FoodSeg103 Model Training

Craig Lawson S00209542

BSc Hons L8 Computing (SG\_KCMPU\_H08)

ATU Sligo

Shane Gilroy

Date

Contents

[***Chapter 1*** Introduction 5](#_Toc164890256)

[***Chapter 2*** Literature Review/ Research 6](#_Toc164890257)

[1.1 Introduction 6](#_Toc164890258)

[2.1 Object detection 6](#_Toc164890259)

[2.2 Datasets and evaluation metrics 6](#_Toc164890260)

[Datasets 6](#_Toc164890261)

[1.2 Table 1: Some well-known datasets and associated statistics 7](#_Toc164890262)

[2.3 Model trained from dataset 8](#_Toc164890263)

[What is a model? 8](#_Toc164890264)

[2.4 Detection, Classification and semantic segmentation 8](#_Toc164890265)

[Detection 8](#_Toc164890266)

[Classification 8](#_Toc164890267)

[Semantic segmentation 9](#_Toc164890268)

[Figure 1: Examples of stages within object detection 9](#_Toc164890269)

[2.5 Computer vision frameworks 10](#_Toc164890270)

[Detectron2 10](#_Toc164890271)

[TensorFlow 10](#_Toc164890272)

[2.6 Calorific and nutritional value breakdown 10](#_Toc164890273)

[Equation based off volume 10](#_Toc164890274)

[Use of API 10](#_Toc164890275)

[2.7 Similar projects/ Related work: 12](#_Toc164890276)

[Food-Recognition 12](#_Toc164890277)

[Key Features and Methodology 12](#_Toc164890278)

[Relevance to Project 12](#_Toc164890279)

[2.8 Literary Review Conclusion 13](#_Toc164890280)

[***Chapter 3*** Project Management/Methodology 14](#_Toc164890281)

[3.1 Data Acquisition 14](#_Toc164890282)

[Types of Datasets 14](#_Toc164890283)

[Factors that were Considered 14](#_Toc164890284)

[Key Sources for Datasets 15](#_Toc164890285)

[3.2 Data Preprocessing 15](#_Toc164890286)

[Transfer Learning Approach 15](#_Toc164890287)

[Rationale for Transfer Learning 15](#_Toc164890288)

[Implementation of Transfer Learning 16](#_Toc164890289)

[Evaluation of Model Architectures 16](#_Toc164890290)

[3.3 Training and Validation 17](#_Toc164890291)

[Setting Up the Training Environment 17](#_Toc164890292)

[Model Selection and Transfer Learning 17](#_Toc164890293)

[Parameter Tuning 17](#_Toc164890294)

[Validation and Overfitting Avoidance 17](#_Toc164890295)

[Training Process 18](#_Toc164890296)

[Result Evaluation 18](#_Toc164890297)

[3.4 Troubleshooting 19](#_Toc164890298)

[3.5 Integration with Nutritional Database 19](#_Toc164890299)

[3.6 Testing and Evaluation 19](#_Toc164890300)

[Model Evaluation 19](#_Toc164890301)

[True Positives, False Positives, and False Negatives 19](#_Toc164890302)

[Precision and Recall 19](#_Toc164890303)

[F1 Score 19](#_Toc164890304)

[Average Precision (AP) 20](#_Toc164890305)

[Intersection over Union (IoU) 20](#_Toc164890306)

[COCOEvaluator 20](#_Toc164890307)

[3.7 Iterative Improvement 21](#_Toc164890308)

[***Chapter 4*** System Design and Implementation 23](#_Toc164890309)

[4.1 Tkinter GUI Python Overall Architecture 23](#_Toc164890310)

[4.2 User Experience 23](#_Toc164890311)

[4.3 Testing 24](#_Toc164890312)

[4.4 Model Training Environment Architecture 24](#_Toc164890313)

[Overview 24](#_Toc164890314)

[4.5 Training Environment Setup 24](#_Toc164890315)

[Hardware and Software Specifications: 24](#_Toc164890316)

[Data Storage: 25](#_Toc164890317)

[4.6 Model Architecture 25](#_Toc164890318)

[Model Selection: 25](#_Toc164890319)

[Training Parameters: 25](#_Toc164890320)

[4.7 Training Workflow 25](#_Toc164890321)

[Data Preprocessing: 25](#_Toc164890322)

[Annotation Processing: 25](#_Toc164890323)

[Model Training: 26](#_Toc164890324)

[Validation and Hyperparameter Tuning: 26](#_Toc164890325)

[4.8 Challenges and Solutions 26](#_Toc164890326)

[Segmentation Mask Issues: 26](#_Toc164890327)

[Hardware Limitations: 26](#_Toc164890328)

[4.9 Tools, Technologies, and Frameworks 26](#_Toc164890329)

[PyTorch and Detectron2: 26](#_Toc164890330)

[TensorBoard: 27](#_Toc164890331)

[Google Collab: 27](#_Toc164890332)

[Integration and Workflow: 27](#_Toc164890333)

[***Chapter 5*** Discussion 28](#_Toc164890334)

[Python 28](#_Toc164890335)

[Challenges with Segmentation and Bounding Box Data 28](#_Toc164890336)

[Adjustments and Simplifications 29](#_Toc164890337)

[Technical and Conceptual Learnings 29](#_Toc164890338)

[Future Directions 29](#_Toc164890339)

[Concluding Remarks 29](#_Toc164890340)

[***Chapter 6*** Conclusion 30](#_Toc164890341)

[Key Findings 30](#_Toc164890342)

[Limitations and Challenges 30](#_Toc164890343)

[Key Learnings 30](#_Toc164890344)

[Future Work 30](#_Toc164890345)

[***Chapter 7*** References 32](#_Toc164890346)

[***Chapter 8*** Table of Figures 35](#_Toc164890347)

# Introduction

The expanding field of computer vision has made significant strides in interpreting and understanding the visual world. Within this technology, one of the most promising applications is in the identification and analysis of food items through image recognition. This project is centred on the hypothesis that utilizing advanced machine learning models can significantly improve the accuracy of food identification and subsequently provide nutritional insights.

The incentive for this research stems from a growing societal focus on health and nutrition. Accurate dietary tracking is crucial for managing weight, controlling intake of specific nutrients, and understanding eating habits. Currently, the task of tracking and analysing dietary intake requires user input, often requiring manual entry into nutrition tracking applications. This process is cumbersome, error-prone, and often awkward.

Conversely, an automated system that can recognize food items from images and estimate nutritional content holds the promise of revolutionizing this space. With this technology it would allow for users to streamline the dietary logging process but also provide users with immediate feedback on their food choices, potentially encouraging healthier eating habits or for users that may have specific dietary requirements or exemptions.

However, the task of identifying multiple food items within a single image presents several challenges. Food items can vary widely in appearance, not only between different dishes but also in servings of the same dish due to differences in preparation, size, and presentation. Furthermore, food items on a plate often overlap or merge visually, complicating the segmentation and identification process.

This project aims to tackle these challenges by training a deep learning model capable of recognizing and distinguishing between various food items within an image. The model will be trained on a comprehensive dataset of annotated food images, using bounding boxes to delineate individual items. Upon successful identification, the model would have interfaced with a nutritional database API, providing a detailed nutritional breakdown of the meal.

The objectives of this project are assorted as:

* To identify an appropriate dataset with accurately annotated images representing a diverse array of food items.
* To train a convolutional neural network model that achieves high precision in identifying multiple food items from a single image.
* To implement the model that can accurately identify food items and present it in a user friendly graphical user interface (GUI).
* To assess the model's performance and identify areas for potential improvement in both accuracy and efficiency.

By accomplishing these objectives, the project endeavours to make a significant contribution to nutritional information and offer an innovative tool for diet management and health awareness.

# Literature Review/ Research

## Introduction

The intent of this literature review is to provide published evidence outlining various methods, concepts and technologies that will be integrated into this project and the reasoning in which I chose these approaches further backed with published journals and academic papers to justify my choice on the features implemented and choice of datasets and models. I will be outlining how object detection works as well as potential models that can be implemented and a further justification for my choice. I will outline datasets and discuss further on why I chose the one I did whilst comparing my choice to other datasets that were available to me.

The advancement of convolutional neural networks (CNN) has aided object detection’s advancements and have been revolutionized by this development. Most object detection models breakdown each task into different subtasks. This was the catalyst for the technology which has brought us to the level of object detection and the utilization of it in many industries such as healthcare and many other sectors. [1], [2]

Coupled with the ever-growing market of readily available food nutritional databases and APIs that allow for breakdowns of named food items can allow for an accurate estimation of the contents of said food items queried.

## Object detection

Object detection, which is a crucial task in computer vision, involves identifying and locating objects in images or videos. It has numerous applications, including pedestrian detection, autonomous driving, and facial detection [3]. Deep learning-based models have significantly improved the accuracy and real-time performance of object detection [3], [4]. However, challenges continue in uncontrolled environments where the object, such as occlusion and clutter [5] to the image and objects being detected. Various techniques, including machine learning based approaches and Fast R-CNN detector, have been developed to address these challenges [5]. Despite these advancements, there is a need for further research to enhance object detection techniques and address existing gaps [4], [5]. Some issues that can occur with object detection include occlusion, where objects are partially obscured, and cluttered backgrounds that can confuse detection algorithms.

## Datasets and evaluation metrics

### Datasets

#### Food 101

The Food 101 dataset, one of the most extensive publicly available food datasets, comprises 101 food categories with 101,000 images. Each category includes 250 manually reviewed test images and 750 training images, [6] providing a robust foundation for training food recognition models. A study by Abiyev and Adepoju [7] utilized this dataset for automatic food recognition using deep convolutional neural networks with a self-attention mechanism, demonstrating its effectiveness in diverse food recognition tasks.

#### Food Recognition 2022

This dataset has been made to train models to look at images of food items and detect the individual food items present in the image. The data has been collected via an app named MyFoodRepo[8] which has users provide images of their daily food intake. This dataset boasts 39,962 images with 76,491 annotations spread over 498 food classes in the version 2 dataset that AICrowd have released.

#### Food11

Developed by EPFL, the Food11 has 16,643 food images grouped into 11 major food categories. The dataset is split into three categories; training, validation and evaluation which allows for much more usability when it comes to this dataset. Although these categories seem to be broad enough it may not be enough to determine the correct class based on the limited classes it has.[9]

#### Food-475 Database

One of the largest publicly available food datasets, Food-475, with 475 classes and 247,636 images obtained by the merging of four publicly available food databases. This dataset looks promising although there are very little credible academic papers that have utilized this dataset. [10]

#### Food2k

Released as a large scale visual food recognition dataset, Food2k facilitates multiple applications. The dataset allows for multiple uses of the dataset and is quite versatile with its use, allowing for possibilities such as food recognition, image retrieval, cross-modal recipe retrieval to name a few. This dataset was a collaborative effort from China and is sadly not publicly available, although there is a smaller version called Food1k that is publicly available[11].

#### FoodSeg103

Created as a subset of the much larger computer vision dataset Recipe1M [12]. FoodSeg103 [13] has 103 class names that is a well-rounded food model with bounding box annotations as well as segmentation data to train instance or semantic segmented models.

## Table 1: Some well-known datasets and associated statistics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Dataset | Year | Images | Food Category | Classes | Image  Sources |
| Food 101 | 2014 | 101,000 | American Foods | 101 | Crawled from web |
| Food Recognition 2022 | 2022 | 39,962 | Generic | 498 | User acquisition via app |
| Food11 | 2016 | 16,643 | Generic | 11 | Other food datasets |
| Food-475 | 2017 | 247,636 | Generic | 475 | Combination of Datasets |
| Food2k | 2021 | 1,036,564 | Generic | 2000 | Unknown |
| FoodSeg103 | 2021 | 9,490 | Generic | 103 | Other food datasets |

Table 1 Some well-known datasets and associated statistics [14]

## Model trained from dataset

### What is a model?

Models are created after the dataset has been chosen appropriate to the use case, in this project it would be on food related data. After training the dataset the model is the output, which is a computational or mathematical representation that captures the relationships and patterns learned from the dataset. The structure of a model can vary depending on its use case. In the figure below, some model’s structures can be visualised. Parameters are incorporated into the model and are learned during the training process. These parameters are adjusted to minimize the margin of error in the output of processing new unseen data and to give a higher accuracy as well as affecting and improving performance. Some key characteristics that must be taken into account for this project when it comes to creating the model would be:  
Training process:

The model is developed by training it with a chosen dataset. The dataset that shall be used for this project will be comprised of food related data, including images of different food items and their names and depending on the dataset chosen other attributes such as food-types.

#### Structural Complexity:

The structure of the model can significantly vary depending on its use case. For food related data the model might need to account for a wide range of variables such as texture, colour, shape, and possibly contextual information about the food item.

#### Performance Optimization:

The ultimate goal is to minimize the margin of error when processing new, unseen data. This involves refining the model to achieve high accuracy and robust performance, especially in a diverse real-world application where food items can vary greatly.

## Detection, Classification and semantic segmentation

There are three main ways in which an object detection model breaks down a task and they are split into these three headings [15]. These three distinct processes play a unique and critical role for accurate identification and analysis of objects within an image.

### Detection

Object detection stands as a foundational step in the object recognition pipeline. It is the process of locating instances of objects within an image. The method of bounding boxes to specify location of detected objects is employed. Many methods are utilized to achieve this such as Region-based Convolutional Neural Networks (R-CNN) [14] and You Only Look Once (YOLO)[16] have significantly enhanced the accuracy and rate which an object is detected enabling real-time applications such as autonomous vehicles, surveillance, and interactive user interfaces.

### Classification

Once an object has been detected it then moves to the next process, classification. This method involves identifying the category or class of the detected object. The development of CNNs has revolutionized object classification, offering effective accuracy and efficiency. CNNs excel in feature extraction, learning complex patterns in data without the need for manual feature engineering. A study by Hütten et al. [17] provides a comprehensive survey on deep learning for automated visual inspection in manufacturing and maintenance, highlighting the integral role of CNNs in image classification and object recognition. Their findings underscore the effectiveness of CNN applications, further cementing their status as a cornerstone technology in the field of computer vision. Furthering this Active Appearance Models (AAMs) are a sophisticated approach in object detection and classification for applications that require a detailed analysis of object shape and texture. AAMs work by first warping the detected object to a mean shape then estimating the combined modes of variation of the concatenated shape and texture models. This process is essential for achieving an accurate and detailed representation of the object. As a result, AAMs can achieve a much more concise and clearer match to the object, even for deformed objects within a few iterations of the model. This feature is particularly valuable in applications such as medical imaging where precise modelling of anatomical structures is vital. A recent study by Ikeda et al. [18] on cell detection and classification in medical imaging exemplifies the application of such models, demonstrating the high accuracy and reliability of AAMs in complex object detection scenarios.

### Semantic segmentation

Semantic segmentation takes the process of object detection and classification one step further by delineating the precise boundaries of each object. This method is akin to pixel-level classification, where each pixel in the image is categorized into a specific class. Recent advancements with semantic segmentation involve employing Deep Convolutional Networks and Fully Convolutional Networks (FCNs) [19] that are capable of producing detailed and precise segmentations. This process is extremely beneficial to a slew of applications such as in medical imaging, where accurate segmentation of tissues and organs is critical or for the application, utilizing it to segment each of the food items on the plate and classifying them for further breakdown of nutritional information.

### Figure 1: Examples of stages within object detection

A dog and cat sitting in front of a christmas tree

Description automatically generated A colorful cat with a purple and green background

Description automatically generated with medium confidence A cat walking on grass

Description automatically generated

Cat 0.91

Figure 1 Examples of stages within the object detection process. (a) Object detection, (b) Semantic Segmentation, (c) Classification

## Computer vision frameworks

### Detectron2

Detectron2 is the second iteration of a computer vision framework released by FAIR, Facebooks Artificial Intelligence research group. It improves upon the first iteration by moving from Caffe2 to PyTorch. This allows users, to make intuitive tests and change the model to work for many different use cases. Detectron2 stands out for its pre-trained models, which are highly adaptable for various use cases. A recent study by Dharwadkar and Math [20] on the application of Detectron2 for enhanced instance segmentation in agriculture showcases the frameworks capability in processing complex visual data and its adaptability in diverse fields.

#### PyTorch

PyTorch is computer vision program that has been released under the Linux license and allows for a much more accurate and easier application of computer vision which is utilized by many object detection frameworks such as Detectron2. PyTorch is utilized often due to its versatility and widely used open-source machine learning library. It is known for its efficiency, flexibility and user-friendly interface making it an essential tool for deep learning and AI research. A study by Rajput et al. [21] highlights PyTorch’s efficiency and performance in energy consumption during model training stages, underscoring its effectiveness in resource-intensive tasks.

### TensorFlow

TensorFlow is an end to end machine learning platform owned and developed by Google. It allows for a wide range of tasks such as allowing users to create models easily with multiple levels of abstraction to allow for a plethora of use cases with these models [22]. TensorFlow excels in its ability to handle small to large-scale machine learning projects with ease. Abdul Salam et al. [23] demonstrated the effectiveness of TensorFlow in their federated learning model for credit card fraud detection highlighting TensorFlow’s robustness and versatility in handling complex machine learning tasks.

## Calorific and nutritional value breakdown

To determine the calorific and nutritional content of detected food items, it’s crucial to consider various methods and their feasibility. Based on my research these were the two methods that deemed to hold merit to gather the nutritional content based on the information that can be gathered.

### Equation based off volume

Determining the nutritional value and caloric content of food based on its volume was a method I was hoping to pursue and utilize but as there are many variables that have to be taken into account such as density, moisture content and the composition of nutrients in the food solely calculating these without the knowledge of these other factors it would be impossible to have an accurate estimation of the nutrients and caloric breakdown based on a formula[24].

### Use of API

As the use of a formula is simply not possible, I have been researching APIs that can allow a user to send off information based of the identified food in the image, and it can return nutritional breakdown and calorific information on said food item. The following are some notable APIs that allow you to do this:

#### Edamam

Edamam is a nutritional and recipe based API that has a wide database of recipes and nutritional information of food items. This API offers a comprehensive database of nutritional information and recipes. A project by Nguyen [25] utilized Edamam’s API for food identification and nutritional analysis, demonstrating its practical application in a project that is similar to this.

#### Nutritionix

Boasting the largest verified nutrition database, Nutritionix is a popular choice for accessing extensive nutritional data. Its wide user base and substantial food item coverage make it a strong candidate for accurate nutrition tracking.

#### USDA Food Composition API

This API provided by the US department of Agriculture also allows us to search their API for nutritional information. Its high request limit and accuracy make it a suitable option, although the geographic focus might limit its applicability for non-American food items [26].

After careful consideration of the features, limitations, and capabilities of these APIs, I have decided to use Edamam for this project. This API provides the necessary flexibility and comprehensive data needed for accurate nutritional analysis, aligning with the project's requirements and scope.

## Similar projects/ Related work:

### Food-Recognition

#### Project Overview

The Food-Recognition [25] project by Lan Nguyen aims to identify food items from images and provide nutritional information. It leverages machine learning algorithms and integrates with nutritional databases to analyse and estimate the nutritional content of the food items detected in the images.

### Key Features and Methodology

#### Image processing and Object detection:

The project utilizes advanced image processing techniques and object detection models to accurately identify food items in various settings and lighting conditions

#### Integration with Nutritional Databases:

It utilizes APIs like the aforementioned Edamam API to fetch nutritional data for the identified food items. This integration allows for an automated process of deriving nutritional information from the detected food items.

#### User Interface and Accessibility:

The project includes a user-friendly interface that simplifies the process of capturing food images and viewing nutritional information, making it accessible to a broad audience.

### Relevance to Project

#### Methodological insights

The methodologies employed in the Food-Recognition project, especially in terms of image processing and API integration, provide valuable insights that can be adapted or improved upon in my project.

#### Challenges and solutions

The projects’ approach to tackling common challenges in food recognition such as handling diverse food types and presentation styles, can inform our strategies in similar scenarios.

#### Innovative features

Any innovative features or approaches in the Food-Recognition can serve as inspiration for enhancing the functionality and accuracy of my project.

## Literary Review Conclusion

My research on computer vision-based identification of foods stands to benefit considerably by the comprehensively conducted review results. Leveraging Convolutional Neural Networks (CNNs) and other comparable vision models for computers, merged with an intensive understanding and judicious choice of data sets paves the path for advanced object detection tactics in food constituents. This wealth of insight aids not just in unpicking intricacies tied with distinct food recognition and nutritional breakdown, but also supports choosing appropriate strategies and innovations.

The knowledge gleaned from this analysis goes past mere dataset discernment – it provides a profound understanding about potentials coupled with challenges tethered to visual computer field harnessed for dietary studies. In essence, over time pursuing this type of work lays down groundwork paving way forward towards enhancing model exactness besides exploring untrodden paths within the meeting point of AI systems and nutrition science.

Insights gleaned above alongside the tools employed are essential facilitators assisting me in steering my project towards becoming more inventive, streamlined yet feasible while still successfully catering to modern demands encapsulated within dietary assessments.

# Project Management/Methodology

The methodology section will describe the deep learning model selection, the dataset preparation including bounding box annotation [27], and the training process with iterations [27]. It will also cover the challenges faced during model training, such as class imbalance and long training times.

## Data Acquisition

Choosing the right dataset is a crucial step in developing an object detection or semantic segmentation project. It impacts not only the model's accuracy and generalization but also the overall project timeline and feasibility. Here's a high-level guide on what was considered when selecting a dataset for object detection and semantic segmentation projects:

### Types of Datasets

Object Detection Datasets: These datasets contain images annotated with bounding boxes, indicating the location of objects within the image. Popular examples include COCO (Common Objects in Context), and Pascal VOC. Object detection datasets are ideal for projects focusing on identifying and localizing objects in an image without requiring pixel-perfect segmentation.

Semantic Segmentation Datasets: These datasets contain images with pixel-level annotations, indicating the precise boundaries of objects within an image. Examples include Cityscapes for urban scenes and ADE20K, which covers a wide variety of object categories and scenes. Semantic segmentation datasets are suitable for projects where accurate object masks is crucial.

### Factors that were Considered

Relevance to Project Goals: Choosing a dataset that aligns with the project's objectives. As there are very few publicly available food datasets with semantic segmentation included with them this can be quite challenging. Finding datasets with the use of Kaggle streamlined my approach to choosing a dataset as it allowed me to consider the file structure and contents before downloading and manipulating the data myself as well as consider feedback from other developers that have used the selected dataset in a project.

Dataset Quality and Annotation Consistency: The quality of annotations, whether bounding boxes or segmentation masks, affects the model's training and accuracy. Inconsistent or inaccurate annotations can lead to poor model performance. Due to my considerable effort of getting the segmentation data working for FoodSeg103’s dataset, the data quality led to some shortcomings during development.

Size and Diversity of the Dataset: A large and diverse dataset is preferable as it provides more examples for the model to learn from, reducing the risk of overfitting. Consider datasets with varied backgrounds, lighting conditions, and object orientations. Datasets that are heavily weighted in certain categories can often bring down the accuracy of the model as the class weights are not properly proportional throughout.

Availability of Pre-trained Models: Choosing a dataset with pre-trained models, like those from the Detectron2 Model Zoo, can speed up development through transfer learning. Careful consideration was given when selecting the model as some models in the Detectron2 Model Zoo are not applicable to all datasets and can lead to varying results or errors in the training process.

Licensing and Permissions: Ensuring that the dataset chosen has appropriate licensing for the project's use case as it was a public dataset.

### Key Sources for Datasets

Supervisely: This platform offers a wide range of labelled datasets for object detection and segmentation, along with tools for data annotation and visualization and manipulation of datasets using inbuilt functions on their site.[28]

Kaggle: A community platform offering various datasets across multiple domains, including many relevant to object detection and segmentation projects. [29] This site allows you to explore a slew of different datasets and you can see users feedback and comments based on the amount of traffic that dataset has seen.

Hugging Face: A popular platform for machine learning models and datasets, Hugging Face [30] has a diverse collection of datasets with detailed metadata and community support. Similar to GitHub, it allows users to save training model data in repositories and has well documented projects that allowed for further insight whilst developing the model.

## Data Preprocessing

Majority of all the datasets for any use case have a plethora of different file structures and normalization in the format of the dataset. This can make training an arduous task as many similar use case datasets will not be correctly formatted for the selected training platform, in this use case Detectron2. As Detectron2 accepts registration of datasets in a COCO like format you must structure your dataset in COCO or create logic functions to get your dataset in that correct format.

Model Selection

### Transfer Learning Approach

In this project, I employed transfer learning to leverage the knowledge gained by a model pre-trained on a comprehensive dataset. Specifically, utilizing a model architecture and weights from the Detectron2 Model Zoo that were pre-trained on the COCO dataset, a large-scale object detection, segmentation, and captioning dataset. This dataset contains a diverse array of objects and scenes, which provides a robust foundation for feature extraction.

### Rationale for Transfer Learning

The rationale behind using transfer learning:

Feature Reusability: Deep learning models trained on large and diverse datasets develop an ability to recognize a wide range of features. These features can be generic, such as edges, shapes, and textures, which are applicable to many visual recognition tasks. By using a pre-trained model, we can capitalize on these pre-learned features as a starting point, which can significantly improve the model's learning efficiency and performance on our custom dataset.

Training Efficiency: Training a model from scratch requires substantial computational resources and time, especially when dealing with large datasets. Starting with a pre-trained model accelerates the training process since the model has already learned a substantial amount of information. This is particularly advantageous when the available dataset for the new task is relatively small or when computational resources are limited. Although this model could further be refined by training a model from scratch to provide better accuracy with results, the training time and computational power required was not feasible for this project.

### Implementation of Transfer Learning

This process involved fine-tuning the model on the FoodSeg103 dataset, a task-specific dataset consisting of various food categories. During fine-tuning, the final layers of the network were re-trained to recognize the new classes present in FoodSeg103, while earlier layers retained their pre-trained weights, allowing the model to utilize the generic features learned from COCO.

By using this approach, we aimed to achieve high accuracy in food item detection while minimizing the required training time and computational expense. The pre-trained model served as an advanced starting point, imbuing the model with the capacity to generalize from the broader context of object detection to the specialized domain of food segmentation and detection.

### Evaluation of Model Architectures

Looking at Detectron2’s included Model Zoo [31], I considered several models based on their performance metrics available on their GitHub page. The models evaluated included Mask R-CNN, Faster R-CNN, and SAM.

Mask R-CNN: Known for its efficiency in instance segmentation tasks, Mask R-CNN [32] was considered due to its ability to provide precise pixel-wise segmentation alongside bounding box detection. Mask R-CNN is an evolution of Faster R-CNN that allows for semantic segmentation by utilizing masks placed on top of the bounding boxes that was already supported in Faster R-CNN. This feature is particularly useful for detailed analysis of food items in complex images.

Faster R-CNN: A robust model for rapid object detection, Faster R-CNN [33] offers an efficient balance between speed and accuracy, making it suitable for scenarios where real-time detection is valued over pixel-level precision.

SAM (Spatial Attention Module): SAM enhances feature representation by focusing on relevant spatial features in an image, potentially improving detection accuracy in cluttered scenes typically found in food images. This model was considered to be implemented when some of the masks were working correctly and some were not mapping correctly. After initially implementing some tester code into the Google Collab environment, the realization that time constraints were a concern, so the decision was made to not implement this model.

Given the project's needs and constraints, especially around the handling of segmentation data which posed significant challenges, Faster R-CNN was selected. The primary reasons for this choice were its efficiency in handling object detection without the added computational overhead of segmentation tasks, which was critical given the limitations in segmentation data handling observed in preliminary tests as the dictionary that was created of the segmentation masks would often take upwards of 40 minutes to populate and all this data was often not fully correct.

The decision to focus on Faster R-CNN was further substantiated by its proven track record in the Detectron2 community, ensuring a reliable and well-supported framework for our application needs.

## Training and Validation

### Setting Up the Training Environment

Before you begin training, you need to set up an environment favourable to deep learning tasks. This involves:

Hardware: Ensure access to a suitable GPU for training such as the T4 GPU included in the Google Collab Pro environment, as Detectron2 and most deep learning tasks are computationally intensive.

Software: Set up a Python environment, install PyTorch with the appropriate CUDA version if using GPU, and install Detectron2. You can use virtual environments to manage dependencies.

Dataset Preparation: Collect and preprocess your dataset. For Detectron2, this means annotating images in a format it understands, like COCO's JSON format, which includes bounding boxes and segmentation masks for object detection and instance segmentation tasks, respectively.

Configuration File: Use of a configuration file where you specify the model architecture, hyperparameters, dataset paths, solver settings (like the learning rate and optimization method), and any other parameters relevant to your model and training process.

### Model Selection and Transfer Learning

Detectron2 provides a wide array of pre-trained models through its Model Zoo. You can choose a model close to your task as a starting point, leveraging transfer learning to initialize your model with weights that have already learned features from a large and varied dataset such as COCO.

### Parameter Tuning

Parameter tuning involves adjusting the hyperparameters [34] to find the most effective settings for your task:

Learning Rate: Perhaps the most critical hyperparameter, which determines how much the weights of the network are updated during training. Too high can lead to divergence, too low leads to slow convergence.

Batch Size: The number of samples processed before the model is updated. Larger batch sizes provide more stable gradient estimates but require more memory.

Epochs: The number of times the entire dataset is passed through the network. More epochs can lead to better training but also to the risk of overfitting.

Regularization: Techniques like weight decay (L2 regularization) can help prevent overfitting by penalizing large weights.

### Validation and Overfitting Avoidance

To ensure the model generalizes well and isn't simply memorizing the training data the following methods can be employed:

Validation Set: Use a subset of your data not seen by the model during training to evaluate performance after each epoch or set number of iterations. This is often done by splitting the dataset into an even 70/30 split of training data and validation data to test inference on the model, if the dataset does not have its own split of training data separated for training and validation.

Early Stopping: Monitor the validation loss and stop training if it begins to increase, indicative of overfitting.

Data Augmentation: Techniques such as random cropping, rotations, and colour jittering can help the model generalize better by artificially increasing the size and diversity of the training data.

Dropout: Randomly dropping entities (along with their connections) during training can prevent the model from becoming too dependent on any single neuron.

### Training Process

With everything set up, the training process in Detectron2 typically follows these steps:

Model Initialization: Load the pre-trained weights if using transfer learning or initialize the weights randomly if training from scratch.

Training Loop: For each epoch, iterate over the training data in batches. For each batch, perform forward propagation to generate predictions, calculate the loss with a suitable loss function and then perform backpropagation to compute gradients. Update the weights with the chosen optimization algorithm. This was not implemented at the time of writing.

Validation: After each epoch, or at a regular interval during training, run the model on the validation set to monitor its performance on unseen data. This step is crucial for detecting overfitting.

Model Saving: Save the model weights periodically or when there is an improvement in validation performance. This allows for further investigation of each model trained and its ability.

Hyperparameter Adjustment: Based on the validation results, you may return to the parameter tuning phase to adjust hyperparameters.

### Result Evaluation

After training, evaluate the model thoroughly using metrics relevant to your task, such as mean Average Precision (mAP) for object detection.

In summary, training a model with Detectron2 involves preparing your dataset and environment, choosing a model architecture, fine-tuning hyperparameters, and employing strategies to prevent overfitting. Throughout this process, continuous validation ensures that the model is learning to generalize to new data effectively.

## Integration with Nutritional Database

After the model has been successfully trained to the level of precision that is acceptable and is quite robust, implementation with the nutritional API can begin. After creating a user friendly graphical user interface, the implementation of the nutritional database would be quite simple as we can take the food items that were identified on a test piece of data and use those values to search the Edamam database for nutritional information of the food type. Further development of the model could also allow for estimation of volume which could in theory lead to more accurate nutritional information provided by the software.

## Testing and Evaluation

### Model Evaluation

Models in object detection are evaluated based on their ability to correctly identify and classify objects in an image. This involves assessing the model's accuracy, precision, recall, and other related metrics. Here's how these evaluations are conducted:

### True Positives, False Positives, and False Negatives

True Positives (TP): When the model correctly identifies an object that is present in the image. This means the bounding box or segmentation mask accurately aligns with the ground truth.

False Positives (FP): When the model identifies an object that is not actually present, indicating a misclassification or incorrect detection.

False Negatives (FN): When the model fails to identify an object that is actually present, suggesting a missed detection.

True Negatives (TN): Although less frequently used in object detection, this term refers to cases where the model correctly identifies that there's no object when indeed there isn't.

### Precision and Recall

Precision: Defined as the ratio of true positives to the total predicted positives (true positives + false positives). Precision indicates how often the model's predictions are accurate.

Recall: Defined as the ratio of true positives to the total actual positives (true positives + false negatives). Recall reflects how well the model detects all relevant instances.

### F1 Score

F1 Score: The harmonic mean of precision and recall. It is useful for evaluating a model's overall performance, balancing the trade-off between precision and recall.

### Average Precision (AP)

Average Precision: Calculated as the area under the precision-recall curve, typically used in object detection to evaluate models across different confidence thresholds. Higher AP indicates better model performance.

### Intersection over Union (IoU)

IoU: A measure used to evaluate how well predicted bounding boxes or segmentation masks overlap with ground truth. It is calculated as the intersection area between predicted and ground truth boxes divided by their union. IoU is crucial in determining whether a predicted bounding box or segmentation mask is accurate enough to be considered a true positive.

### COCOEvaluator

The COCOEvaluator is an assessment tool integral to the field of computer vision, specifically designed to evaluate object detection algorithms. This evaluator is derived from the Common Objects in Context (COCO) dataset, a large-scale object detection, segmentation, and captioning dataset. The COCO dataset was originally presented in a paper by Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick in 2014, becoming a benchmark in the field.

The primary function of the COCOEvaluator is to provide a standardized method of evaluating the performance of object detection models against the diverse and challenging scenarios presented in the COCO dataset. It calculates several metrics that are critical for determining the efficacy of these models. The main metrics reported by COCOEvaluator are:

Average Precision (AP): This metric evaluates the precision of the object detector at different intersection-over-union (IoU) thresholds. It is often reported as AP@0.5 (IoU threshold of 0.5), AP@0.75, and AP@[.50:.05:.95] (averaged over IoU thresholds from 0.5 to 0.95 with a step size of 0.05)[35]. This allows for a detailed understanding of model performance across varying levels of localization accuracy.

Average Recall (AR): This evaluates the maximum recall achieved by the model at different IoU thresholds and for varying numbers of detections (e.g., AR for 1, 10, and 100 detections). This metric helps in assessing the model's ability to detect all relevant objects while varying the number of detections allowed per image.

Precision-Recall Curve (PR Curve): This is a plot that shows the trade-off between precision and recall for different probability thresholds. This curve is useful for analyzing the performance of detection models beyond just single-number summaries.

These metrics are essential for comparing the performance of different detection systems and are widely used in the development and refinement of object detection algorithms. The COCOEvaluator, by providing these metrics, plays a crucial role in the advancement of research in object detection within the machine learning and computer vision communities as it allows for further understanding of the training process of a model and give a base line of how accurate the model may work with live data.

A screenshot of a computer

Description automatically generated

Figure 2 COCO Evaluator Results based on one of the training iterations of the FoodSeg103 dataset

## Iterative Improvement

During the initial stages of the project, I encountered significant challenges related to the distribution and representation of class instances in the FoodSeg103 dataset. Specifically, some food item classes within the dataset were underrepresented in the validation set, despite being labelled in the training set. This imbalance posed a potential risk for overfitting and poor generalization to real-world scenarios where such food items might appear more frequently.

To address this, I intended to adopt an iterative training approach. Initially, the model was trained on the entire set of available classes, including those with limited representation in the validation dataset. This initial model served as a baseline for the model experiments.

#### Experimentation and Model Adjustment

Realizing the potential limitations of training with underrepresented classes, I intended to conduct a series of experiments where the model was trained without these classes. By excluding classes with minimal or no images in the validation set, I aimed to assess whether the model could achieve higher precision and recall for the remaining classes, which were more adequately represented.

#### Results and Observations

The exclusion of underrepresented classes would have in theory resulted in noticeable improvements in model performance metrics. Using the COCOEvaluator, it can be observed that an increase in Average Precision (AP) and Average Recall (AR) across multiple IoU thresholds. This would be particularly evident in the precision-recall curves, where the model would demonstrate better discrimination between relevant and irrelevant objects in the scene.

#### Reflecting on Model Performance

This experiment would have highlighted the impact of class distribution on model training and validation. It underscored the importance of considering how data representation affects learning outcomes and the necessity of adjusting training strategies accordingly. The improved model would have not only performed better statistically but also showed enhanced practical usability by focusing on well-represented classes.

#### Next Steps

Building on these insights, the next phase of the project will focus on augmenting the dataset with additional images for underrepresented classes, either through data augmentation techniques or by sourcing more diverse data samples. This will allow us to re-integrate these classes into the training process and assess the potential for further performance enhancements.

# System Design and Implementation

## Tkinter GUI Python Overall Architecture

The architecture of the system is designed as an integrated pipeline combining advanced image processing and information retrieval to provide a seamless end-to-end solution for food identification and nutritional analysis. At its core, the system architecture consists of four primary components:

Image Data Ingestion: The system begins with the user uploading an image via the graphical user interface (GUI). This image acts as the input data for the object detection model.

Object Detection Model: Upon image selection, the Detectron2-based model processes the image to detect and classify food items. The model has been trained on the FoodSeg103 dataset and can recognize a diverse range of food categories with high accuracy. I set up Detectron2 to use the CPU of the device to run the model as it would not be that intensive as there is one image evaluated at any one time as opposed to batches which would affect the performance of the user’s machine

Nutritional Database API Interface: Identified food items are then queried against the Edamam Nutritional Database API, which returns detailed nutritional data for each recognized food item.

Data Presentation: Finally, the retrieved nutritional information is displayed to the user through the GUI. This includes a breakdown of calories, macronutrients, and other pertinent dietary information.

Each component is critical to the system's operation, from initial data capture to the final user interaction, forming a coherent pipeline that serves the user's informational needs.

## User Experience

The GUI is the focal point of user interaction, designed with a clear and intuitive layout to enhance user experience. Users start by uploading an image through a straightforward select file button on the interface. The system then displays a visual cue, such as a spinner or progress bar, indicating that image processing is underway. Upon completion, a visual representation of the detected food items, along with a concise nutritional breakdown, is presented. This display utilizes familiar nutritional labels to maximize readability and user understanding.

A screenshot of a menu

Description automatically generated

Figure 3 GUI with image and model selected to show use case of the program

Each interaction point in the GUI has been carefully crafted to ensure that users, regardless of their technical expertise, can navigate and utilize the system with ease.

## Testing

Testing was conducted at multiple levels to ensure robust system performance:

Integration Testing: Components were integrated stepwise, with tests conducted at each integration point to verify data handoffs and interactions.

System Testing: The complete system was then tested end-to-end. Predefined scenarios, including images with single and multiple food items, were used to confirm the system's operational integrity.

## Model Training Environment Architecture

### Overview

This chapter details the environment setup, model architecture, and the workflow adopted for training the object detection model used in the food recognition system. It outlines the computational resources, software dependencies, and the architectural choices that underpin the model's ability to accurately identify and classify food items from images.

## Training Environment Setup

### Hardware and Software Specifications:

Hardware: The model was trained on a system equipped with an NVIDIA Tesla T4 GPU in the Google Collab environment, which significantly accelerated the training process by leveraging CUDA cores optimized for deep learning tasks.

Software: The system operated on Ubuntu 20.04 LTS, with Python 3.8, PyTorch 1.7, and Detectron2. This software stack was chosen for its robustness and wide supported environment in the machine learning community.

### Data Storage:

Training data, sourced from the FoodSeg103 dataset HuggingFace repository [36], was stored on my personal Google Drive to reduce up and downloading times, enhancing the efficiency of iterative training cycles.

## Model Architecture

### Model Selection:

The architecture selected for this project was Faster R-CNN with a ResNet-50 backbone and an FPN. FPN (Feature Pyramid Network) is the method which the model is trained. It combines low-resolution, semantically strong features with high-resolution. This choice was guided by Faster R-CNN's proven capability in handling complex object detection tasks and ResNet-50's balance between accuracy and computational efficiency.

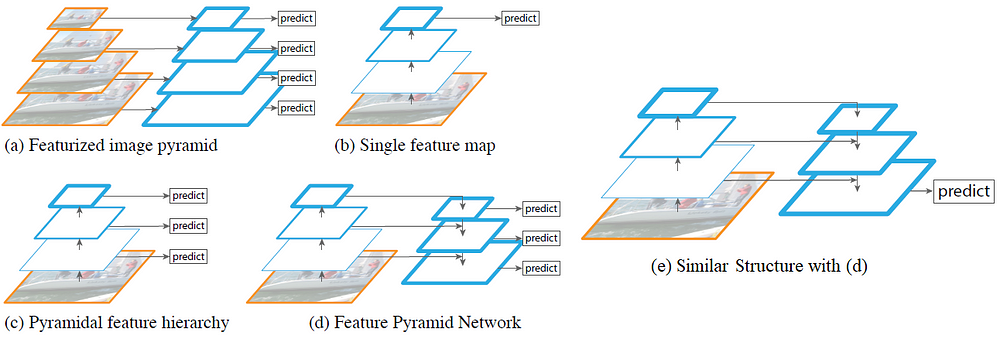


Figure 4 A diagram of different architectures for object detection; FPN shown at article (d)

### Training Parameters:

Learning Rate: Initially set to 0.005 and adjusted via a scheduler that reduced the rate by a factor of 0.1 every 7 epochs, responding to plateauing validation loss.

Batch Size: Set to 512, balancing between computational demand and model performance.

Iterations: Set to 3000, which did not lead to overfitting or convergence.

## Training Workflow

### Data Preprocessing:

Prior to training, images were resized to a uniform scale, and data augmentation techniques such as random flipping and cropping were applied to enhance model robustness and prevent overfitting.

### Annotation Processing:

Annotations in the COCO format were parsed and converted to fit Detectron2's requirements, with particular attention paid to ensuring the accuracy of bounding boxes and segmentation masks.

### Model Training:

Training involved multiple epochs where each epoch processed the entire dataset. Performance metrics such as loss and average precision (AP) were monitored using TensorBoard to track progress and facilitate debugging.

### Validation and Hyperparameter Tuning:

A separate validation set was used to evaluate the model periodically. This process helped in fine-tuning hyperparameters and preventing overfitting by early stopping when necessary.

## Challenges and Solutions

### Segmentation Mask Issues:

Initial challenges with segmentation masks were the logic behind getting the masks into the correct format for Detectron2 as it requires each mask to be stored as a NumPy array. The segmentation masks that were in FoodSeg103 were colour index bitmap images. These are png file type images that mask the food items with specific colour so they can correctly be identified based on their colour code.

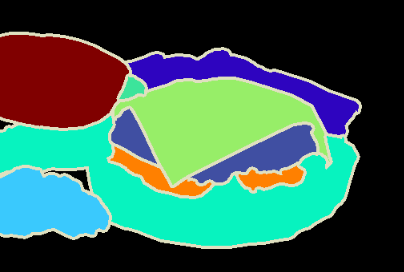
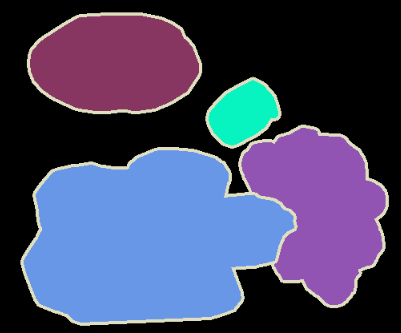


Figure 5 Example of Colour indexed bitmap segmentation masks from FoodSeg103 dataset

### Hardware Limitations:

The high demand for GPU memory was addressed by optimizing batch sizes and utilizing mixed precision training, which allowed the model to train faster and with lower memory consumption.

## Tools, Technologies, and Frameworks

This section details the primary tools, technologies, and frameworks utilized in the development and execution of the food detection system. Each tool was selected based on its ability to facilitate different aspects of the project, from model development and training to performance monitoring and environmental setup.

### PyTorch and Detectron2:

Purpose and Use: PyTorch provided the neural network backend, enabling flexible and powerful computations that are essential for deep learning. Detectron2, built on PyTorch, was specifically used for its state-of-the-art architecture in object detection tasks. This combination allowed for robust model customization, easy integration of pre-trained models, and efficient training workflows.

Advantages: The dynamic computation graph of PyTorch and the modular design of Detectron2 significantly enhanced our ability to prototype quickly, experiment with different model architectures, and fine-tune the parameters on the go.

Challenges: While powerful, Detectron2 requires a steep learning curve to fully utilize its advanced features and optimize performance for custom tasks.

### TensorBoard:

Purpose and Use: TensorBoard was employed as the visualization toolkit for TensorFlow (compatible with PyTorch through the TensorBoard library), which helped in tracking metrics such as loss and accuracy during the training phase. It provided real-time insights into the model's behaviour, which was instrumental in making quick adjustments to the training process.

Advantages: The ability to visually monitor different aspects of the training sessions, such as learning curves and model predictions, helped in identifying overfitting, underfitting, or any anomalies in model training promptly.

Challenges: Integration of TensorBoard with PyTorch required additional setup, and occasionally, the logging of extensive data slowed down the training process if not managed correctly.

### Google Collab:

Purpose and Use: Google Collab was chosen for its cloud-based environment that offers free access to GPUs and TPUs. This platform was particularly beneficial for running high-computation models without the need for local hardware resources. Collab also facilitated easy sharing and collaboration on the project scripts.

Advantages: The ability to access powerful hardware for training complex models without significant investment and the user-friendly notebook interface made Collab an ideal choice for iterative development and debugging.

Challenges: Runtime disconnections and the temporary nature of the virtual machines posed challenges, as long sessions would often reset, leading to loss of unsaved data unless managed with external storage solutions like Google Drive.

Integration and Workflow: The integration of these tools created a cohesive workflow that supported rapid development cycles, performance optimization, and effective problem-solving throughout the project. The synergy between PyTorch’s computational capabilities, Detectron2’s specialized object detection frameworks, TensorBoard’s monitoring features, and Google Collab’s hardware access formed the backbone of the development environment.

# Discussion

This section will interpret the results, discussing potential reasons for low-performing categories and high variance in accuracy [15]. It will delve into considerations for future work, including model architecture tweaks and dataset augmentation [16].

During the course of this project there was a slew of new information that I gathered throughout the research and developmental phase. I believe that, although the AP was = {AJSDKLJSLK} I would like to go back and further refine the dataset for a much more accurate and precise detection of the food items.

As there was both segmentation data and bounding box data included with the dataset chosen, FoodSeg103 allowed for the possibility of segmentation within the model. These segmentation masks were stored as colour indexed bitmap images. As each of these images could have multiple instances of food items in each image the mapping was slightly off when trying to convert each of the polygon’s contours into a NumPy array that would be in the correct format for Detectron2’s registration of the dataset and associated annotations. After getting some of the segmentation data working, I looped through the dataset after registering it to show me each of the entries that did not have any segmentation masks. This block was critical whilst troubleshooting this error as it allowed me to extract a list of the names of each of the images that were incorrectly mapped. I then used these object names to attempt to recalculate the polygons for each of the instances but sadly it did not work. I subsequently, altered my Google Collab environment to not include the code blocks that were created for the use of the binary masks that were needed for the segmentation data which sped up creation time of the model as it did not have to calculate the masks. I suspect one reason why the masks were not being converted would be that the threshold for contours could be higher than is acceptable for these masks in these images. I attempted to recalculate the masks with different values for some variables in the reading of the bitmap images, but it also did not work.

Python

As I am a network stream student I had very little python experience prior to going into this project. To assist in learning the syntax and library use of python I utilized online tutorials on YouTube and documentation hosted by GeeksforGeeks which has great documentation and examples on each section. Alongside these techniques that helped me during the development stages, personal projects that have been utilizing python also allowed me to further hone my skills with Python and some of the popular libraries.

### Challenges with Segmentation and Bounding Box Data

The use of the FoodSeg103 dataset, which includes both segmentation and bounding box data, presented unique challenges and learning opportunities. The segmentation masks, stored as colour-indexed bitmap images, posed significant issues during conversion into the format required for Detectron2’s dataset registration. This conversion problem was particularly evident when attempting to map polygon contours to NumPy arrays, leading to misalignment and mapping errors in the segmentation data.

During troubleshooting, a systematic review of the dataset entries without valid segmentation masks was conducted. This process was instrumental in identifying images with incorrect mappings. Despite efforts to recalculate the polygons for these instances, the adjustments were unsuccessful. Further investigation suggested that the contour threshold settings used during the bitmap conversion process might have been too restrictive, leading to the exclusion of valid data.

### Adjustments and Simplifications

In response to these challenges, the decision was made to simplify the model's computational requirements by excluding binary mask calculations for segmentation from the training process. This modification significantly reduced the model training time but at the expense of potentially richer data that segmentation masks could have provided.

### Technical and Conceptual Learnings

Throughout this project, the steep learning curve of applying theoretical knowledge in a practical, real-world setting became evident. Python, being self-taught through hobby projects, provided a solid foundation however, the complexity of integrating multiple advanced data processing techniques showcased the need for continuous learning and adaptation.

### Future Directions

Looking forward, several strategies are considered to enhance the model’s performance and reliability:

Dataset Augmentation: To address the variance in model accuracy and improve underperforming categories, augmenting the dataset with additional images, especially for those categories, could be beneficial.

Model Architecture Tweaks: Experimenting with different architectures and tuning hyperparameters could optimize performance. Additionally, integrating a more robust method for handling segmentation data might recover the potentially lost insights from the excluded masks.

Advanced Image Preprocessing Techniques: Exploring advanced techniques in image preprocessing to better prepare data for model training, potentially including adaptive threshold techniques for better segmentation mask generation.

User Feedback Integration: Incorporating user feedback into model training, especially from real-world application usage, could guide data corrections and system refinements.

### Concluding Remarks

The challenges encountered underscore the complexity of deploying machine learning models in real-world applications. The insights gained not only from successes but also from the hurdles are invaluable for future projects. Continued exploration and refinement are required to fully harness the potential of machine learning in food recognition and nutritional analysis.

# Conclusion

The project explored the application of deep learning for object detection in food identification and its integration with nutritional databases to provide users with detailed dietary information. Throughout the project, several key objectives were achieved, while some challenges emerged that offer insights into potential future work.

### Key Findings

Model Training and Performance: The use of transfer learning with a pre-trained Faster R-CNN model demonstrated success in food item detection, yielding satisfactory levels of precision and recall. The flexibility of Detectron2 and PyTorch allowed for robust model customization and iterative improvements during training.

Troubleshooting and Problem Solving: The project encountered challenges, particularly with segmentation data and bounding box mappings. Despite these issues, the iterative troubleshooting approach and community resources such as Hugging Face forums and Detectron2 documentation proved invaluable to the success of this project.

User Interface and Integration: The implementation of a simple GUI using Tkinter allowed for user-friendly interaction with the object detection system. Integration with the Edamam Nutritional Database API enabled the retrieval of nutritional information, enhancing the application's utility.

### Limitations and Challenges

Segmentation Data Issues: The conversion of colour-indexed bitmap segmentation masks into a format compatible with Detectron2 proved difficult. This led to the exclusion of some segmentation-related processes, impacting the potential richness of the model's output.

Scope and Time Constraints: The project's timeline limited the scope of what could be achieved, particularly in terms of exploring more complex model architectures and further refining the dataset.

### Key Learnings

The project demonstrated the importance of a flexible approach to model training, recognizing when simplifications are necessary to ensure progress.

The iterative troubleshooting method was crucial for resolving technical issues, emphasizing the value of community resources.

### Future Work

Model Refinement: Future iterations could focus on resolving segmentation issues and exploring additional architectures, such as Mask R-CNN, to enhance precision in object detection and segmentation.

Expanded Dataset: Augmenting the dataset with additional images, particularly for underrepresented classes, would improve the model's ability to generalize.

Advanced Image Preprocessing: Investigating adaptive threshold techniques and other advanced preprocessing methods could lead to better segmentation outcomes.

In conclusion, the project achieved its primary goals, demonstrating the feasibility of deep learning for food item detection and nutritional analysis. While challenges were encountered, the overall success and insights gained lay a solid foundation for future development and refinement in this domain.

# References

[1] F. M. T. R. Kinasih, C. MacHbub, L. Yulianti, and A. S. Rohman, “Centroid-Tracking-Aided Robust Object Detection for Hospital Objects,” in *6th International Conference on Interactive Digital Media, ICIDM 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020. doi: 10.1109/ICIDM51048.2020.9339679.

[2] A. S. Abdullahi Madey, A. Yahyaoui, and J. Rasheed, “Object Detection in Video by Detecting Vehicles Using Machine Learning and Deep Learning Approaches,” in *Proceedings - 2021 International Conference on Forthcoming Networks and Sustainability in AIoT Era, FoNeS-AIoT 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 62–65. doi: 10.1109/FoNeS-AIoT54873.2021.00023.

[3] U. Pundir, “An approach for object detection: Based on image AI.,” *Internation Hournal of Applied Research*, vol. 6, no. 9, pp. 1–4, Aug. 2020.

[4] B. Kaur and S. Singh, “Object Detection using Deep Learning,” in *Proceedings of the International Conference on Data Science, Machine Learning and Artificial Intelligence*, New York, NY, USA: ACM, Aug. 2021, pp. 328–334. doi: 10.1145/3484824.3484889.

[5] K. V. Sriram and R. H. Havaldar, “Analytical review and study on object detection techniques in the image,” *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 12, no. 05, p. 2150031, Oct. 2021, doi: 10.1142/S1793962321500318.

[6] L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101-Mining Discriminative Components with Random Forests”.

[7] R. Abiyev and J. Adepoju, “Automatic Food Recognition Using Deep Convolutional Neural Networks with Self-attention Mechanism,” *Human-Centric Intelligent Systems*, Jan. 2024, doi: 10.1007/s44230-023-00057-9.

[8] MyFoodRepo, “MyFoodRepo,” https://www.myfoodrepo.org/en.

[9] EPFL, “Food Image Dataset,” https://www.epfl.ch/labs/mmspg/downloads/food-image-datasets/.

[10] G. Ciocca, P. Napoletano, and R. Schettini, “CNN-based features for retrieval and classification of food images,” *Computer Vision and Image Understanding*, vol. 176–177, pp. 70–77, Nov. 2018, doi: 10.1016/j.cviu.2018.09.001.

[11] W. Min *et al.*, “Large Scale Visual Food Recognition,” Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.16107

[12] “facebookresearch/inversecooking: Recipe Generation from Food Images.” Accessed: Apr. 28, 2024. [Online]. Available: https://github.com/facebookresearch/inversecooking

[13] “A Large-Scale Benchmark for Food Image Segmentation.” Accessed: Apr. 22, 2024. [Online]. Available: https://xiongweiwu.github.io/foodseg103.html

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2014, pp. 580–587. doi: 10.1109/CVPR.2014.81.

[15] B. Leibe, A. Leonardis, and B. Schiele, “Robust Object Detection with Interleaved Categorization and Segmentation,” *Int J Comput Vis*, vol. 77, no. 1–3, pp. 259–289, May 2008, doi: 10.1007/s11263-007-0095-3.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 779–788, Jun. 2015, doi: 10.1109/CVPR.2016.91.

[17] N. Hütten, M. Alves Gomes, F. Hölken, K. Andricevic, R. Meyes, and T. Meisen, “Deep Learning for Automated Visual Inspection in Manufacturing and Maintenance: A Survey of Open- Access Papers,” *Applied System Innovation*, vol. 7, no. 1, p. 11, Jan. 2024, doi: 10.3390/asi7010011.

[18] K. Ikeda *et al.*, “Staining, magnification, and algorithmic conditions for highly accurate cell detection and cell classification by deep learning,” *Am J Clin Pathol*, Dec. 2023, doi: 10.1093/ajcp/aqad162.

[19] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015, pp. 3431–3440. doi: 10.1109/CVPR.2015.7298965.

[20] N. V. Dharwadkar and R. M. Math, “Enhanced rendering-based approach for improved quality of instance segmentation in detecting green gram (Vigna Rediata) pods,” *Smart Agricultural Technology*, vol. 7, p. 100386, Mar. 2024, doi: 10.1016/J.ATECH.2023.100386.

[21] S. Rajput, M. Kechagia, F. Sarro, and T. Sharma, “Greenlight: Highlighting TensorFlow APIs Energy Footprint”, Accessed: Jan. 26, 2024. [Online]. Available: https://github.com/...

[22] Tensorflow / n.a., “Why Tensorflow,” https://www.tensorflow.org/about .

[23] M. Abdul Salam, K. M. Fouad, D. L. Elbably, and S. M. Elsayed, “Federated learning model for credit card fraud detection with data balancing techniques,” *Neural Comput Appl*, Jan. 2024, doi: 10.1007/s00521-023-09410-2.

[24] P. Wang, J. Huang, J. Sun, R. Liu, T. Jiang, and G. Sun, “Evaluating the Nutritional Properties of Food: A Scoping Review,” *Nutrients*, vol. 14, no. 11, Jun. 2022, doi: 10.3390/NU14112352/S1.

[25] “GitHub - lannguyen0910/food-recognition: 🍔🍟🍗 Food analysis baseline with Theseus. Integrate object detection, image classification and multi-class semantic segmentation 🍞🍖🍕.” Accessed: Jan. 26, 2024. [Online]. Available: https://github.com/lannguyen0910/food-recognition

[26] “FoodData Central.” Accessed: Dec. 06, 2023. [Online]. Available: https://fdc.nal.usda.gov/api-guide.html

[27] S. Li, B. Chang, B. Yang, H. Wu, S. Zhong, and F. Xu, “Dataset Preparation for Arbitrary Object Detection: An Automatic Approach based on Web Information in English,” in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA: ACM, Jul. 2023, pp. 749–759. doi: 10.1145/3539618.3591661.

[28] “What’s Supervisely | Supervisely.” Accessed: Apr. 22, 2024. [Online]. Available: https://docs.supervisely.com/

[29] “Kaggle: Your Machine Learning and Data Science Community.” Accessed: Apr. 22, 2024. [Online]. Available: https://www.kaggle.com/

[30] “huggingface (Hugging Face).” Accessed: Apr. 22, 2024. [Online]. Available: https://huggingface.co/huggingface

[31] “detectron2/MODEL\_ZOO.md at main · facebookresearch/detectron2.” Accessed: Apr. 22, 2024. [Online]. Available: https://github.com/facebookresearch/detectron2/blob/main/MODEL\_ZOO.md

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Trans Pattern Anal Mach Intell*, vol. 42, no. 2, pp. 386–397, Mar. 2017, doi: 10.1109/TPAMI.2018.2844175.

[33] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, Accessed: Apr. 22, 2024. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/results

[34] P. Probst and B. Bischl, “Tunability: Importance of Hyperparameters of Machine Learning Algorithms,” *Journal of Machine Learning Research*, vol. 20, pp. 1–32, 2019, Accessed: Apr. 22, 2024. [Online]. Available: http://jmlr.org/papers/v20/18-444.html.

[35] “detectron2.evaluation — detectron2 0.6 documentation.” Accessed: Apr. 24, 2024. [Online]. Available: https://detectron2.readthedocs.io/en/latest/modules/evaluation.html

[36] “EduardoPacheco/FoodSeg103 · Datasets at Hugging Face.” Accessed: Apr. 28, 2024. [Online]. Available: https://huggingface.co/datasets/EduardoPacheco/FoodSeg103

# Table of Figures

[Figure 1 Examples of stages within the object detection process. (a) Object detection, (b) Semantic Segmentation, (c) Classification 9](#_Toc164889944)

[Figure 2 COCO Evaluator Results based on one of the training iterations of the FoodSeg103 dataset 21](#_Toc164889945)

[Figure 3 GUI with image and model selected to show use case of the program 24](#_Toc164889946)

[Figure 4 A diagram of different architectures for object detection 25](#_Toc164889947)

[Figure 5 Example of Colour indexed bitmap segmentation masks from FoodSeg103 dataset 26](#_Toc164889948)