

# Food Calories calculation using depth calculation

Submitted By  
**Patel Khushi**  
**21BCE203**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
INSTITUTE OF TECHNOLOGY  
NIRMA UNIVERSITY  
AHMEDABAD-382481

May 2025

# Food Calories calculation using depth estimation

## Project Report

Submitted in partial fulfillment of the requirements

for the degree of

**Bachelor of Technology in Computer Science and Engineering**

By

**Patel Khushi**

**(21BCE203)**

Guided By

**Prof. Sonia Mittal**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INSTITUTE OF TECHNOLOGY**

**NIRMA UNIVERSITY**

**AHMEDABAD-382481**

**May 2025**



## Certificate

This is to certify that the major project entitled “**Food Calories calculation using Depth calculatiione**” submitted by **Patel Khushi (21BCE203)**, towards the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Nirma University, Ahmedabad, is the record of work carried out by him/her under my supervision and guidance. In my opinion, the submitted work has reached the level required for being accepted for examination.

Prof. Sonia Mittal  
Assistant Professor  
Department of Computer Science & Eng.  
Institute of Technology,  
Nirma University, Ahmedabad.

Dr Sudeep Tanwar  
Professor and Head,  
Department of Computer Science & Eng.  
Institute of Technology,  
Nirma University, Ahmedabad

## CERTIFICATE OF PROJECT COMPLETION

This is to certify that **Patel Khushi**, bearing Roll Number **21BCE203**, has successfully completed her industry internship project titled: **Food calorie calculation using depth calculation**.

The project was carried out at **Vovance private limited** under the guidance of **Om Rajani**, from **8thth January 2025 to 30th April 2025**, as part of her academic requirements.

We appreciate her sincere efforts and contribution during the project period and wish her all the best for her future endeavors.

**Date:** 29/4/25



**Amit Patel**  
Project manager  
Vovance Pvt Ltd

## Statement of Originality

---

I, **Patel Khushi**, Roll. No. **21BCE203**, give an undertaking that the major project entitled **“Food Calories calculation using Depth calculation”** submitted by me, towards the partial fulfilment of the requirements for the degree of Bachelor of Technology in **Computer Science and Engineering**, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.



Signature of Student

Date:

Place: Ahmedabad



Endorsed by

Prof. Sonia Mittal

(Signature of Guide)

# Acknowledgements

I would like to extend my deepest appreciation for the valuable assistance and advice I have been given in developing my project, Food Calories Calculation Using Depth Estimation.

Special thanks to **Mr. Om Rajani**, whose guidance was instrumental throughout the project. His timely support, thoughtful feedback, and patient instruction guided me through different technical difficulties and shed light when I was lost or in doubt. His support always inspired me to step out of my comfort zone and aim for higher results. I would also like to thank my internal guide, Assistant Professor **Sonia Mittal**, for her academic guidance and insightful feedback. Her involvement played a crucial role in shaping the project's direction.

This project has been an eye-opening learning experience, enabling me to move from theory to practice. It has enhanced my technical skills in computer vision and artificial intelligence, and increased my appreciation for the possibilities of how technology can be used to address real-world issues like nutritional analysis and healthy living.

I am equally thankful to fellow colleagues and friends who made their contributions through helpful discussions, moral support, and sporadic brainstorming that generated new ideas. Their insights tended to make me see issues differently and develop creative solutions.

- **Patel Khushi**  
**21BCE203**

# Abstract

This paper introduces a holistic solution for food detection, classification, segmentation, and volume estimation based on deep learning methods. The system is constructed on a multi-stage pipeline, which involves food image classification, food portion segmentation, and food volume estimation from depth information.

During the food categorization stage, we use the Food-101 and Recipe-1M datasets and fine-tune the pre-trained Inception-ResNet-V2 model on ImageNet. The training of the model is done using several GPUs simultaneously, enabled through a tower loss paradigm, enabling scalable and efficient learning. Preprocessing of the dataset is performed into TFRecord format, and model evaluation is conducted on accuracy for classification.

The food segmentation step utilizes the RefineNet architecture, which is pretrained on the PASCAL VOC 2012 dataset, to segment single portions of food from input images. The model is fine-tuned on manually labeled food segmentation data to enhance its accuracy in the case of food images. The segmentation step produces pixel-level masks for various portions of food.

During the volume estimation stage, we employ the latest depth estimation methods to create depth maps of the food portion. A model pre-trained on the NYU-Depth V2 dataset is employed to make predictions of depth from RGB food images. By leveraging prior plate diameter knowledge and fusing depth maps with segmentation masks, the system estimates volume for each food portion.

The entire pipeline combines these elements to yield precise food detection, segmentation, and volume estimation, which may be beneficial for food tracking, nutrition analysis, and robotics applications in the food service sector. The approach has promising outcomes for these tasks, presenting a scalable and efficient solution to real-world food detection problems.

# Abbreviations

<b>GPU</b>	Graphics Processing Unit
<b>RGB</b>	Red-Green-Blue
<b>RGB-D</b>	Red-Green-Blue with Depth
<b>TFRecord</b>	TensorFlow Record
<b>API</b>	Application Programming Interface
<b>TF</b>	TensorFlow
<b>PIL</b>	Python Imaging Library
<b>JSON</b>	JavaScript Object Notation
<b>VOC</b>	Visual Object Classes
<b>Food-101</b>	Name of dataset (no acronym expansion)
<b>Recipe-1M</b>	Recipe One Million
<b>NYU Depth V2</b>	New York University Depth Dataset V2
<b>PASCAL VOC</b>	Pattern Analysis, Statistical Modelling and Computational
<b>SGD</b>	Stochastic Gradient Descent
<b>IoU</b>	Intersection over Union
<b>MAE</b>	Mean Absolute Error
<b>MSE</b>	Mean Squared Error
<b>ReLU</b>	Rectified Linear Unit
<b>BN</b>	Batch Normalization
<b>RMSE</b>	Root Mean Squared Error
<b>mAP</b>	Mean Average Precision

---



# Contents

<b>Certificate</b>	<b>iii</b>
<b>Statement of Originality</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>Abbreviations</b>	<b>vii</b>
<b>List of Tables</b>	<b>x</b>
<b>List of Figures</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 About company - Vovance Pvt Ltd . . . . .	1
1.2 Different products and services . . . . .	1
1.3 Company Values And Philosophy . . . . .	2
<b>2 Literature Survey</b>	<b>3</b>
2.1 Models used for Image classification :- . . . . .	3
2.2 Methods used for Volume estimation :- . . . . .	5
<b>3 Project Introduction</b>	<b>7</b>
3.1 Overview . . . . .	7
3.2 Purpose . . . . .	7
3.3 Objectives . . . . .	8
3.4 Datasets used . . . . .	8
3.4.1 Food-101 . . . . .	8
3.4.2 Nutrition5k . . . . .	10
3.4.3 Recipie-1M . . . . .	10
3.4.4 NYU-Depth V2 . . . . .	11
3.5 Technologies used . . . . .	12
<b>4 Methodology Followed</b>	<b>15</b>
4.1 Food image classification . . . . .	15
4.1.1 Various models used for classification . . . . .	15
4.1.2 Final Model used in project(Inception-ResNet-V3 model) . . . . .	16
4.2 Food segmentation . . . . .	18
4.2.1 Configuration and Initialization . . . . .	18

4.2.2	Data Preparation and Augmentation . . . . .	18
4.2.3	Model Architecture . . . . .	19
4.2.4	Training Strategy . . . . .	19
4.2.5	Inference and Evaluation . . . . .	20
4.3	Volume Estimation . . . . .	21
4.3.1	Image and Annotation Preparation . . . . .	21
4.3.2	Depth Estimation Using a Pre-trained Deep Network . . . . .	21
4.3.3	Semantic Segmentation via Polygon Masking . . . . .	22
4.3.4	Volume Estimation Through Depth Integration . . . . .	22
4.3.5	Output and Logging . . . . .	23
<b>5</b>	<b>Algorithms</b>	<b>24</b>
5.1	Food classification: . . . . .	24
5.2	Image segmentation: . . . . .	24
5.3	Depth calculation: . . . . .	24
<b>6</b>	<b>Model Results</b>	<b>28</b>
6.1	EfficientNetB0 . . . . .	28
6.2	ResNet50 . . . . .	28
6.3	Inception-V3 . . . . .	29
6.3.1	Inception Model Intermediate Layers . . . . .	30
6.4	Image Segmentation . . . . .	33
6.5	Volume/Depth Calculation . . . . .	33
6.6	Calorie Calculation Results . . . . .	34
<b>7</b>	<b>Conclusion</b>	<b>37</b>
<b>8</b>	<b>Future Work</b>	<b>39</b>
	<b>Bibliography</b>	<b>41</b>

# List of Tables

2.1	Comparison of Deep Learning Models for Image Classification . . . . .	3
2.2	Comparison of volume estimation methods using image-based and deep learning approaches . . . . .	6

# List of Figures

3.1	Food-101 dataset images . . . . .	9
3.2	Nutrition5k dataset . . . . .	10
3.3	NRecipe1M dataset . . . . .	11
3.4	NRecipe1M dataset . . . . .	12
4.1	ResNet50 Architecture . . . . .	16
4.2	MobileNetV2 Architecture . . . . .	17
4.3	Inception-ResNet-V2 Architecture . . . . .	18
4.4	RefineNet Architecture . . . . .	21
5.1	Food classification Algorithm . . . . .	25
5.2	Image segmentation Algorithm . . . . .	26
5.3	Depth calculation Algorithm . . . . .	27
6.1	EfficientNetB0 accuracy . . . . .	28
6.2	ResNet50 model accuracy . . . . .	28
6.3	Inception model accuracy . . . . .	29
6.4	Inception model loss . . . . .	30
6.5	Heat map . . . . .	31
6.6	Conv2D . . . . .	31
6.7	Batch Normalization . . . . .	31
6.8	Activation . . . . .	31
6.9	Conv2D-1 . . . . .	31
6.10	Batch Normalization-1 . . . . .	32
6.11	Activation-1 . . . . .	32
6.12	Conv2D-2 . . . . .	32
6.13	Batch Normalization-2 . . . . .	32
6.14	Activation-2 . . . . .	33
6.15	Max Pooling . . . . .	33
6.16	Image segmentation output . . . . .	33
6.17	Depth calculation result . . . . .	34
6.18	Calorie estimation result - Example 1 . . . . .	34
6.19	Calorie estimation result - Example 2 . . . . .	35
6.20	Calorie estimation result - Example 3 . . . . .	35
6.21	Calorie estimation result - Example 4 . . . . .	36

# Chapter 1

## Introduction

### 1.1 About company - Vovance Pvt Ltd

Vovance Private Limited is a growing IT services and software development company headquartered in Ahmedabad, Gujarat, India. Established in 2017, the company focuses on delivering cutting-edge digital solutions across a variety of industries, with a core specialization in mobile app development, web development, cloud solutions, and eCommerce platforms. Led by professionals with a vision for empowering businesses through technology, Vovance has positioned itself as a reliable technology partner that blends deep technical expertise with strong industry knowledge.

### 1.2 Different products and services

- **Web Design & Development:** Vovance offers full-stack web development services using popular technologies such as Laravel, PHP, Node.js, React, and Angular. The company focuses on developing responsive, secure, and SEO-optimized websites and web applications that enhance user experience and business performance.
- **Mobile App Development:** From concept to deployment, we bring your mobile app ideas to life on Android platforms. Whether developing native or hybrid apps, we prioritize seamless user experiences and robust performance, making your app stand out in a competitive market. Each app is tailored to meet the unique needs of your target audience, providing exceptional value and engagement.

- **E-Commerce Solutions:** The company builds customized eCommerce websites and platforms using tools like Magento, Shopify, and custom-built solutions. These platforms come equipped with features like shopping cart integration, payment gateways, inventory management, and customer engagement tools, helping businesses succeed in the digital marketplace.
- **Custom Software Development:** Vovance develops tailor-made software solutions based on specific business requirements. These can range from ERP and CRM systems to booking engines, data management tools, and reporting dashboards. The software is designed to improve workflow, enhance productivity, and support digital transformation.

### 1.3 Company Values And Philosophy

- **Excellence:** In every service, from client projects to student training, we are committed to delivering high-quality solutions that exceed expectations. Our focus on excellence ensures lasting value and impactful outcomes across all our offerings.
- **Integrity** We conduct our work with honesty, transparency, and ethical standards, building trust with clients, partners, and trainees. Integrity is fundamental to our approach, ensuring reliability in every interaction.
- **Innovation** Embracing innovation, we seek fresh and effective solutions for our clients' needs while continually updating our training curriculum. By integrating the latest technologies and approaches, we help clients and students stay competitive in their respective fields.
- **Collaboration** Recognizing the power of teamwork, we foster a collaborative environment where skills and perspectives unite for better outcomes. We prioritize open communication with clients, partners, and students to cultivate an inclusive and effective learning and project development experience.
- **Client & Student-Centricity** For clients, we offer tailored solutions that address unique challenges. For students, we provide training that is relevant, hands-on, and aligned with their career goals, focusing on equipping them with practical skills and industry insights.

# Chapter 2

## Literature Survey

### 2.1 Models used for Image classification :-

In recent studies, various deep learning models have been proposed and evaluated using different datasets to address image classification issues efficiently. Each research considered various datasets, architectures, and optimization techniques to improve performance while overcoming computational issues.

Paper Title	Dataset Used	Models Implemented	Accuracy Achieved	Future Improvements
DenseMobile Net: Deep Ensemble Model for Precision and Innovation in Indian Food Recognition [1]	26 Indian Food Dataset (Kaggle)	DenseNet-121, MobileNetV3, Ensemble (Max Voting)	DenseNet: 90.00%, MobileNetV3: 87.64%, Ensemble: 92.38%	Data augmentation, real-time optimization, multi-modal classification, healthcare application deployment
Development of Korean Food Image Classification Model Using Public Food Image Dataset and Deep Learning Methods [2]	AI-Hub Korean Food Image Dataset (150 classes)	ResNet-50V2, ResNet-101V2, ResNet-152V2, InceptionResNetV2, MobileNetV2, NasNetLarge	Best: <b>74% (InceptionResNetV2)</b>	Hierarchical classification, improved augmentation, explore more backbones (EfficientNet, CoCa)
Image Classification using Deep Learning: A Comparative Study of VGG-16, InceptionV3 and EfficientNet B7 Models [3]	Not explicitly stated (likely CIFAR-10 or ImageNet)	VGG-16, InceptionV3, EfficientNet B7	VGG-16: 85.2%, InceptionV3: 91.8%, EfficientNetB7: 94.6%	Dataset expansion, hybrid models, scalable optimization, real-world deployment
MobileNetV2 Model for Image Classification [4]	Likely CIFAR-10 or small ImageNet subset	MobileNetV1, MobileNetV2	MobileNetV1: 71.8%, MobileNetV2: 75.6%	Further latency optimization, edge-device efficiency, integration into hybrid models
Deep Learning Models for Image Classification: Comparison and Applications [5]	MNIST, ImageNet, medical and agricultural datasets	AlexNet, ResNet, GoogLeNet, VGG-16	ResNet-50: 99.3% (MNIST), GoogLeNet: 98.82%, VGG-16: 97.87%	Lightweight CNNs, custom loss functions, domain-specific tuning, hybrid CNN frameworks

Table 2.1: Comparison of Deep Learning Models for Image Classification

As shown in Table 2.1, the paper DenseMobile Net[1] addressed Indian food identification using ensemble learning between DenseNet-121 and MobileNetV3. The model recorded 92.38% testing accuracy and proved robustness and accuracy, especially in application to healthcare and diet monitoring.

The paper Development of Korean Food Image Classification Model[2] made use of an extensive dataset available at AI-Hub to label 150 classes of Korean dishes. Among many pre-trained Inception CNNs, InceptionResNetV2 demonstrated outstanding performance with accuracy of 74%. The effort pointed out similarity in food class and proposed hierarchical models as the future direction.

In the paper Image Classification using Deep Learning, there was a comparative study of VGG-16, InceptionV3, and EfficientNetB7. EfficientNetB7[3] produced the highest accuracy of 94.6%, surpassing legacy models by optimizing depth, width, and resolution for scalable and efficient computation.

The paper MobileNetV2 Model for Image Classification[4], meanwhile, focused on lightweight and mobile-friendly architecture enhancements. With 75.6% accuracy, MobileNetV2 greatly outperformed the legacy model MobileNetV1, being best suited for real-time embedded AI use.

Lastly, the extensive survey Deep Learning Models for Image Classification[5] compared a number of top-performing models such as AlexNet, ResNet, GoogLeNet, and VGG. ResNet-50 was the best performer with 99.3% accuracy on MNIST, demonstrating that deeper networks with skip connections perform better in dealing with challenging datasets.

Overall, across all the studies, the trend is firmly set towards optimizing deep learning architectures to improve accuracy, efficiency, and responsiveness to real-world applications. Future research continually indicates hybridization, more intelligent data augmentation, edge-device optimization, and movement into more sophisticated and multi-modal data types.



## 2.2 Methods used for Volume estimation :-

The literature reviewed provides a wide variety of methods for estimating the volume of objects and food products through image-based methods, with a high focus on utilizing deep learning and depth information. A particular study suggests a new method that utilizes RGB-D images obtained by an Intel RealSense D415 camera to estimate the volume of irregular objects. The technique has an average volume estimation error of just 2.37%, which signifies better accuracy compared to conventional geometric techniques.

In dietary assessment, a broad review is presented that indicates the advancement of image-based food recognition and volume estimation. Some of the publicly available datasets like Food101 and UEC-FOOD100 have been extensively employed for training and testing these models. The review highlights the need for system integration that has the capability to surpass the individual drawbacks of each technique, and the possibility of delivering more accurate dietary intake predictions.

A second survey targets systems of artificial intelligence for estimating food volume. It highlights the shift from traditional machine learning methods to contemporary deep learning models, facilitated by extensive food image datasets such as NutriNet and UNICT-FD889. Techniques discussed include 3D reconstruction, shape template matching, and depth-sensing devices. Further improvement of dataset development, segmentation accuracy, and deep learning models to enhance system robustness and performance in the real world is recommended by the paper.

A last study presents a smartphone-based approach that estimates the volume of fruits and vegetables from brief monocular video clips along with inertial measurement data. With a combination of Mask R-CNN for object detection and LSTM networks for motion and volume estimation, the approach has a classification accuracy of 95% and a mean absolute percentage error of 16% in volume estimation for novel objects. This method shows the viability of employing easily accessible consumer hardware to estimate volumes accurately and affordably without the necessity of reference objects or other devices.

Title	Dataset Used	Methods used	Results	Future Aspects
Volume Estimation Method for Irregular Object Using RGB-D Deep Learning [6]	101 RGB-D image pairs from Intel RealSense D415; includes regular and clay-based irregular objects with known volumes	Object detection via SAM on RGB images, Depth correction using $5 \times 5$ mode filter, Height from background comparison, Deep learning on $2 \times 2$ pixel units	Avg. volume estimation error: 2.37%	Improve model robustness, Real-time/dynamic object support, Handle occlusions/complex shapes
Image-Based Food Classification and Volume Estimation for Dietary Assessment: A Review [7]	Multiple datasets including Food101, UEC-FOOD100, UEC-FOOD256	Stereo-based, Model-based, Depth camera, Perspective, transformation, Deep learning	DL methods outperform traditional ones in accuracy	Develop integrated dietary systems, Overcome limitations of individual methods
A Review of Image-Based Food Recognition and Volume Estimation AI Systems [8]	Wide range: Food101, UEC-Food100/256, VIREO, NutriNet, UNICT-FD889, etc.	3D reconstruction, Predefined shape templates, Perspective transformation, Depth camera Deep learning	DL improves accuracy/efficiency over traditional ML	Enhance food image databases, Improve segmentation/classification, Use advanced DL models
Learning Metric Volume Estimation of Fruits and Vegetables from Short Monocular Video Sequences [9]	Custom dataset from Huawei Mate 20 Pro (videos + IMU for 11 fruit/veg classes)	Mask R-CNN for segmentation, LSTM-based IMU encoder for pose, LSTM-based volume estimator combining image + IMU	Classification accuracy: 95%, Volume error (MAPE): 16%	Leverage smartphones for practical apps, Dataset for future research and modeling

Table 2.2: Comparison of volume estimation methods using image-based and deep learning approaches

# Chapter 3

## Project Introduction

### 3.1 Overview

The project suggests a single food analysis system with deep learning techniques for three critical phases: food categorization through a fine-tuned Inception-ResNet-V2 model on Recipe-1M, Food-101 and Nutrition5k datasets; food segmentation through RefineNet architecture to produce pixel-level masks of individual food portions; and volume estimation through combining depth maps (obtained from RGB images with models pre-trained on NYU-Depth V2) with segmentation masks and plate diameter. This integrated pipeline enables accurate food detection, classification, segmentation, and volume estimation, which have numerous potential applications in nutritional monitoring, food service automation, and healthcare monitoring systems.

### 3.2 Purpose

The goal of this project is to build an advanced computer vision system that is capable of automatically analyzing food images to detect food types, segment out individual portions, and estimate their volume. The technology seeks to replace the manual labor involved in food logging and nutritional analysis through precise, automated breakdown of meals using photographs. Through the integration of deep learning methods in classification, segmentation, and volume estimation, the system aims to offer an end-to-end solution that can revolutionize how individuals monitor and comprehend their food consumption.

### 3.3 Objectives

- Implement a strong food image classification system through Inception-ResNet-V2 architecture for precise identification of food items across varied cuisines and presentations.
- Implement an exact food segmentation model through RefineNet that is able to differentiate individual portions of food within composite meal images.
- Develop a depth estimation method that translates 2D food images into precise 3D models for volume calculation.
- Implement an end-to-end pipeline that perfectly integrates classification, segmentation, and depth estimation for end-to-end food analysis.

### 3.4 Datasets used

#### 3.4.1 Food-101

- Comprises 101,000 food images in 101 food categories 1,000 images per category (750 for training, 250 for testing)
- Built from real images gathered from Foodspotting, a food photo sharing website
- Training images contain some erroneous labels on purpose to mimic real-world data complexity
- Images are shot under varying lighting conditions, angles, and plating configurations
- Contains varied food categories: American foods, Italian cuisine, Asian foods, desserts, breakfast foods, etc.
- Challenging due to visual similarity between some food categories Available publicly through various machine learning libraries and repositories
- Regarded as a difficult benchmark because of its real-world variability

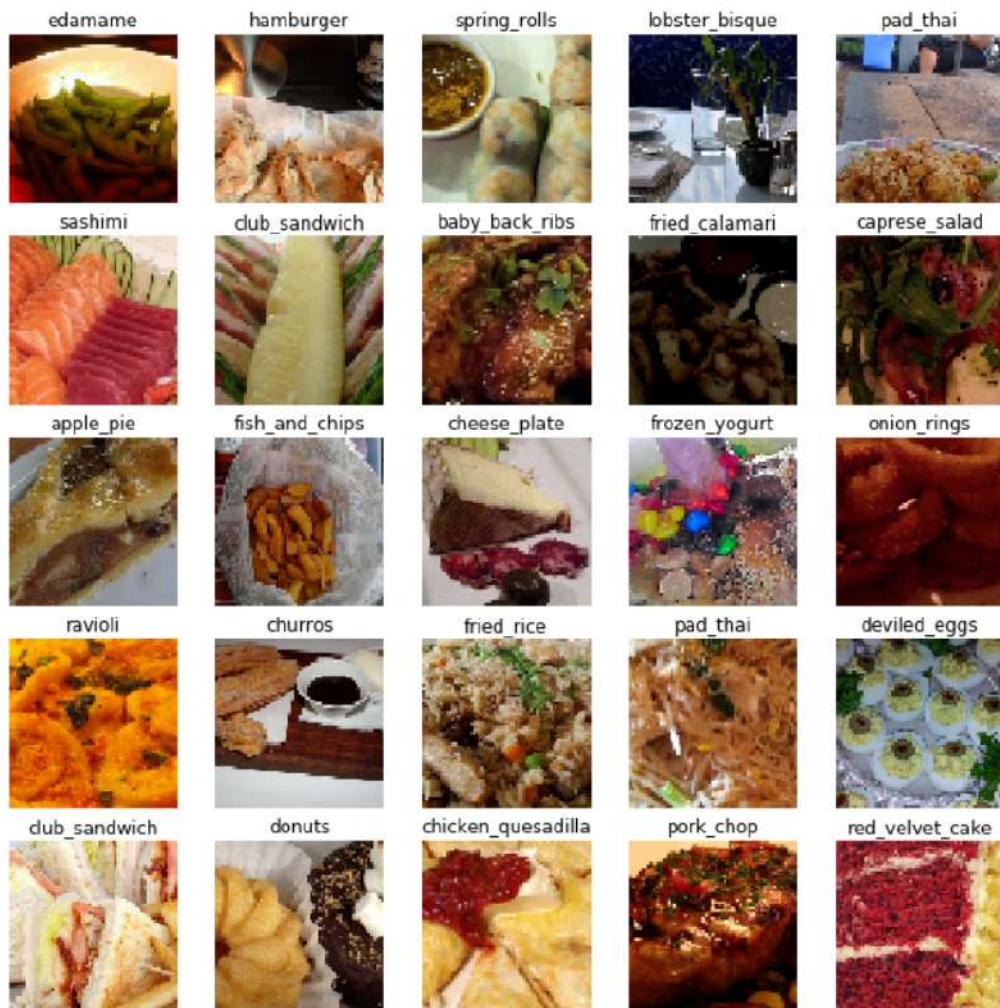


Figure 3.1: Food-101 dataset images

### 3.4.2 Nutrition5k

- Includes around 5,000 images of food trays with accurate nutritional measurements. Each example provides RGB images, depth maps, and segmentation masks
- Four measurements of nutrition for each food: mass, calories, carbs, and protein
- Food trays were photographed using a standardized imaging station with controlled illumination. Each tray holds 1-4 foods from a total of around 300 distinct foods
- Images per food plate from multiple views for a total of 40,000 images with ground truth nutritional labels
- Much more extensive nutritional annotations compared to other food image datasets
- Supports classification and regression ML tasks involving food Issued with baseline models and benchmarked results for nutrition estimation
- Specifically created to handle real-world food nutrition tracking scenarios Provides fine-grained metadata for every food item

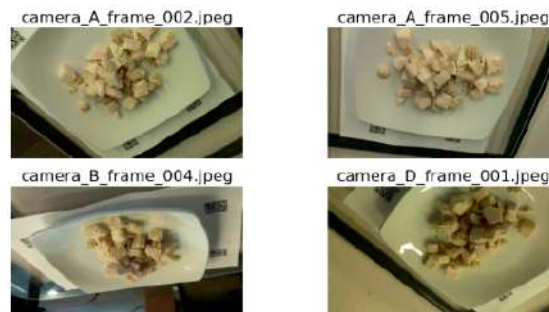


Figure 3.2: Nutrition5k dataset

### 3.4.3 Recipe-1M

- Comprises more than 1 million recipes for cooking and 13 million related food images
- Illuminated each recipe contains structured text data: title, ingredients list, and cooking instructions
- Has around 333 semantic labels for food categories

- Supports recipe metadata like cuisine and course information. Facilitates cross-modal retrieval problems
- Includes natural language understanding challenges from variations in recipe text
- Contains nutritional data for a portion of the recipes. Also used as baseline for food identification and recipe lookup systems Issued with baseline models and benchmarked results for nutrition estimation
- Enables procedural learning of cooking processes and methods




Meal Image	Ingredient List	Instruction List
	<ol style="list-style-type: none"> <li>1. chicken</li> <li>2. garlic</li> <li>3. butter</li> <li>4. lemon</li> <li>5. rosemary</li> <li>6. basil</li> <li>7. thyme</li> <li>8. salt and pepper.</li> </ol>	<ol style="list-style-type: none"> <li>1. Preheat oven to 370F.</li> <li>2. Rub salt and pepper to the chicken and set aside for around 10 mins.</li> <li>3. rub half of the butter to the chicken.</li> <li>4. Mince 1 whole garlic and rub on the chicken.</li> <li>5. Rub chicken with Herbs.</li> <li>6. Stuff chicken with lemon, butter, 1 whole garlic, salt, pepper and herbs.</li> <li>7. Put chicken in oven covered with foil for 45 mins.</li> <li>8. Remove cover and cook for another 45 mins at 400F</li> </ol>
	<ol style="list-style-type: none"> <li>1. ladyfingers</li> <li>2. cream cheese</li> <li>3. lemon juice</li> <li>4. lemon jelly powder</li> <li>5. boiling water</li> <li>6. ice cubes</li> <li>7. cool whip topping</li> <li>8. fresh raspberries</li> <li>9. jelly powder</li> </ol>	<ol style="list-style-type: none"> <li>1. Grease 9 inch (23 cm) springform pan.</li> <li>2. Place lady fingers around inside rim and set aside</li> <li>3. Beat cream cheese in large bowl of electric mixer.</li> <li>4. Add lemon juice and rind, beating on low speed until blended.</li> <li>5. Dissolve lemon jelly powder in boiling water.</li> <li>6. Add ice cubes, stirring until slightly thickened.</li> <li>7. Add lemon jelly slowly to cream cheese mixture while beating.</li> <li>8. Increase speed and beat just until well blended.</li> <li>9. etc. ....</li> </ol>
	<ol style="list-style-type: none"> <li>1. strawberries</li> <li>2. pineapple</li> <li>3. non-dairy coffee creamer</li> <li>4. orange juice</li> </ol>	<ol style="list-style-type: none"> <li>1. In blender add strawberries, pineapple, non-dairy creamer and orange juice and blend until smooth.</li> <li>2. Poor into frosted glass and enjoy</li> </ol>

Figure 3.3: NRecipe1M dataset

### 3.4.4 NYU-Depth V2

- Includes 1,449 densely labeled RGB-D image pairs taken from indoor scenes
- Acquired with Microsoft Kinect sensors across varied indoor settings. Comprises 464 unique indoor scenes spread over 3 cities
- Offers a color and depth synchronous data at  $640 \times 480$  resolution
- Further categorized into 40 popular semantic classes for benchmarking. Includes depth maps with measurements in meters
- Very popular benchmark for depth estimation from RGB images. Enables research on scene understanding, 3D reconstruction, and semantic segmentation



- Object instance segmentation for a subset of images
- Supports both supervised and unsupervised learning methods. Allows research on 3D scene completion from partial observations

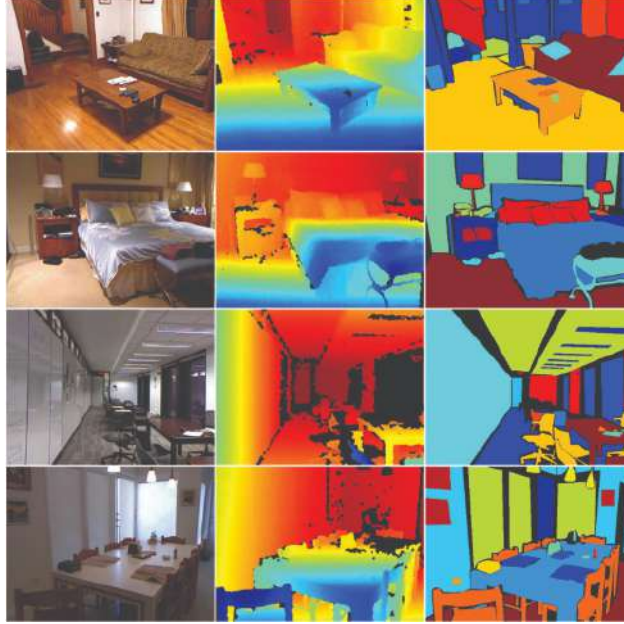


Figure 3.4: NRecipie1M dataset

### 3.5 Technologies used

1. **TensorFlow 1.8.0** A deep learning framework employed to train and test both the food classification and segmentation models (RefineNet and Inception-ResNet-V2). Multi-GPU training with the tower loss scheme is supported in version 1.8.0.

2. **PyTorch 0.4.1**

Employed for depth estimation during the volume calculation stage. Version 0.4.1 offers flexible and dynamic computational graphs to simplify prototyping of the depth model.

3. **Inception-ResNet-V2**

A high-capacity convolutional neural network using Inception modules and residual links. The model is pretrained and tuned for food image classification in the Food-101 dataset.



#### 4. **RefineNet**

A semantic segmentation model for high-resolution inputs that is applied to segment various food products in an image. Pretrained on PASCAL VOC 2012 first, then tuned using food segmentation data.

#### 5. **Hu et al.’s Depth Estimation Model**

A single-image depth estimation model pretrained on NYU-Depth V2. It produces depth maps from RGB images to aid in food volume estimation.

#### 6. **Food-101 Dataset**

A public dataset with 101 food categories and 1,000 images per category, utilized for training and testing the food classification model.

#### 7. **Recipe-1M Dataset**

Another large-scale food dataset (optional in this project) that is paired with recipes, utilized similarly to Food-101 for classification purposes.

#### 8. **NYU-Depth V2**

A dataset for pretraining the depth estimation model. It consists of indoor scenes and their corresponding RGB and depth images.

#### 9. **OpenCV**

A library for image processing used to handle and transform images, particularly in segmentation and visualization operations.

#### 10. **Pillow**

A Python imaging library employed to read, write, and transform image files in various pipeline stages.

#### 11. **Numpy**

A fundamental library for numerical computing in Python. It supports large, multi-dimensional arrays and is used throughout the data pipeline.

#### 12. **Scipy**

Used for scientific and mathematical computations, such as image processing and interpolation.

13. **Matplotlib**

Used for plotting and visualizing depth maps, segmentation results, and other intermediate outputs.

14. **Pickle**

A Python module used to serialize and deserialize Python objects, especially useful for saving intermediate model data or annotations.

15. **JSON**

Used to save segmentation annotations and configuration files in a compact, human-readable format.

# Chapter 4

## Methodology Followed

### 4.1 Food image classification

#### 4.1.1 Various models used for classification

##### **Resnet50:**

ResNet50 training for food detection starts with fine-tuning the pre-trained architecture for food classification, with the last layers changed to detect food categories rather than ImageNet classes. The model is subsequently trained iteratively with GPU acceleration, with food image batches passed through the network as an optimizer updates parameters according to computed loss between predictions and true food labels.

The training adheres to a structured pattern of switching between training and validation periods over several epochs, with rigorous performance monitoring of both accuracy and loss metrics. Overfitting is avoided by early stopping methods that track validation performance, which saves the optimal model weights automatically when improvements are detected and stops training when performance stabilizes. This balanced strategy allows the ResNet50 model to build effective food detection capabilities while ensuring effective generalization to new images. The architecture of ResNet50 is shown in the Figure 4.1.

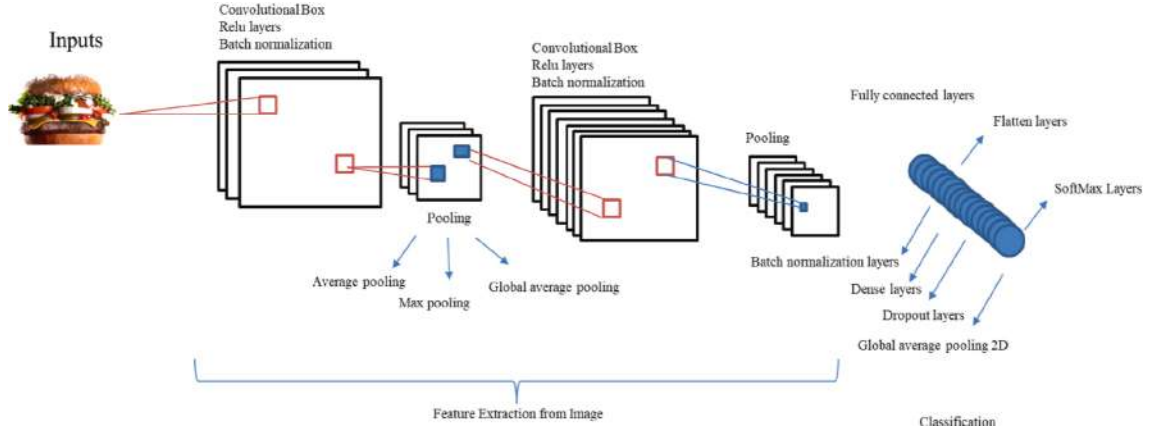


Figure 4.1: ResNet50 Architecture

### MobileNetV2-EfficientNetB0:

Training on the food detection involves transfer learning based on pre-trained EfficientNetB0 and MobileNetV2 architectures where the convolutional base layers remain the same, and new classification layers (GlobalAveragePooling2D followed by a Dense layer with softmax activation) are appended in order to classify 20 categories of foods. Both deployments utilize the Adam optimizer with small learning rates (0.0001-0.0005) and categorical cross-entropy loss, but have a different training strategy—EfficientNetB0 trains for a maximum of 20 epochs with patience of 5, whereas MobileNetV2 employs a briefer 10-epoch training period augmented by a learning rate reduction strategy when validation performance stagnates.

The training system for both models incorporates early stopping mechanisms that track validation loss to avoid overfitting, restore weights automatically to the best, and pass batches of  $224 \times 224$  pixel images through training and validation generators, rendering an effective and efficient method of food detection that balances model performance and computational cost. Figure 4.2 demonstrates the model architecture used.

#### 4.1.2 Final Model used in project(Inception-ResNet-V3 model)

Food detection methodology entails an organized method applying deep learning tailored to classify and detect food categories from images. The process is initiated with a meticulously selected dataset, where preprocessed food images are made uniformly consistent. All images are resized to a predetermined size and normalizing techniques applied to normalize the pixel values. Preprocessing images is crucial to achieve the optimum per-

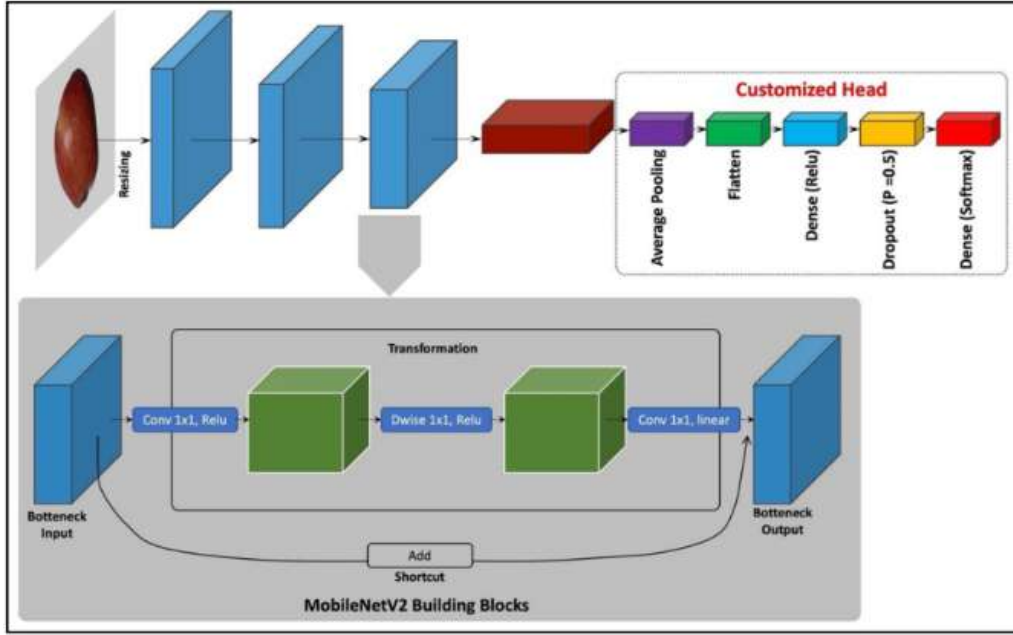


Figure 4.2: MobileNetV2 Architecture

formance of the model and ensuring uniformity between training and test times.

After the dataset is ready, it is split into training and validation sets. During the training process, batches of preprocessed images are fed into a deep convolutional neural network model. The model is trained to learn complex hierarchical features from the input images, progressively differentiating between food classes. To improve training efficiency and precision, the learning process is spread across several computing units (e.g., GPUs), enabling the model to process bigger batches and update weights more efficiently. The model learns continuously by minimizing a loss function that quantifies the discrepancy between its predictions and the true food labels.

Throughout training, several optimization techniques are used to enhance convergence and avoid overfitting. They include the utilization of momentum-based optimization, regularization methods, and adaptive learning rates. Training is done for a number of epochs, with performance evaluated intermittently on an independent validation set to check its ability to generalize.

Once training has finished, the model is utilized for testing. In this stage, it learns on unseen images of the validation dataset and makes predictions of the most probable food

class per image. The predictions are stored in an orderly fashion by filing each image into folders labeled with the food class recognized. This facilitates the visual inspection of the model’s performance as well as ensuring the accuracy of the prediction. The organized output serves as a tangible representation of the model’s classification capabilities, enabling both qualitative and quantitative analysis.

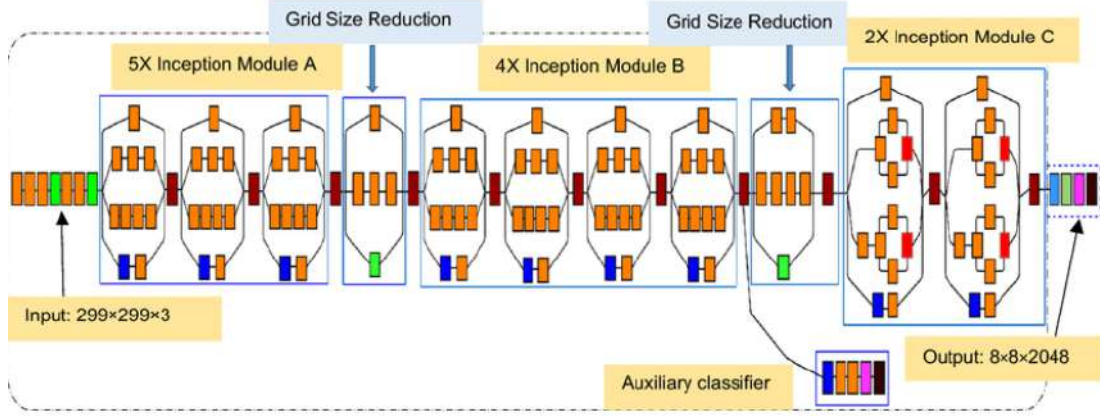


Figure 4.3: Inception-ResNet-V2 Architecture

## 4.2 Food segmentation

### 4.2.1 Configuration and Initialization

The workflow starts by determining critical configuration parameters. These are:

- **Training parameters:** Batch size, learning rate, and image dimensions.
- **Paths:** Directories where logs, model checkpoints, and datasets are stored.
- **Model configuration:** For example, output class count per food category.

These parameters make both the training and inference processes fully customizable and reproducible.

### 4.2.2 Data Preparation and Augmentation

Preprocessed food image data along with pixel-level annotations (masks) are provided as input to the training pipeline. Input images and masks are serialized and stored in a format optimized for TensorFlow.

**Augmentation methods** are used at training time to enhance model robustness and generalizability:

- **Random horizontal flipping:** To mimic varied orientations.
- **Color distortions:** To render the model invariant to lighting conditions.
- **Random scaling:** To accommodate size variability of foodstuffs.

The augmented data are batched and pre-fetched to optimize GPU usage during training.

### 4.2.3 Model Architecture

A multi-stage deep model specifically designed for semantic segmentation is employed. This design is renowned for maintaining spatial detail and integrating deep semantic information, making it suitable for segmenting fine-grained details in food images.

Key features include:

- **Multi-resolution feature refinement:** Improves segmentation accuracy by fusing low-level spatial features with high-level semantic features.
- **Support for multi-GPU training:** Facilitates faster convergence and scalability.

### 4.2.4 Training Strategy

The training process is carefully designed for optimal performance:

- **Dynamic learning rate:** Initialized at a fixed value and decayed over time to stabilize convergence.
- **Adam optimizer:** Utilized for adaptive learning rate updates.
- **Synchronized weight updates:** Achieved through gradient averaging across multiple GPUs.
- **Loss calculation:** Incorporates both regularization loss and classification error.

- **Exponential Moving Average (EMA):** Applied to model parameters to enhance stability and inference accuracy.

Progress is monitored by:

- Saving model checkpoints periodically.
- Writing detailed summaries to logs.
- Visualizing sample predictions during training.

### 4.2.5 Inference and Evaluation

The trained model is applied to segment unseen food images during the testing phase. This involves:

- **Input preparation:** Images are read, resized to meet model requirements (multiples of 32, maximum side length of 2400 pixels), and properly scaled.
- **Model restoration:** The inference graph loads the trained weights (EMA-applied) to ensure high prediction fidelity.
- **Prediction:** Each image is run through the network to produce per-pixel class probabilities, followed by an argmax operation to obtain the final segmentation map.
- **Visualization:** Predictions are rendered using a specified color map to visually differentiate each food class.
- **Performance tracking:** Inference time per image is tracked and reported.



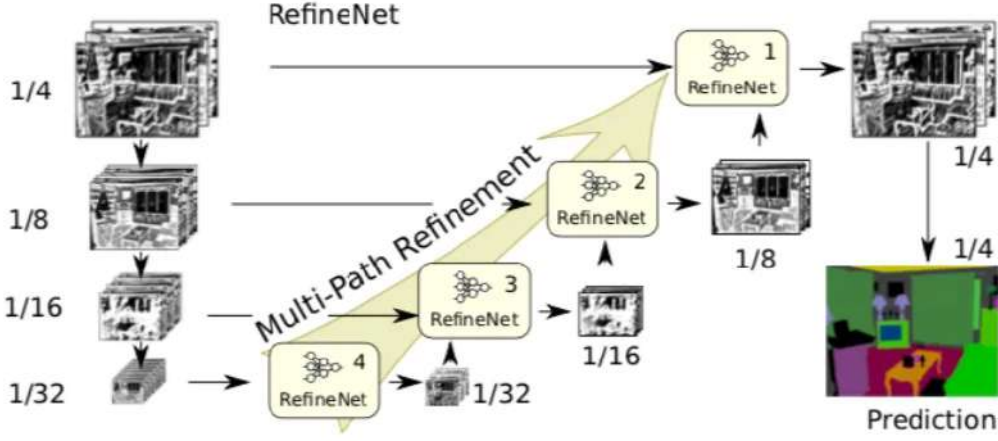


Figure 4.4: RefineNet Architecture

## 4.3 Volume Estimation

### 4.3.1 Image and Annotation Preparation

The pipeline starts with a source RGB food image on a plate and its related JSON file having polygon-based annotation for each of the foods. The annotation includes the vertices' coordinates defining a closed shape surrounding individual foods such as rice, vegetables, and chicken. A label that denotes the category of food is related to every polygon.

The JSON file has a standard form generated by utilities such as LabelMe, having entries within shapes, wherein every shape will have a label and a collection of points — the polygon's vertices. This file is read by the system in order to capture the geometry and semantics to perform segmentation.

### 4.3.2 Depth Estimation Using a Pre-trained Deep Network

A convolutional neural network with the ResNet architecture is utilized to predict per-pixel depth from the input RGB image. The pre-trained ImageNet and fine-tuned monocular depth prediction model comprises an encoder-decoder architecture.

#### Network Inference

The RGB image is fed into the network, producing a 2D depth map of size  $H$  times  $W$ . The raw depth predictions are generally unbounded and hence normalized to

an 8-bit grayscale range for visualization and subsequent processing. Normalization is performed according to the formula:

$$D_{\text{norm}}(x, y) = \frac{D(x, y) - D_{\min}}{D_{\max} - D_{\min}} \times 255$$

where  $D(x, y)$  is the depth estimated in pixel  $(x, y)$ , and  $D_{\min}, D_{\max}$  are the minimum and maximum depth values on the map.

### Output Generation

Two output files are written:

- A grayscale depth image (out\_grey.png) containing normalized depth values.
- A pseudocolored depth image (out\_color.png) based on the JET colormap.

### 4.3.3 Semantic Segmentation via Polygon Masking

Segments are executed in polygons using annotations. For one food item:

- A rasterized binary mask of the polygon is obtained from PIL's `ImageDraw` API.
- The bounding box containing the polygon is calculated to keep processing overhead at a minimum:

$$\text{BBox} = [\min(x), \min(y), \max(x), \max(y)]$$

- For every pixel  $(i, j)$  within this bounding box, OpenCV's `pointPolygonTest` is used to determine whether the pixel lies inside the polygon.
- If inside, the corresponding pixel in the segmentation mask is colored according to the food label using a predefined RGB mapping.

This produces a color-coded segmentation mask identifying all food regions except the plate.

### 4.3.4 Volume Estimation Through Depth Integration

When depth and segmentation data are known, volume is approximated by integrating the depth values over each food region.

Let  $R$  be the set of pixels for a segmented piece of food. The volume  $V$  is calculated as:

$$V = \sum_{(x,y) \in R} \left( \frac{D(x,y)}{255} \right) \cdot a \cdot s$$

Where:

- $D(x, y)$  is the grayscale depth value at pixel  $(x, y)$ .
- The division by 255 scales it back to the original  $[0, 1]$  range.
- $a$  is the physical area per pixel in  $\text{cm}^2$ , camera calibration dependent.
- $s$  is a scale factor that maps normalized depth intensity to height in cm.

The outcome is a measurement in cubic centimeters ( $\text{cm}^3$ ). The process is done for every segmented food item (e.g., rice, chicken, vegetables).

### Calibration Note

In practice,  $a$  and  $s$  have to be experimentally established using calibration data:

- $a$  can be calculated from known sizes of objects within the image or from camera intrinsic parameters.
- $s$  can be estimated by comparing predicted depth intensities with actual height measurements.

### 4.3.5 Output and Logging

The final outputs are:

- A depth visualization (grayscale and color).
- A semantic segmentation mask.
- A text file logging the volume estimation results in  $\text{cm}^3$ .

# Chapter 5

## Algorithms

### **5.1 Food classification:**

Food classification algorithm shown in figure 5.1.

### **5.2 Image segmentation:**

Image segmentation algorithm shown in figure 5.2.

### **5.3 Depth calculation:**

Depth calculation algorithm shown in figure 5.3.

---

**Algorithm 1** Food Classification Using Deep Learning

---

```

1: Input: Labeled food image dataset
2: Output: Predicted food classes and grouped classified images
3: for each image in dataset do                                     ▷ Preprocessing
4:     Resize image to target dimensions (e.g.,  $299 \times 299$  pixels)
5:     Normalize pixel values to a standard scale
6: Split dataset into training and validation subsets
7: Initialize model with pre-defined architecture (Inception-ResNet-V2)
8: Configure model for multi-GPU usage and specify number of output classes
9: repeat                                                             ▷ Training
10:     Feed training images in batches to the model
11:     Perform forward propagation to get predictions
12:     Compute loss between predicted and true labels
13:     Backpropagate loss and update weights
14:     Apply regularization to prevent overfitting
15: until all training epochs are completed
16: Evaluate model using validation dataset
17: for each validation image do                                     ▷ Inference and saving
18:     Generate predicted class and confidence
19:     Assign image to folder corresponding to predicted class
20:     Save predicted image with class label

```

---

Figure 5.1: Food classification Algorithm

---

**Algorithm 1** Semantic Segmentation of Food Images

---

- 1: **Input:** Preprocessed food image dataset, pixel-level annotations, configuration parameters
- 2: **Output:** Trained segmentation model and segmented test images

**Configuration and Initialization**

- 3: Set training parameters: batch size, learning rate, image size
- 4: Define paths for checkpoints, logs, and dataset
- 5: Specify model configuration: number of classes, GPU settings

**Data Preparation and Augmentation**

- 6: **for** each image and mask pair **do**
- 7:     Serialize and store in optimized format
- 8:     Apply augmentations: flip, color distortion, scaling

**Model Architecture**

- 9: Initialize multi-stage segmentation model
- 10: Enable multi-resolution feature fusion
- 11: Set up multi-GPU training support

**Training Process**

- 12: **while** not converged **do**
- 13:     Update learning rate with decay
- 14:     Compute loss: classification + regularization
- 15:     Average gradients across GPUs
- 16:     Apply Adam optimizer updates
- 17:     Update parameters with EMA
- 18:     **if** checkpoint interval reached **then**
- 19:         Save model checkpoint
- 20:         Visualize predictions and write logs

**Inference and Evaluation**

- 21: **for** each test image **do**
  - 22:     Load and resize to fit model (multiple of 32, max side = 2400px)
  - 23:     Restore model weights with EMA
  - 24:     Run inference to obtain class probabilities
  - 25:     Compute segmentation map using argmax
  - 26:     Apply color map for visualization
  - 27:     Save output image and record inference time
- 

Figure 5.2: Image segmentation Algorithm

---

**Algorithm 1** Food Volume Estimation via Depth and Polygon Segmentation

---

```
1: Input: RGB Image  $I$ , annotation JSON  $J$ , pretrained depth model  $M$ 
2: Output: Depth map, Segmentation mask, Volume estimates per food item

3: // Step 1: Preprocessing
4: Load image  $I$ 
5: Load annotation  $J$  and extract labeled polygons  $\mathcal{P} = \{(l_i, p_i)\}$ 
6: Remove entries with label "plate"

7: // Step 2: Depth Estimation
8: Resize and normalize  $I$  for model  $M$ 
9:  $D \leftarrow M(I)$  ▷ Predict raw depth map
10: Normalize depth:  $D_{\text{norm}}(x, y) = \frac{D(x, y) - D_{\min}}{D_{\max} - D_{\min}} \cdot 255$ 
11: Apply color map to  $D_{\text{norm}}$  and save images

12: // Step 3: Polygon Segmentation
13: Initialize mask  $S \in \mathbb{Z}^{H \times W \times 3} \leftarrow 0$ 
14: for each  $(l_i, p_i) \in \mathcal{P}$  do
15:   Rasterize polygon  $p_i$  to binary mask  $M_i$ 
16:   for each pixel  $(x, y)$  where  $M_i(x, y) = 1$  do
17:     Assign color of label  $l_i$  to  $S(x, y)$ 
18: Save  $S$  as segmentation output

19: // Step 4: Volume Estimation
20: for each  $(l_i, p_i) \in \mathcal{P}$  do
21:   Create binary mask  $M_i$  for  $p_i$ 
22:    $R_i \leftarrow \{(x, y) \mid M_i(x, y) = 1\}$ 
23:   Initialize  $V_i \leftarrow 0$ 
24:   for each  $(x, y) \in R_i$  do
25:     Convert  $D_{\text{norm}}(x, y)$  to real depth  $h$  using scale  $s$ 
26:      $V_i \leftarrow V_i + h \cdot a$ 
27: Save volume estimates  $V_i$  with labels  $l_i$ 
```

---

Figure 5.3: Depth calculation Algorithm

# Chapter 6

## Model Results

### 6.1 EfficientNetB0

The result for the EfficientNetB0 model for image classification is shown in Fig. 6.1.

```
Epoch 29/30
618/618 [=====] - 128s 206ms/step - loss: 0.2575 - accuracy: 0.9231 - val_loss: 2.9004 -
val_accuracy: 0.0809
Epoch 30/30
618/618 [=====] - 127s 206ms/step - loss: 0.2658 - accuracy: 0.9161 - val_loss: 3.1832 -
val_accuracy: 0.0827
Evaluating model...
68/68 [=====] - 3s 49ms/step - loss: 3.1832 - accuracy: 0.0827
Test Loss: 3.183178663253784
Test Accuracy: 0.08272058516740799
Model saved as 'final_efficientnet_model.keras'
```

Figure 6.1: EfficientNetB0 accuracy

### 6.2 ResNet50

The result for the ResNet50 model for image classification is shown in Fig. 6.2.

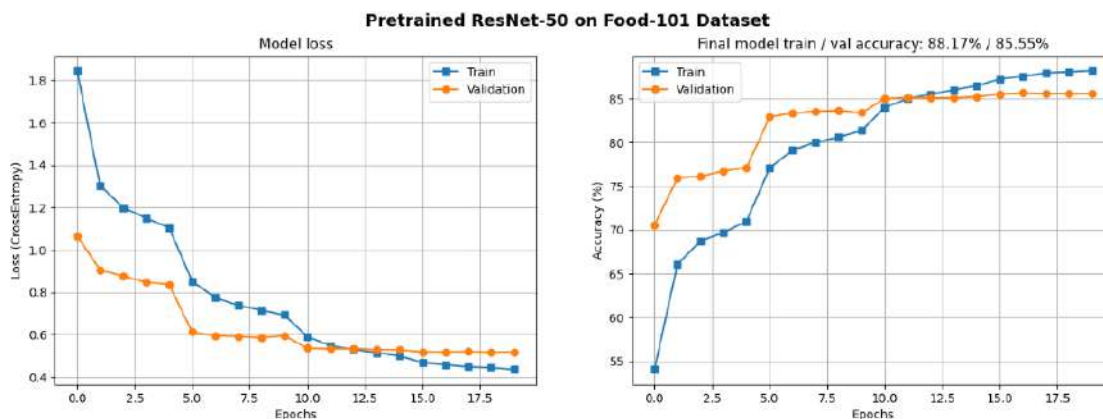


Figure 6.2: ResNet50 model accuracy



## 6.3 Inception-V3

The accuracy result for the Inception model is shown in Fig. 6.3 and the loss result is in Fig. 6.4.

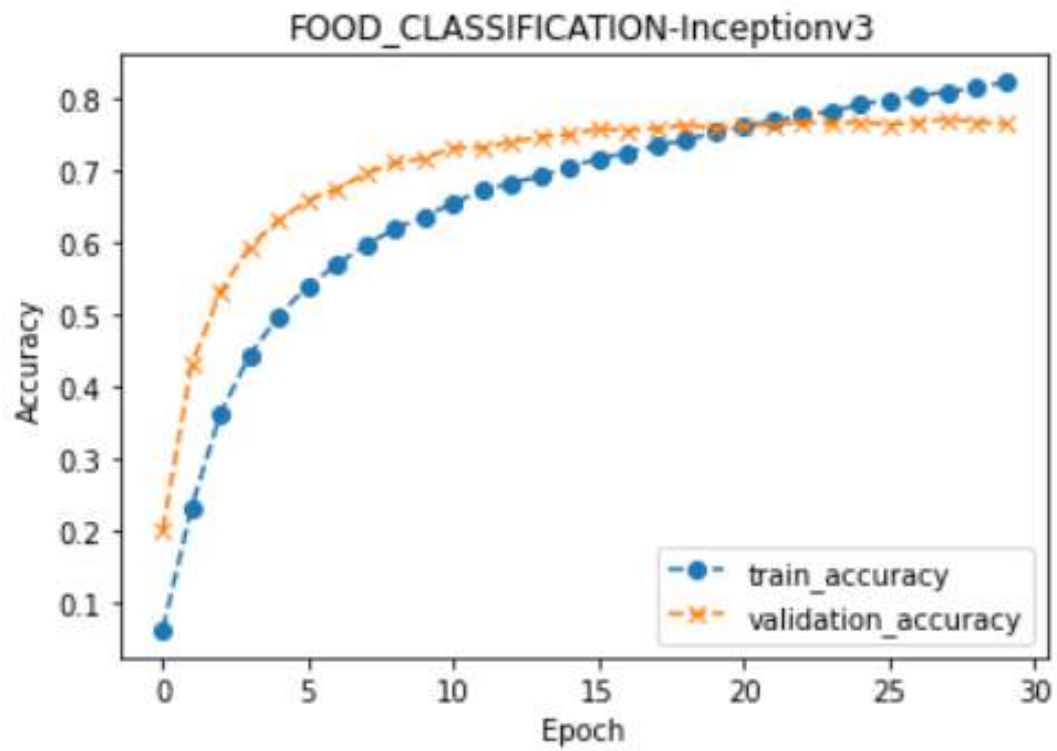


Figure 6.3: Inception model accuracy

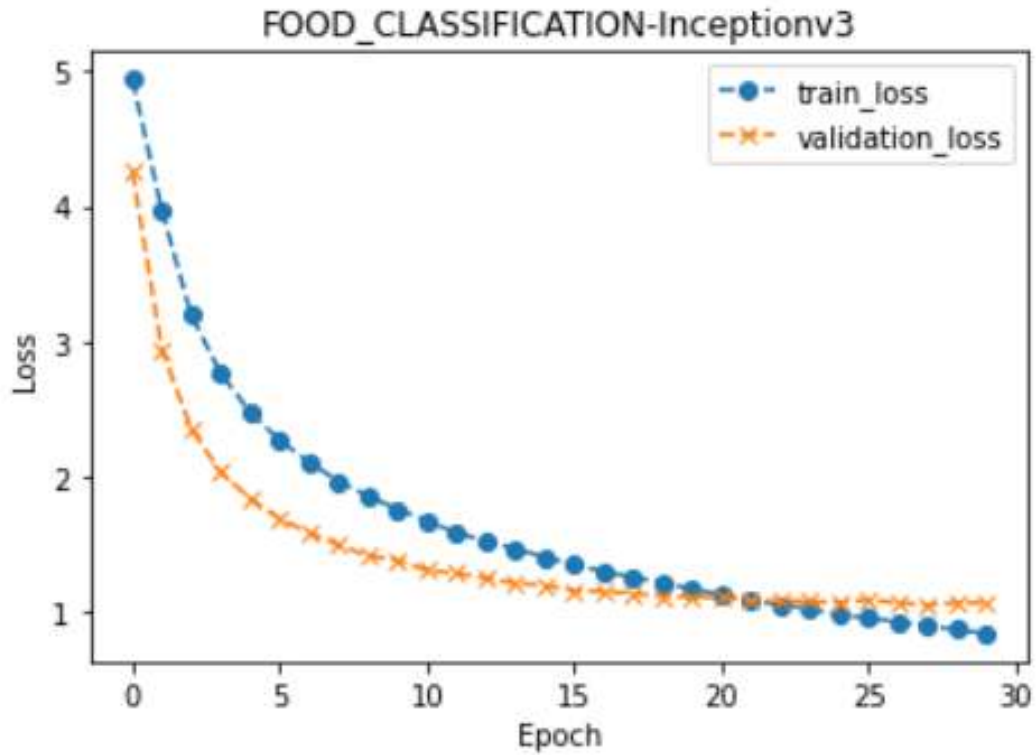


Figure 6.4: Inception model loss

### 6.3.1 Inception Model Intermediate Layers

The following figures show the intermediate outputs of the Inception model:

- Heatmap: Fig. 6.5
- Conv2D: Fig. 6.6
- Batch Normalization: Fig. 6.7
- Activation: Fig. 6.8
- Conv2D-1: Fig. 6.9
- Batch Normalization-1: Fig. 6.10
- Activation-1: Fig. 6.11
- Conv2D-2: Fig. 6.12
- Batch Normalization-2: Fig. 6.13
- Activation-2: Fig. 6.14

- Max Pooling: Fig. 6.15

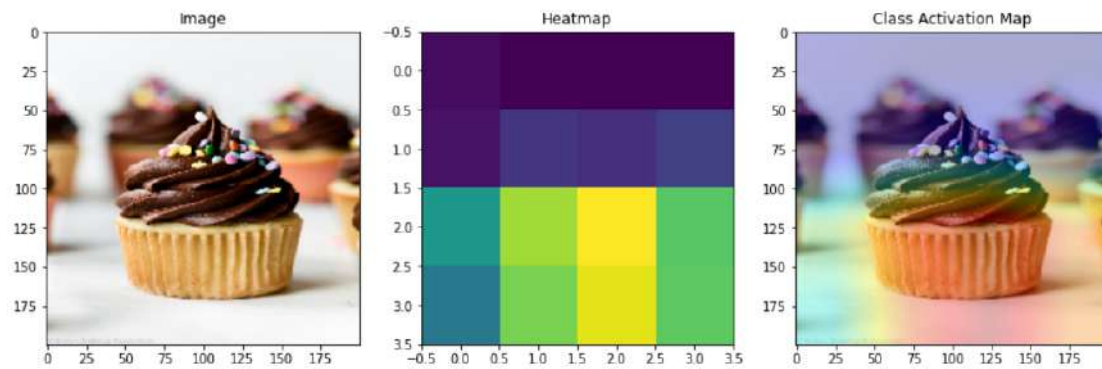


Figure 6.5: Heat map

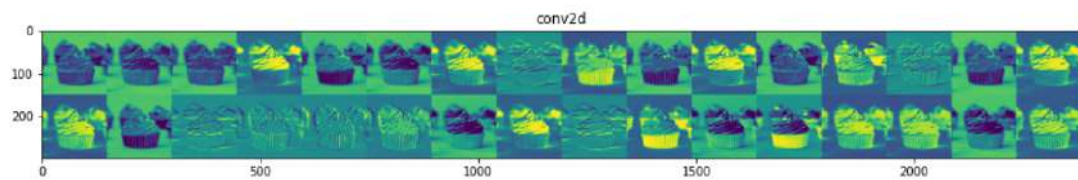


Figure 6.6: Conv2D

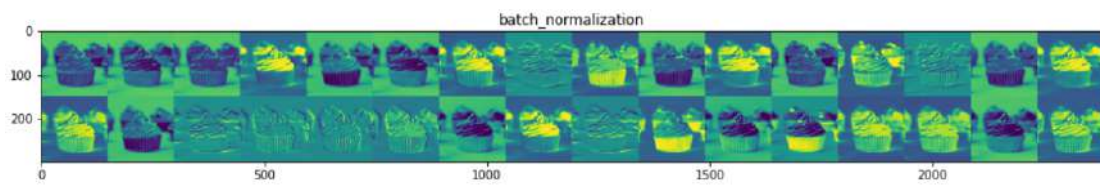


Figure 6.7: Batch Normalization

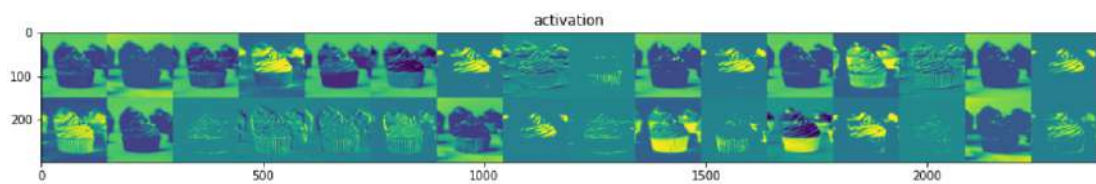


Figure 6.8: Activation

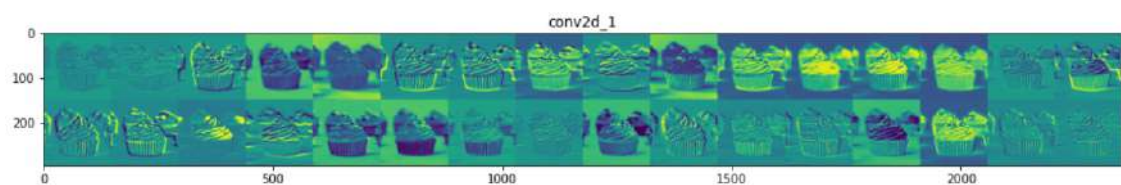


Figure 6.9: Conv2D-1

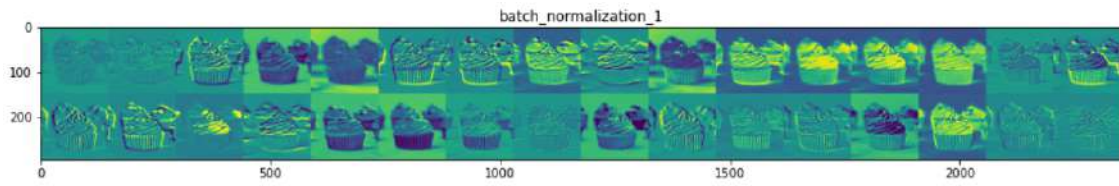


Figure 6.10: Batch Normalization-1

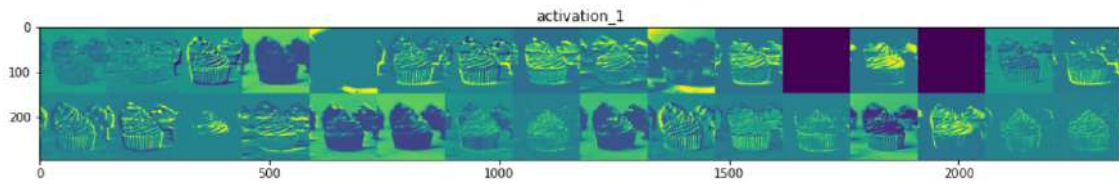


Figure 6.11: Activation-1

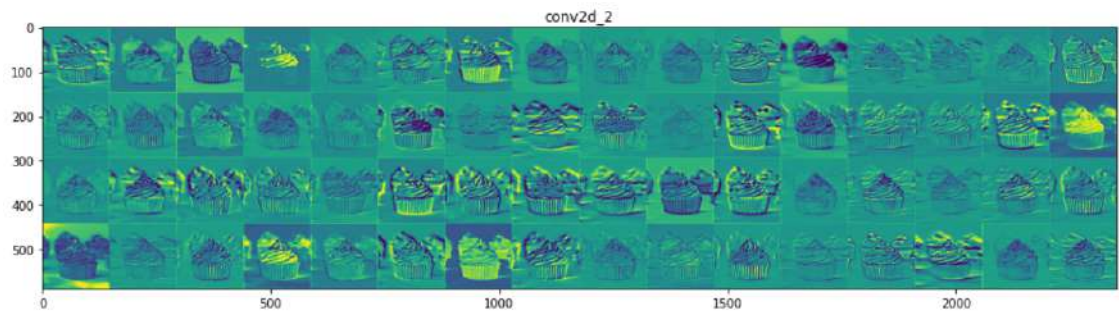


Figure 6.12: Conv2D-2

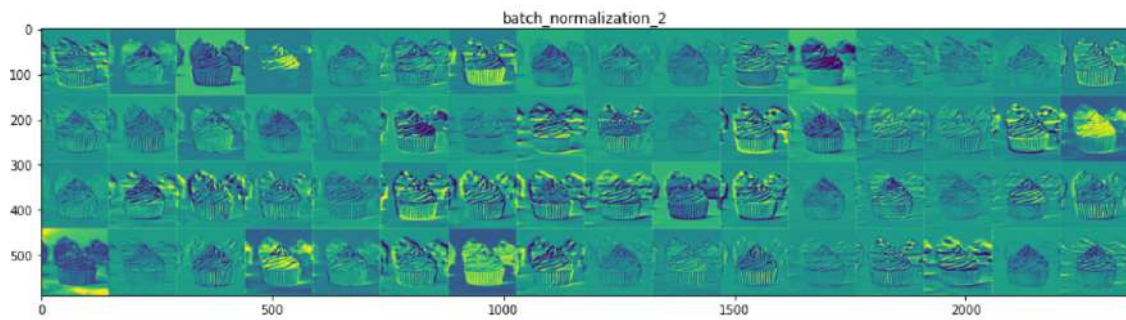


Figure 6.13: Batch Normalization-2



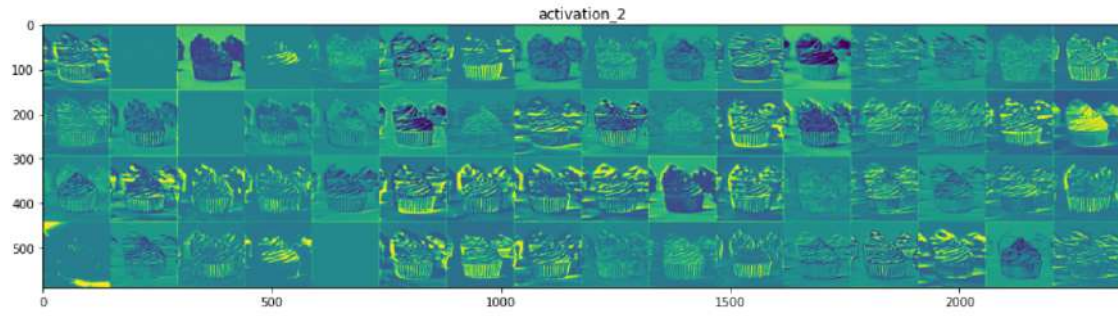


Figure 6.14: Activation-2

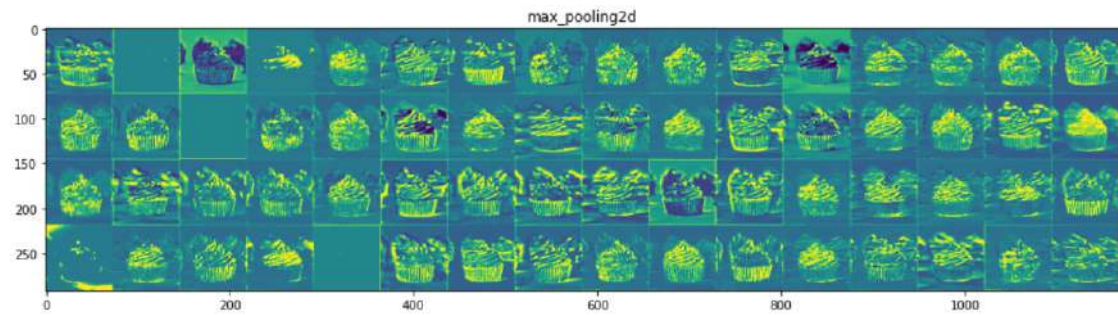


Figure 6.15: Max Pooling

## 6.4 Image Segmentation

The result for image segmentation is shown in Fig. 6.16.



Figure 6.16: Image segmentation output

## 6.5 Volume/Depth Calculation

The result for depth calculation is shown in Fig. 6.17.

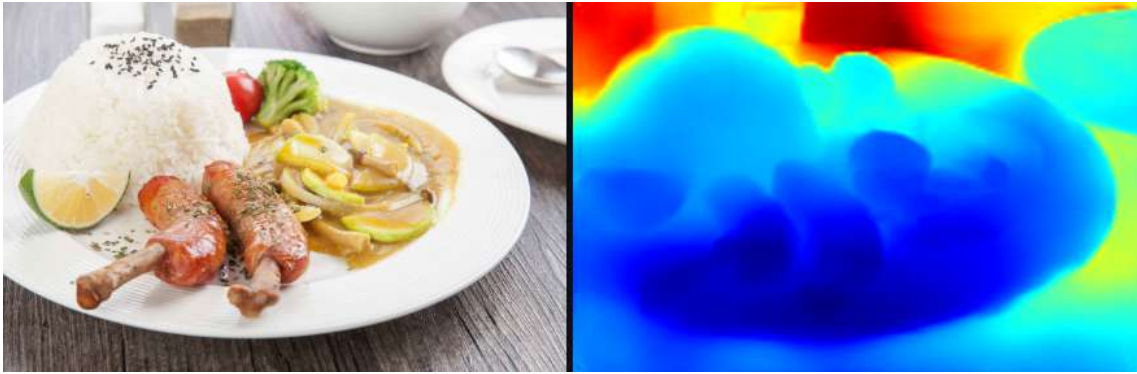



Figure 6.17: Depth calculation result

## 6.6 Calorie Calculation Results

The calorie calculation results are presented in Figs. 6.18, 6.19, 6.20, and 6.21.

### Recognition Results

Your food has been identified!



Name	Probability (%)
hamburger	100.0
donuts	0.0
spring rolls	0.0

More Nutrition Info

Inception-V3 model | Trained on Food101 dataset | Base: 100kg

The image displays 1 item/ with an estimated 500-700 calories for all items and give accurate calories /

Powered by Generative AI

Figure 6.18: Calorie estimation result - Example 1

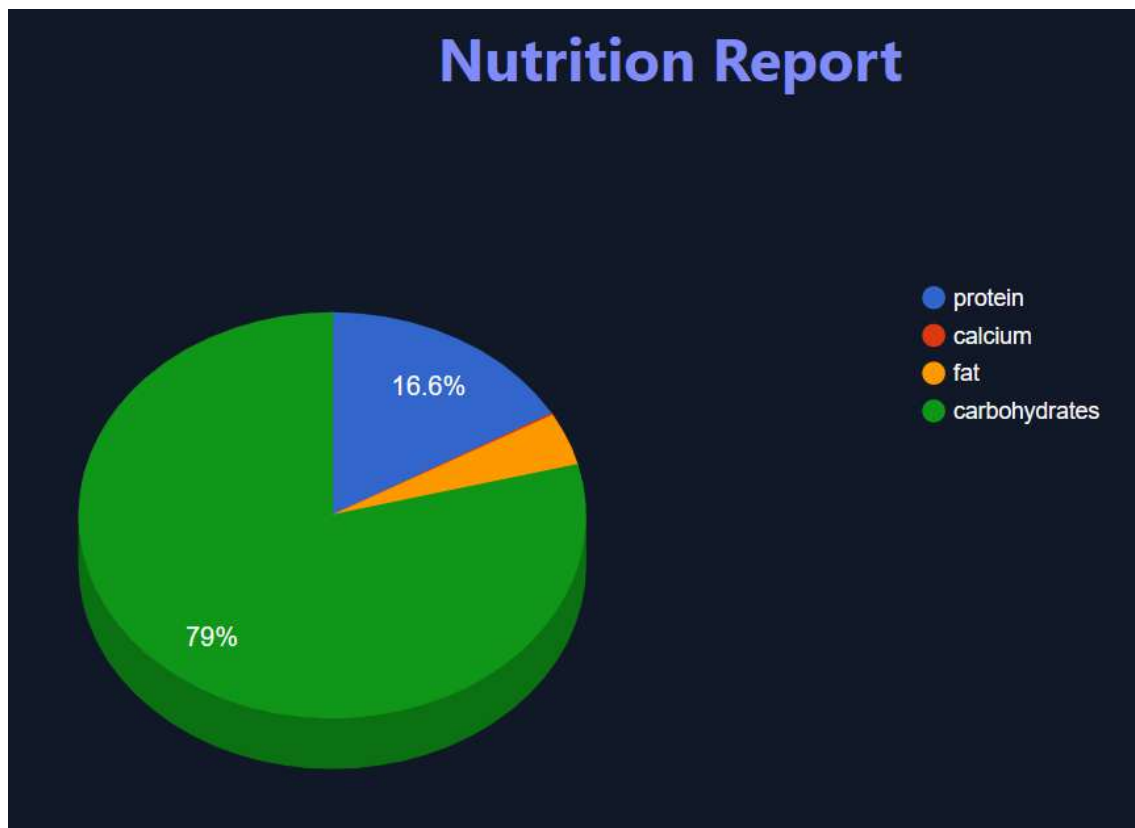


Figure 6.19: Calorie estimation result - Example 2



Figure 6.20: Calorie estimation result - Example 3

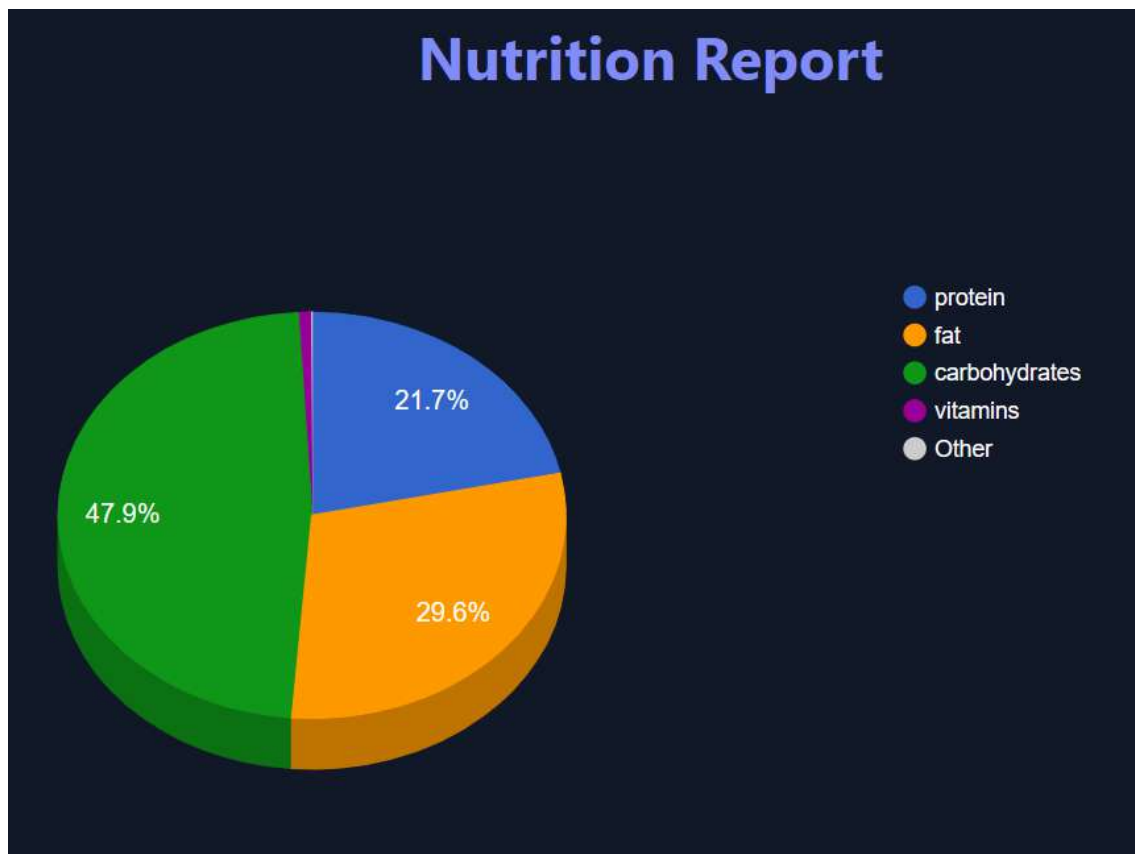


Figure 6.21: Calorie estimation result - Example 4



# Chapter 7

## Conclusion

This work effectively demonstrates a single deep learning pipeline that classifies food, segments food, and estimates food volume from static images. The food classification module is constructed with a fine-tuned Inception-ResNet-V2 model trained on the Food-101 and Recipe-1M datasets. Through the use of pretrained weights and support for multi-GPU training through the tower loss scheme, the system attains high classification accuracy over a broad range of food categories. Dataset preparation, which involves TFRecord conversion, facilitates seamless data loading during training, thus making the pipeline scalable and robust to large datasets. This ground level enables the proper identification of food categories, which acts as an important input for the following stages.

The segmentation step employs the RefineNet architecture for creating sharp masks of the individual foods in a scene. Because publicly available food-specific segmentation datasets are not common, the model is pre-trained on the PASCAL VOC 2012 dataset and fine-tuned afterward with a custom set of manually annotated food segmentation masks. This transfer learning mechanism allows the model to adapt to the food domain with fairly minimal training data. The segmentation outputs are necessary for separating each food region, ensuring both visual interpretability and correct boundaries necessary for volume estimation. The OpenCV, Pillow, and TensorFlow implementation guarantees the system is easy to extend or integrate into other pipelines.

The last step of the pipeline solves the difficult task of estimating food volume from a single RGB image. This is achieved with a monocular depth estimation model from

Hu et al., pre-trained on the NYU-Depth V2 dataset. Coupled with the segmentation masks and a priori reference dimensions (e.g., plate diameter), the depth maps enable real-world volume approximation of each food portion. This approach bypasses the necessity for depth cameras or specialized hardware, thus enabling consumer applications such as mobile dietary tracking. In general, the project proves that a completely automated food analysis system using computer vision and deep learning methods is viable to assist in health monitoring, calorie estimation, and intelligent diet management. There are several avenues for future development, such as increasing segmentation dataset sizes, utilizing newer model structures, or incorporating real-time inference support for mobile or embedded environments.

# Chapter 8

## Future Work

Although the existing system shows excellent performance in food classification, segmentation, and volume estimation, there is much room for improvement and expansion. One of the promising avenues is the incorporation of more sophisticated and newer deep learning models. For example, vision transformers (ViTs) and hybrid CNN-transformer models have performed better on several image understanding tasks and can be investigated for both classification and segmentation. Furthermore, few-shot and self-supervised approaches can potentially lower dependence on extensive labeled datasets, particularly for the segmentation task for which annotation by hand is lengthy and expensive.

Another direction for future work is the generation and utilization of a specialized large-scale RGB-D food dataset. The absence of domain-specific depth data restricts volume estimation precision, particularly in cases where pretraining occurs from non-food datasets such as NYU-Depth V2. Construction of a hand-curated dataset with synchronized RGB and depth images of different types of food, imaged under controlled settings, would greatly enhance depth estimation and calibration. Alternatively, synthetic data creation through photorealistic rendering of 3D food models might also be an effective method for training models towards improved generalization to real-world settings.

Finally, the system could be optimized for real-time and user-interactive usage. Deploying the pipeline on edge devices, such as smartphones or embedded hardware, would require optimizing models for inference speed and memory usage, which could involve techniques like model pruning, quantization, or TensorRT acceleration. Additionally, a

user interface could be designed to enable users to take images, annotate or correct segmentations, and receive immediate feedback on estimated food type and portion size. The addition of context information like user food likes, time of meal, and past consumption could further allow for individualized nutritional analysis and recommendation systems. Such future enhancements would not just make the system stronger and more accurate but also more useful and effective in actual health and lifestyle use cases.

# Bibliography

- [1] J. Patel, G. Lakhani, R. Vaghela, and D. Labana, “Densemobile net: Deep ensemble model for precision and innovation in indian food recognition,” p. 3, 02 2025.
- [2] M. Chun, H. Jeong, H. Lee, T. Yoo, and H. Jung, “Development of korean food image classification model using public food image dataset and deep learning methods,” *IEEE Access*, vol. PP, pp. 1–1, 01 2022.
- [3] S. Aggarwal, A. Sahoo, C. Bansal, and P. Sarangi, “Image classification using deep learning: A comparative study of vgg-16, inceptionv3 and efficientnet b7 models,” pp. 1728–1732, 05 2023.
- [4] K. Dong, C. Zhou, Y. Ruan, and Y. Li, “Mobilenetv2 model for image classification,” in *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pp. 476–480, 2020.
- [5] S. Sharma and K. Guleria, “Deep learning models for image classification: Comparison and applications,” in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 1733–1738, 2022.
- [6] J.-h. Kim, D.-s. Lee, and S.-k. Kwon, “Volume estimation method for irregular object using rgb-d deep learning,” *Electronics*, vol. 14, no. 5, 2025.
- [7] F. P. W. Lo, Y. Sun, J. Qiu, and B. Lo, “Image-based food classification and volume estimation for dietary assessment: A review,” *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 7, pp. 1926–1939, 2020.
- [8] F. Konstantakopoulos, E. Georga, and D. Fotiadis, “A review of image-based food recognition and volume estimation artificial intelligence systems,” *IEEE reviews in biomedical engineering*, vol. PP, 06 2023.

- [9] J. Steinbrener, V. Dimitrievska, F. Pittino, F. Starmans, R. Waldner, J. Holzbauer, and T. Arnold, “Learning metric volume estimation of fruits and vegetables from short monocular video sequences,” *Heliyon*, vol. 9, no. 4, p. e14722, 2023.





# 10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




## Filtered from the Report

- Bibliography
- Quoted Text

## Match Groups

-  **44 Not Cited or Quoted 10%**  
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**  
Matches that are still very similar to source material
-  **0 Missing Citation 0%**  
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

## Top Sources

- 8%  Internet sources
- 4%  Publications
- 6%  Submitted works (Student Papers)