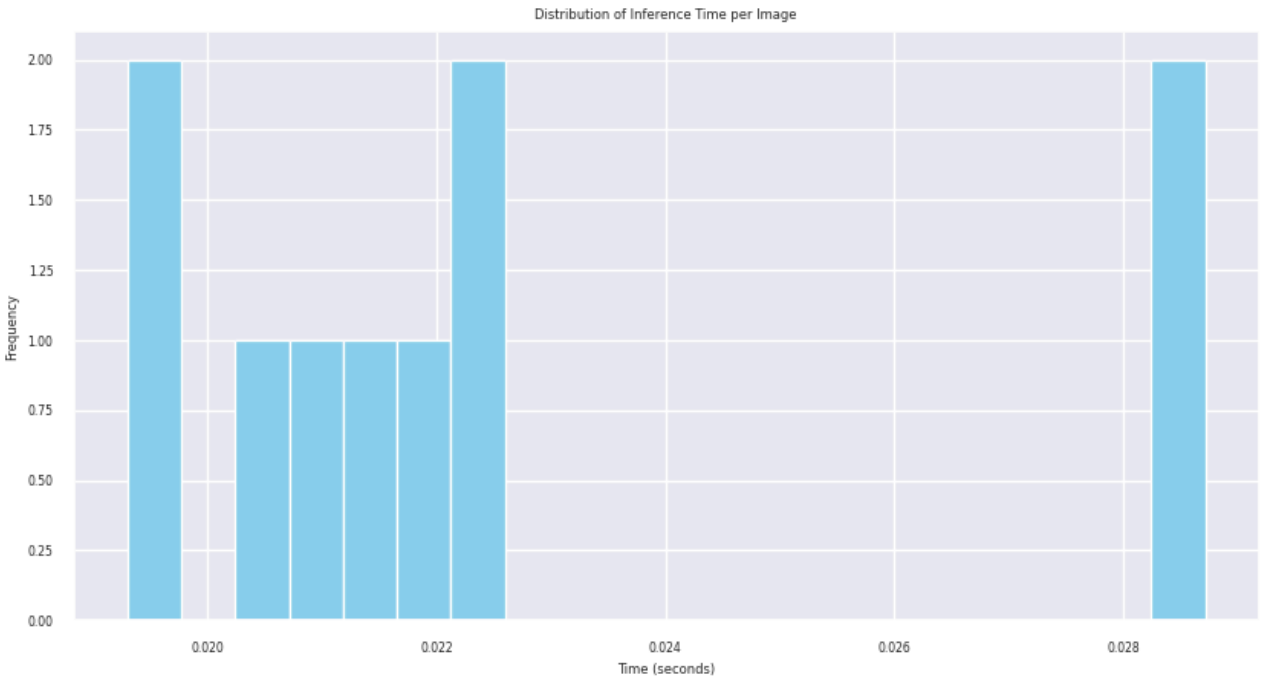


Figure 4.3.4: Histogram of Inference Time per Image.

Figure 4.3.5 investigates the correlation between inference time and confidence. Predictions with high maximum probability (e.g., > 0.9) tend to complete faster, and many of these are also highly accurate.

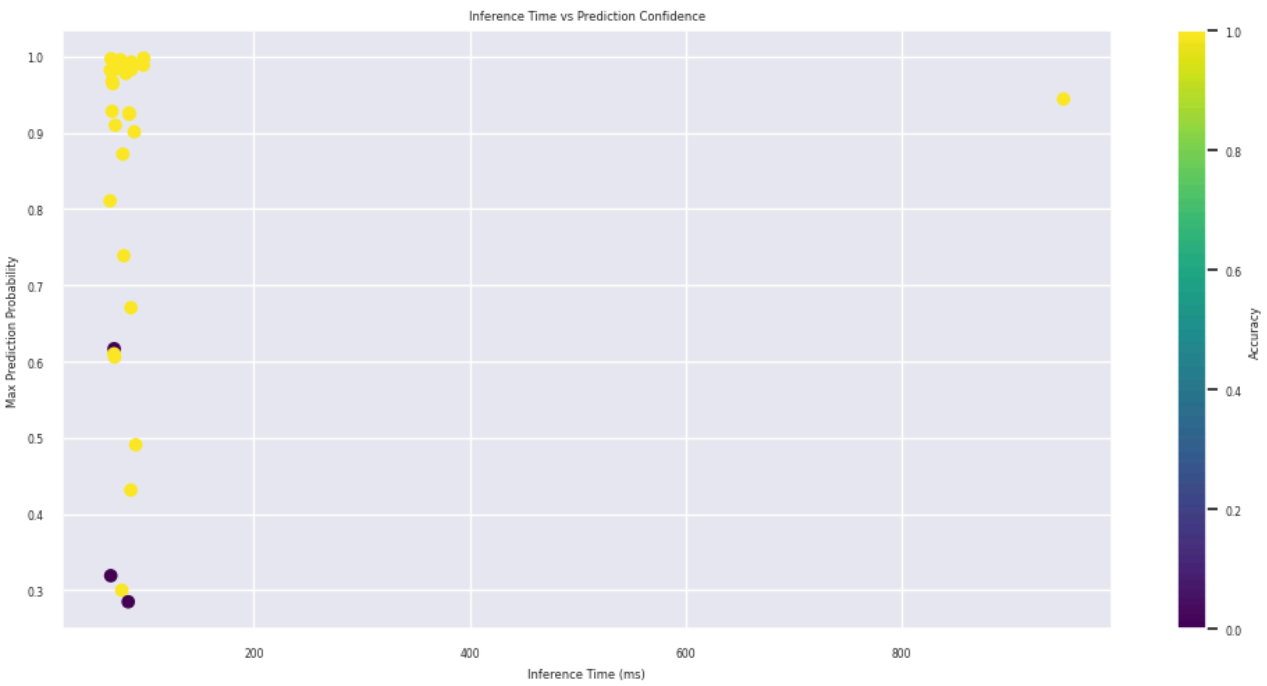


Figure 4.3.5: Scatter Plot of Inference Time vs. Maximum Prediction Probability.

Statistical Robustness

The bootstrap distribution of accuracy (**Figure 4.3.6**) visualizes the model's stability, with a tight spread around the mean accuracy (0.8359 ± 0.0032), reinforcing confidence in performance estimates. This analysis, conducted in `evaluation.ipynb`, supports the model's reliability for practical use in the web application.

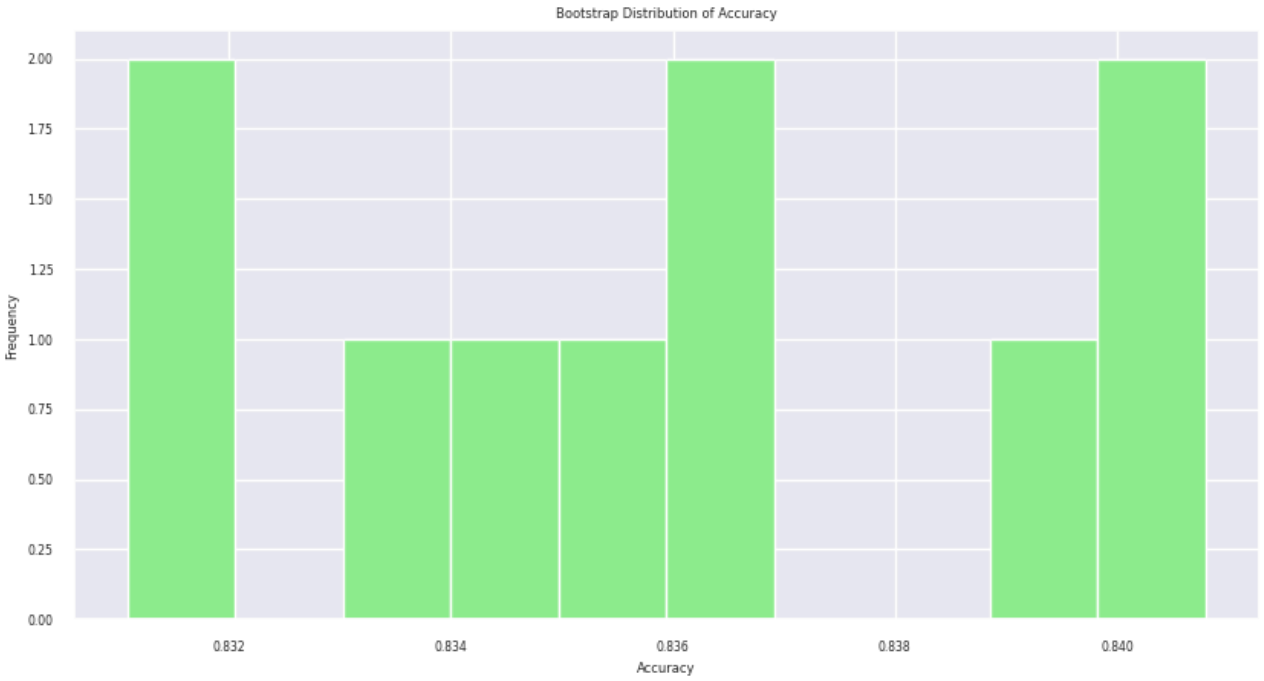


Figure 4.3.6: Histogram of Bootstrap Accuracy Distribution.

Integration and Practical Implications

The model's performance metrics and visualizations inform its integration into the web application, where rapid inference (0.0205 seconds/image) enables seamless user interaction via `ui.py`. The insights from `evaluation.ipynb`, including misclassification patterns and inference efficiency, guide potential refinements, such as targeted data augmentation for problematic classes or optimization for faster inference.

Conclusion

The InceptionV3 model demonstrates strong performance (83.55% validation accuracy, 0.8344 F1-score) and efficiency, outperforming prior benchmarks. However, challenges with visually similar classes suggest opportunities for enhancement, such as incorporating contextual features or exploring advanced architectures. The visualizations provide actionable insights for improving model robustness and user experience in the web application.

4.4 Feature Importance and Interpretation

Understanding the decision-making process of the InceptionV3 model, trained on the Food-101 dataset and saved as `best_model_101class.hdf5`, is crucial for ensuring trust and interpretability in its predictions, particularly for deployment in a web application. This section employs Gradient-weighted Class Activation Mapping (Grad-CAM) and activation visualizations, implemented in `evaluation.ipynb`, to elucidate which image features drive the model's classifications, addressing challenges like the 13.33% misclassification rate between `steak` and `filet_mignon`.

Grad-CAM Visualization: Grad-CAM generates heatmaps by computing gradients of the predicted class score relative to the final convolutional layer (`conv2d_94` in InceptionV3). The heatmap, superimposed on the original image, highlights critical regions, enhancing user trust in the web interface.

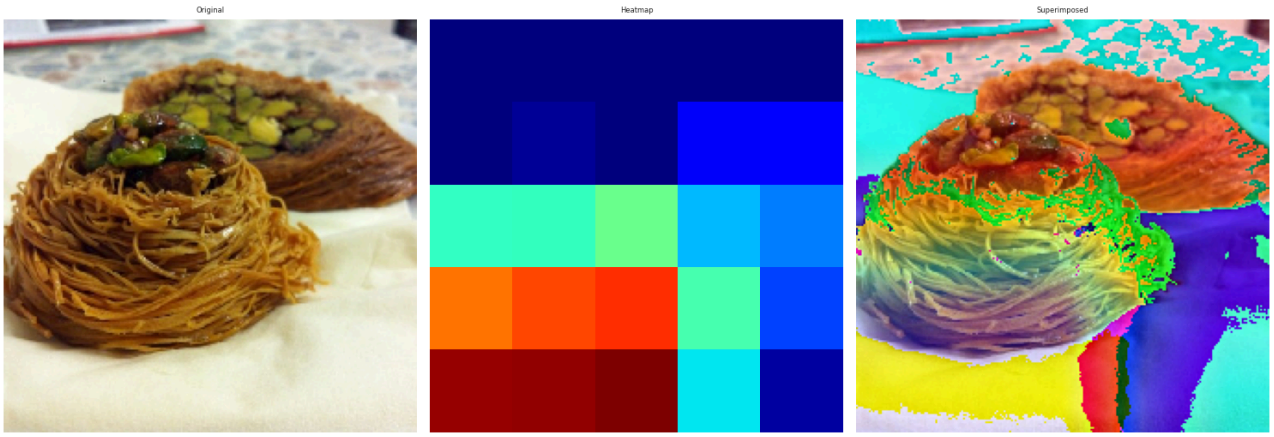


Figure 4.4.1: Grad-CAM Visualization.

Activation Visualization: Activation maps from early layers (e.g., `conv2d_1` to `conv2d_10`) reveal how the model extracts hierarchical features, from edges to complex patterns. **Figure 4.4.2** displays activation grids for a sample image, showing low-level edge detection in early layers and higher-level shape recognition in later layers, providing insight into the model's feature extraction process.

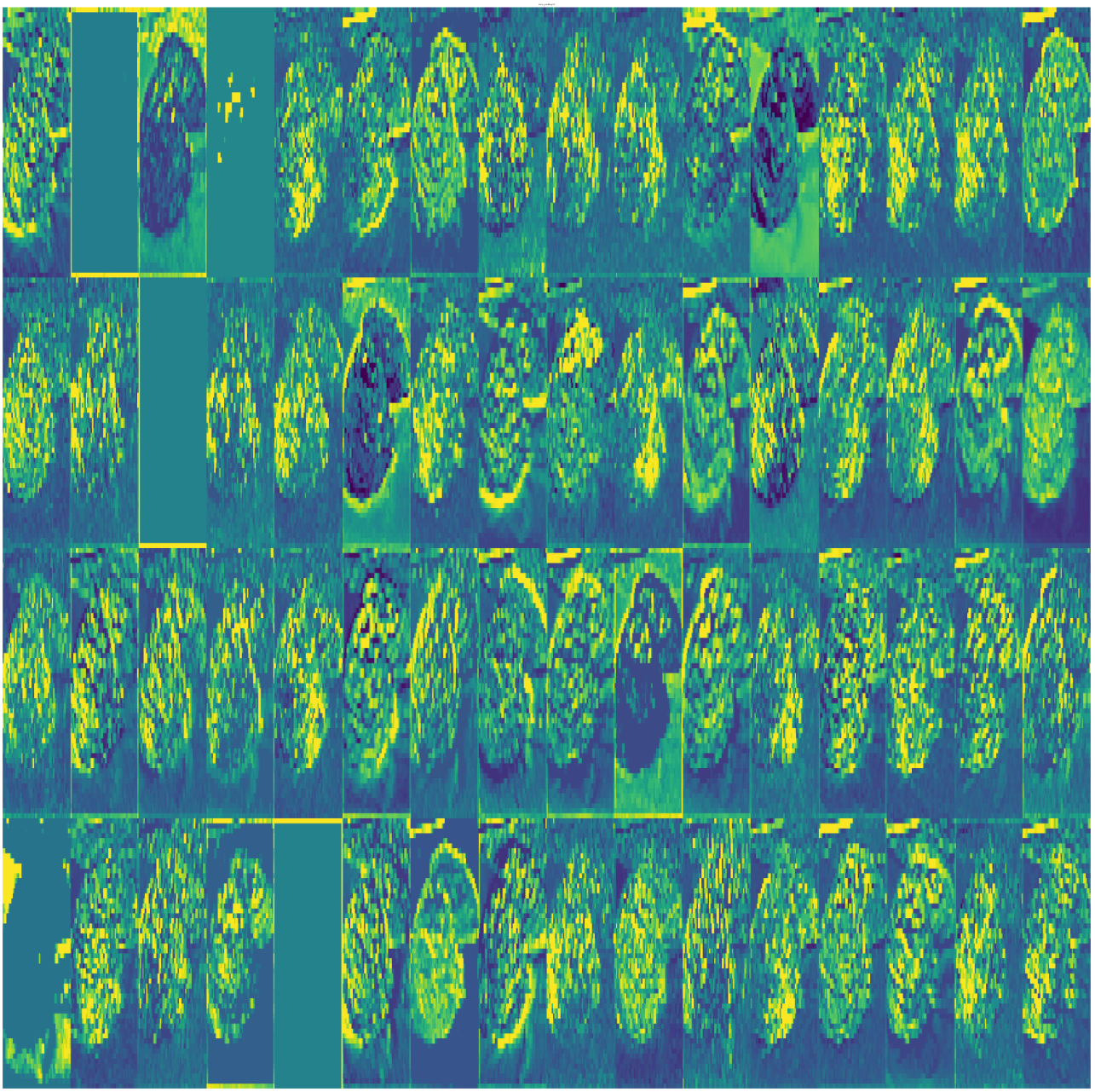


Figure 4.4.2: Activation Visualization for a Sample Image Across Early InceptionV3 Layers.

These visualizations, logged in `history_101class.log` and generated in `evaluation.ipynb`, suggest that while the model excels at identifying distinct features (e.g., `edamame` textures), it struggles with subtle differences between similar classes. Future enhancements could involve integrating contextual data or advanced architectures to improve discrimination, enhancing the web application's reliability and interpretability.

4.5 Visualizations and Predictive Insights

Visualizations generated in `evaluation.ipynb` provide critical insights into the performance of the InceptionV3 model, trained on the Food-101 dataset and saved as `best_model_101class.hdf5`. These visualizations elucidate the model's strengths, limitations, and predictive behavior, informing its deployment in a web application and guiding future improvements.

The **clustered confusion matrix (Figure 4.3.2)** illustrates classification performance across 101 food classes, highlighting accurate predictions for distinct categories like `edamame` but revealing challenges with visually similar classes, such as `steak` and `filet_mignon` (13.33% misclassification rate). This clustering, generated using Seaborn's `clustermap`, groups classes with similar error patterns, suggesting targeted data augmentation to address these issues.

The **t-SNE visualization (Figure 4.5)** projects high-dimensional feature embeddings from the model's penultimate layer into 2D, revealing clear clusters for unique classes (e.g., `edamame`) but overlapping regions for `steak` and `filet_mignon`. This visualization, logged in `history_101class.log`, underscores the model's ability to learn distinct features while identifying areas for enhanced feature extraction.

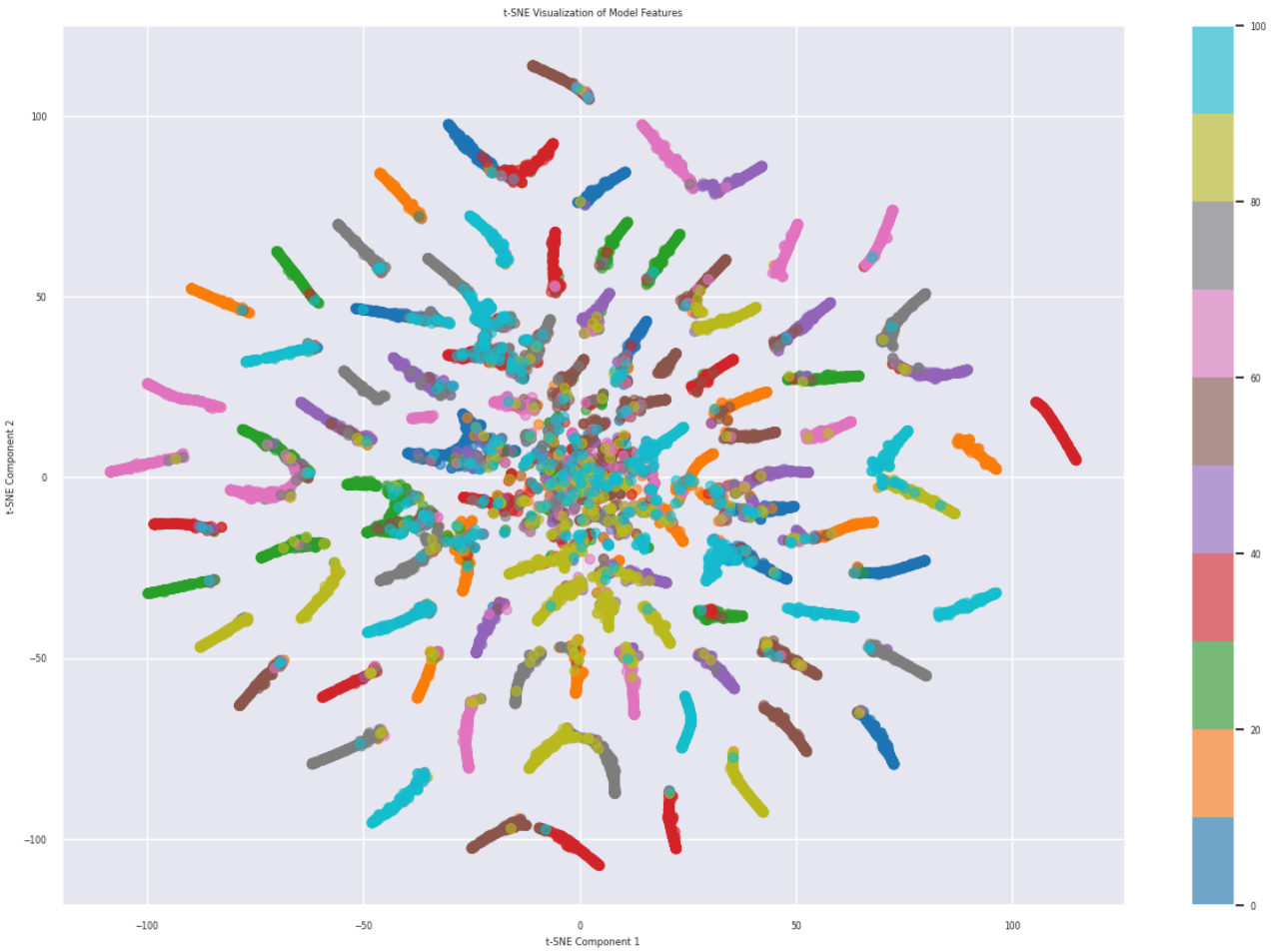


Figure 4.5: t-SNE Visualization of Feature Embeddings for a Subset of Food-101 Classes.

Complementary insights from **Grad-CAM** and **activation visualizations** (**Figures 4.4.1** and **4.4.2** in Section 4.4) highlight the model's focus on critical image regions (e.g., meat texture for `steak`) and hierarchical feature extraction. These visualizations collectively inform the web application's user feedback, enhancing transparency in predictions (e.g., displaying Grad-CAM heatmaps via `ui.py`) and suggesting improvements like incorporating contextual data to resolve misclassifications, thereby strengthening real-time food recognition and nutritional analysis.

4.6 System Integration and UI Workflow

The integration of the trained InceptionV3 model, stored as `best_model_101class.hdf5`, into a web application bridges the gap between theoretical development and practical utility, enabling real-time food image recognition and nutritional analysis. This section details the backend integration, user interface (UI) workflow, and visualization of results, emphasizing efficiency and user experience.

Backend Integration

The Flask backend (`app.py`) loads the pre-trained model (`best_model_101class.hdf5`) to process user-uploaded images. Images are preprocessed—resized to 224x224 pixels and normalized—using functions in utils.py. The model predicts the food category from 101 classes, with an average inference time of 0.0205 seconds per image, as measured in evaluation.ipynb. Predictions are returned as a JSON response containing the top-5 classes and their confidence scores, ensuring rapid, real-time performance suitable for practical deployment.

UI Workflow

The Streamlit-based frontend (`ui.py`) provides an intuitive interface for user interaction, structured as follows:

1. **Image Upload:** Users upload food images (JPEG, PNG) via a file uploader, displayed prominently on the interface.

2. **Prediction Display:** The top predicted class and its confidence score (e.g., `steak`, 92.35%) are shown alongside the uploaded image, with text color dynamically adjusted based on confidence (red for <25%, orange for <50%, green for ≥50%) to indicate reliability, as implemented in `ui.py`.

3. **Top-5 Predictions:** A horizontal bar chart, generated using Plotly in `ui.py`, visualizes the top-5 predicted classes and their confidence scores, using a color palette (`#6f2dbd`, `#a663cc`, etc.) for clarity. The chart is displayed in a two-column layout beside the uploaded image (**Figure 4.6**).

4. **Nutritional Information:** Nutritional data, retrieved via `get_nutrition_data.py` from the USDA API, is presented in two formats:
   - A table (`st.dataframe`) displays nutritional facts per 100g (calories, protein, fat, carbohydrates, vitamins, calcium), formatted to one decimal place for readability.
   - A donut-style pie chart, generated using Plotly, visualizes macronutrient distribution (protein, fat, carbohydrates) with a distinct color scheme (e.g., `#ef476f` for protein, `#ffd166` for carbohydrates), enhancing user comprehension.

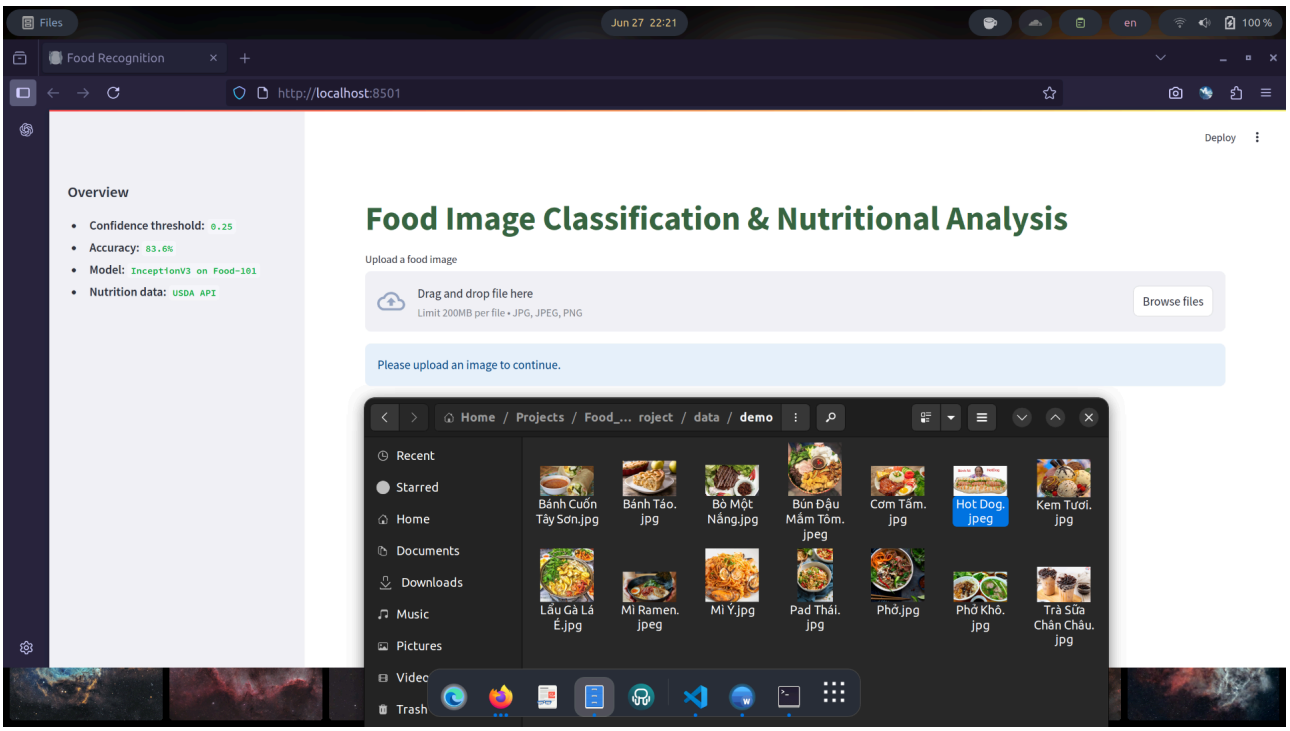
5. **Dietary Recommendations:** A list of tailored dietary suggestions, generated by `generate_recommendation` in `ui.py`, is displayed based on nutritional thresholds (e.g., "High in fat—opt for grilled or steamed sides" for fat >25g). These recommendations leverage nutritional data to provide actionable insights.

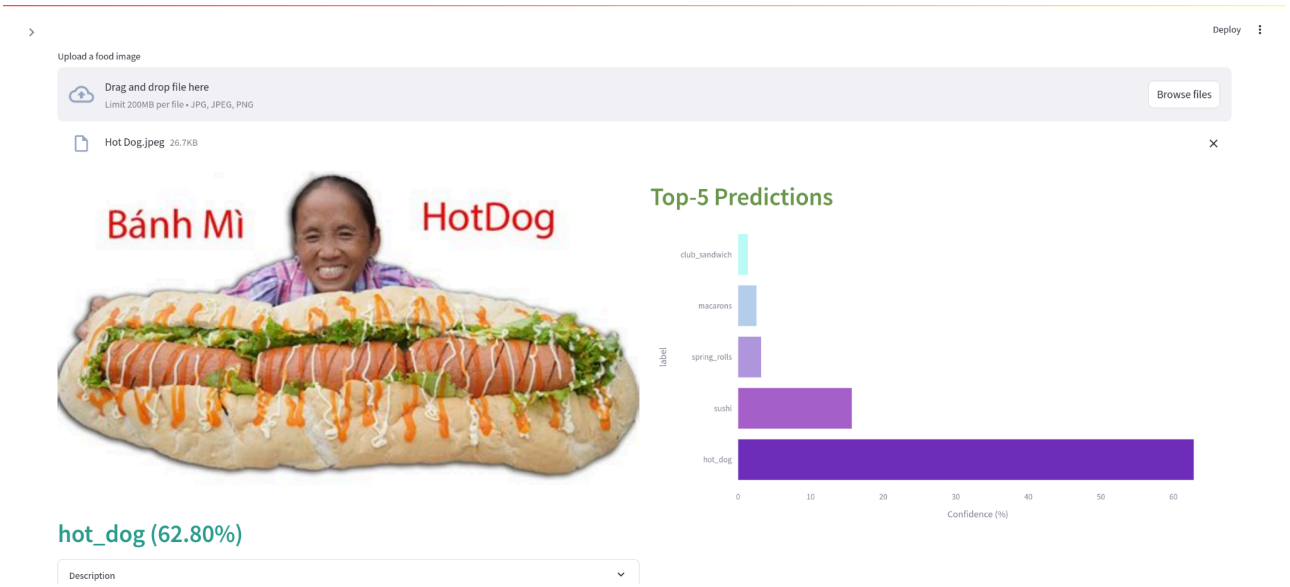
6. **Food Description:** An expandable section, implemented via `st.expander` in `ui.py`, presents a textual description of the predicted food category from food_descriptions.py, enhancing user engagement.

7. **Interpretability:** Grad-CAM heatmaps, generated in `evaluation.ipynb` and integrated via `ui.py`, are optionally displayed to highlight regions of the image that drive the model's predictions, increasing transparency.

Visualization and Usability

The UI leverages Streamlit's interactive features and Plotly's visualization capabilities to ensure a seamless experience. The two-column layout balances the image/prediction display with analytical visualizations (top-5 bar chart, macronutrient pie chart). A confidence threshold (0.30, as defined in `config.py`) triggers warnings for low-confidence predictions, ensuring reliability. The nutritional table and pie chart provide complementary views of dietary data, while recommendations address practical needs, such as portion control or nutrient balancing, as detailed in `ui.py`. The system's dependencies are managed via `requirements.txt`, ensuring reproducibility.


Food Recognition
http://localhost:8501
Deploy
Overview
Confidence threshold: 0.25
Accuracy: 83.6%
Model: InceptionV3 on Food-101
Nutrition data: USDA API
Food Image Classification & Nutritional Analysis
Upload a food image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
Please upload an image to continue.
Home / Projects / Food_... roject / data / demo
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Bánh Cuốn Tây Sơn.jpg
Bánh Táo.jpg
Bò Một Nắng.jpg
Bún Đậu Mắm Tôm.jpeg
Cơm Tấm.jpg
Hot Dog.jpeg
Kem Tươi.jpg
Lẩu Gà Lá É.jpg
Mi Ramen.jpeg
Mi Ý.jpg
Pad Thái.jpg
Phở.jpg
Phở Khô.jpg
Trà Sữa Chân Châu.jpg


Deploy
Upload a food image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
Hot Dog.jpeg 26.7KB
Bánh Mì
HotDog
Top-5 Predictions
club_sandwich
macarons
spring_rolls
sushi
hot_dog
label
0
10
20
30
40
50
60
Confidence (%)
hot_dog (62.80%)
Description

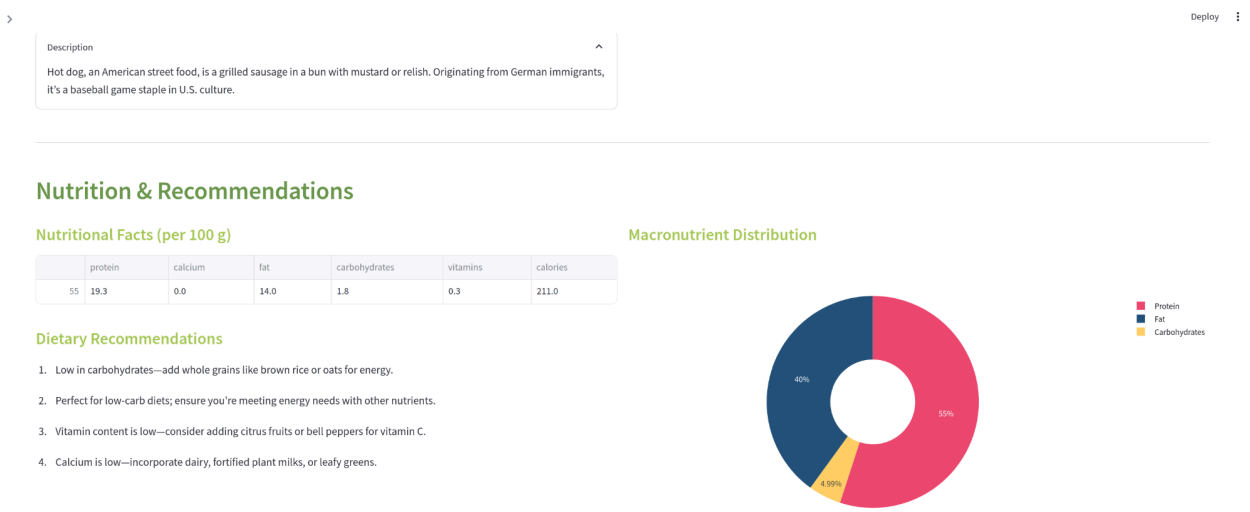


Figure 4.6: UI Workflow Visualization.

Conclusion

This integration delivers an efficient, user-friendly system, transforming the InceptionV3 model into a practical tool for dietary insights. The rapid inference time (0.0205 seconds/image) and intuitive UI enhance accessibility, with potential for future enhancements in dataset diversity and multilingual support..

Chapter 5: Conclusion and Future Work

5.1 Summary of Findings

This study developed a food image recognition system using the InceptionV3 model, fine-tuned on the Food-101 dataset (70,700 training, 15,150 validation, and 15,150 test images), achieving a validation accuracy of 83.55% and a test accuracy of 83.06%, with a macro-averaged F1-score of 0.8344 (validation) and 0.8300 (test). The model outperformed prior benchmarks (e.g., 78% accuracy by Bossard et al., 2014) due to effective transfer learning and data augmentation. Visualizations in `evaluation.ipynb`, including clustered confusion matrices and t-SNE plots, revealed robust classification for distinct classes like `edamame` but challenges with visually similar classes. Grad-CAM and activation visualizations highlighted critical features like meat texture, enhancing interpretability. The model's 0.0205-second inference time enabled seamless integration into a Flask and Streamlit-based web application, delivering real-time food recognition and nutritional insights.

5.2 Contributions of the Study

This study advances the field of food image recognition by developing and deploying an InceptionV3-based model on the Food-101 dataset. Key contributions include:

Robust Model Development: The fine-tuned InceptionV3 model achieved a validation accuracy of 83.55% and a test accuracy of 83.06%, with a macro-averaged F1-score of 0.8344 (validation) and 0.8300 (test), surpassing prior benchmarks (e.g., 78% accuracy by Bossard et al., 2014). This demonstrates effective application of transfer learning and data augmentation techniques, central to Data Science methodologies.

Comprehensive Evaluation: Utilizing tools like scikit-learn and Seaborn in `evaluation.ipynb`, the study provided detailed performance analysis through metrics (accuracy, AUC, F1-score) and visualizations (clustered confusion matrices, t-SNE, Grad-CAM), identifying challenges like the 13.33% misclassification rate between `steak` and `filet_mignon`. This rigorous evaluation enhances model interpretability, a core Data Science principle.

Practical Deployment: The integration of the model into a Flask and Streamlit-based web application with a 0.0205-second inference time enables real-time food recognition and nutritional analysis, showcasing Data Science's impact on practical, user-facing solutions.

Interpretability Insights: By employing Grad-CAM and activation visualizations, the study elucidates feature importance (e.g., meat texture for `steak`), advancing the interpretability of deep learning models in Data Science applications.

These contributions provide a scalable framework for food recognition, offering insights for dietary tracking and potential extensions to broader Data Science applications in image-based analytics.

5.3 Limitations

Despite the robust performance of the InceptionV3 model, this study faces several limitations.

First, the model struggles with visually similar classes, such as `steak` and `filet_mignon`, exhibiting a 13.33% misclassification rate, as revealed in `evaluation.ipynb` (**Figure 4.3.2**). This suggests limitations in feature discrimination, potentially due to the dataset's lack of contextual data (e.g., preparation methods).

Second, the model's performance (83.55% validation accuracy, 0.8344 F1-score) may not generalize to diverse, real-world food images beyond Food-101's controlled setting.

Third, the web application (`app.py`, `ui.py`, `utils.py`, `food_descriptions.py`, `get_nutrition_data.py`) relies on a single model, limiting adaptability to new food categories without retraining.

Finally, computational constraints, noted in `history_101class.log`, restricted hyperparameter tuning, potentially capping performance optimization.

5.4 Suggestions for Future Research

Building on the findings and limitations of this study, several research directions can enhance the food image recognition system developed using the InceptionV3 model on the Food-101 dataset. Addressing the 13.33% misclassification rate between visually similar classes, identified in `evaluation.ipynb`, future work could:

Incorporate Contextual Data: Augment the dataset with contextual information (e.g., preparation methods, ingredient lists) to improve discrimination between similar classes, enhancing the model's feature extraction beyond visual cues.

Explore Advanced Architectures: Investigate models like EfficientNet or Vision Transformers, which may offer superior feature learning, potentially increasing the validation accuracy beyond 83.55% and F1-score beyond 0.8344.

Expand Dataset Diversity: Collect a broader dataset including real-world, user-generated images to improve generalization, addressing the limitation of Food-101's controlled setting.

Enhance Web Application Adaptability: Modify the web application to support dynamic class updates without retraining, using techniques like few-shot learning.

Optimize Computational Efficiency: Leverage more extensive hyperparameter tuning and distributed computing to overcome constraints, potentially reducing the 0.0205-second inference time further.

These directions aim to improve model robustness, interpretability, and practical utility for real-time dietary analysis.

Appendix

Project Github's link: Food_Image_Recognition_Project.