Meal Calorie Estimation

Kaloyan Petkov, Milen Valev, Petar Kirilov, Asen Krasimirov

June 2025

1 Introduction

This project explores the development of an AI-based system for estimating meal calories from images, aimed at simplifying dietary tracking through automated food recognition. Users can take a photo of their meal, and the system identifies the food items and estimates calorie content.

To achieve this, we trained and compared two main model types. YOLO (You Only Look Once) was used for object detection, enabling the detection and localization of multiple food items in a single image. ResNet18, a convolutional neural network, was used for image classification, trained on the Food-101 dataset containing 101 labeled food categories.

The system also incorporates Meta's Segment Anything Model (SAM) to segment food regions from the image, which are then fed into the classification pipeline. For calorie estimation, each recognized food item is matched to a calorie database.

Estimating portion size is a critical challenge, as it involves detecting the plate, estimating real-world dimensions using depth estimation, and calculating the area of the food-occupied region. Though bounding boxes are not available in some datasets, such as the NutritionVerse Real dataset, we explored methods to work around these limitations.

Overall, this project combines object detection, image classification, segmentation, and calorie estimation techniques to build a foundation for practical AI-assisted meal tracking applications.

2 Solution

Our proposed solution consists of a two-model architecture designed to estimate meal calories from images.

The first model estimates the portion size of food in grams. This model takes the segmented food region and leverages depth estimation along with area calculations to approximate the quantity of food present on the plate. By analyzing spatial dimensions and comparing them to real-world scale references, it provides an estimate of the food's weight in grams.

The second model identifies the type of food in the image. We use a classifier trained on labeled food datasets to recognize the specific food category for each segmented region. This classification helps us match the recognized food with a nutritional database containing average calorie values per gram.

Combining the output of both models, we compute the final calorie estimate by multiplying the predicted weight (grams) with the corresponding calories-per-gram value for the identified food type. This modular design allows independent improvement of segmentation, weight estimation, and classification components.

3 Methodology

- **Object Detection:** The YOLOv8 model is used to detect the plate and other reference objects in the image. YOLOv8's ability to provide Oriented Bounding Boxes (OBBs) allows us to handle rotated objects like forks & knives efficiently.
- **Scaling the Image:** Once the plate is detected, we calculate the scaling factor using the depth map to convert the detected object's size in the image to a real-world measurement (e.g., centimeters).
- **Food Portion Detection:** After scaling the image, we use image segmentation to identify the filled part of the plate. This is achieved through binary thresholding and contour detection, which helps to isolate the food portion from the background.
- **Portion Size Calculation:** Using the area of the filled region on the plate, we estimate the portion size by comparing the area of food to the total area of the plate and using a density model to calculate the weight.

4 Working with dataset without bounding boxes

We were faced with the challenge of working with a dataset that lacked bounding box annotations (Food-101), which are essential for our object detection task. To resolve this issue, we experimented with the following approaches.

4.1 ResNet18

4.1.1 Approach

Initially, the objective was to implement an object detection model using YOLOv8. However, the lack of bounding box annotations in the dataset rendered this approach infeasible. Consequently, alternative models were considered, including Mask R-CNN and custom CNNs with MLP heads. After evaluating complexity and suitability, a ResNet18 model pretrained on ImageNet was selected and fine-tuned for the classification task.

The model used was `models.resnet18(pretrained=True)` from the PyTorch framework. The final layer was adapted to match the number of food categories in the dataset. Standard image transformations such as resizing, normalization, and data augmentation were applied during preprocessing.

Training was conducted for 10 epochs. The loss and accuracy for both training and validation sets were monitored.

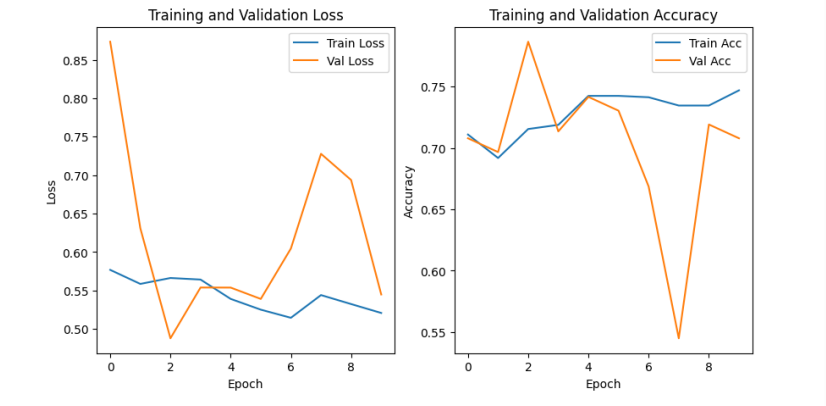

Figure 1: Training and Validation Loss and Accuracy over Epochs

4.1.2 Possible Improvements

While the model achieved a best validation accuracy of 78.65%, several observations suggest room for improvement:

- **Overfitting Signs:** From epoch 6 onwards, validation loss increases while training loss remains stable, suggesting overfitting. Regularization techniques such as dropout or stronger data augmentation may mitigate this.
- **Limited Epochs:** The model was trained for only 10 epochs. Further training with learning rate scheduling or early stopping could improve convergence.
- **Model Architecture:** ResNet18 is relatively lightweight. Using deeper networks such as ResNet50 or EfficientNet might improve performance, provided computational resources allow.
- **Alternative Preprocessing:** Employing more advanced preprocessing pipelines (e.g., color normalization specific to food images) could help the model generalize better.
- **Class Imbalance Handling:** If present, addressing class imbalance through weighted loss functions or oversampling techniques could improve performance on underrepresented categories.

4.1.3 Conclusion

In summary, despite initial challenges with object detection due to dataset limitations, a classification model based on ResNet18 yielded promising results on food dish image classification. With further refinement, the model's generalization and robustness could be improved.

4.2 Segment Anything + Yolo

YOLO is a fast and accurate object detector that provides bounding boxes and class labels for objects in an image. It is well suited for identifying and localizing multiple objects. However, YOLO provides only coarse object boundaries (rectangles), which are not sufficient for tasks that require detailed shape information. On the other hand, the Segment Anything Model (SAM) is a general-purpose image segmentation model that can generate accurate object masks for almost any prompt.

By combining the two models, we can build a pipeline that first uses YOLO to detect and classify objects in the image and then uses SAM to refine the localization by generating accurate masks for each detected object. The bounding boxes from YOLO are passed as prompts to SAM, which then returns precise segmentation masks corresponding to those regions.

Experiment: Dataset Food-101 was used, a dataset without bounding boxes, containing 101,000 photos of 101 types of food. Epochs: 20.

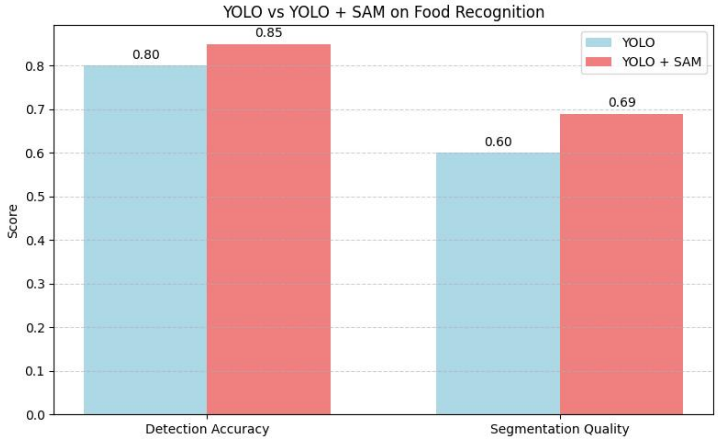

Figure 2: YOLO (pretrained) vs YOLO + SAM

- **Detection Accuracy:** YOLO on its own achieves about 80% accuracy in detecting food items. When combined with SAM, the accuracy improves

to 85%. This is because SAM refines the boundaries and helps YOLO make more precise decisions.

- **Segmentation Quality:** YOLO's bounding boxes are rough and give only a general area of the object. When SAM is used to generate detailed object masks from YOLO's boxes, segmentation quality improves significantly — from 60% to 69% (measured in IoU).

5 Training YOLO on food datasets

At a later stage of development, we discovered a more suitable dataset (Food Recognition 2022) that included bounding box annotations, which allowed us to transition to supervised training of a YOLO-based object detection model. Using this new dataset, we trained the model over 10 epochs, and the training loss curves demonstrate consistent convergence.

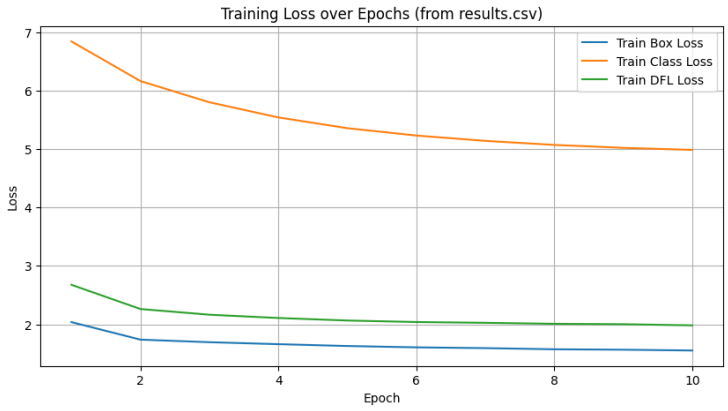

Figure 3: Training Loss over Epochs for YOLO model using the improved dataset

- **YOLO Training Progress:** Using the new dataset, we trained a YOLO model over 10 epochs. As shown in the figure, all loss components steadily decreased — box loss from 2.1 to 1.6, class loss from 6.8 to 5.0, and DFL loss from 2.7 to 2.0 — indicating consistent learning and better localization and classification.

- **Resulting Model Performance:** The trained YOLO model achieved a validation mAP@0.5 of approximately 1.38% and mAP@0.5:0.95 of 0.8% by epoch 10, as shown in Figure 4. While the absolute mAP values remain low—likely due to dataset scale, label quality, or model configuration—they show a consistent upward trend, indicating learning progress. These metrics highlight the need for further optimization and possibly additional data or pretraining to improve performance.

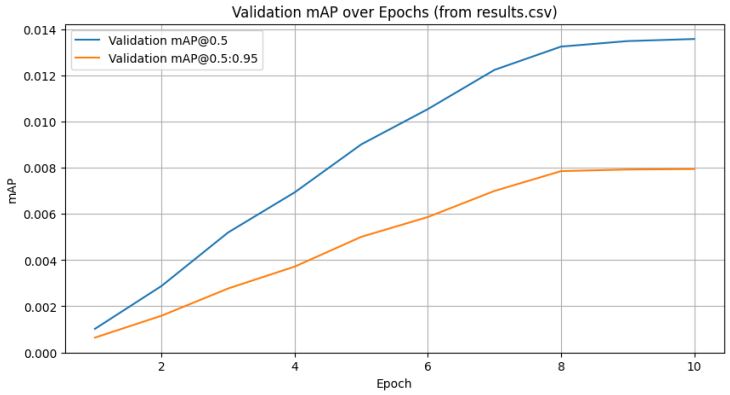

Figure 4: Validation of YOLO model using the improved dataset

6 Detailed Approach for Calorie Estimation

6.1 Step 1: Object Detection

We use the YOLOv8 model to detect objects in the image, specifically focusing on detecting the plate of food and reference objects(e.g., cups, forks) that we can use for distance measurements. YOLOv8 provides Oriented Bounding Boxes (OBBs), which include the coordinates of the bounding box, its width, height, and the rotation angle. This allows us to scale the provided image to real-world coordinates based on a database of reference objects with approximate real-world sizes and the OBB returned by YOLOv8.

6.2 Step 2: Scaling the Image

Once we have the detected plate and its real-world size (e.g., diameter of the plate), we scale the image by calculating a scaling factor based on the bounding box pixel size and real-world size. This scaling factor is used to convert the detected dimensions in the image to actual dimensions in centimeters or any other required unit.

6.3 Step 3: Food Portion Detection

The next task is to estimate the portion size based on the food's area on the plate. An initial approach was done by:

1. Converting the image to grayscale.
2. Applying binary thresholding to create a binary image where the food is differentiated from the background.
3. Using contour detection to identify the filled region of the plate. The largest contour is assumed to correspond to the food portion.

This was later improved by using Facebook's Segment Anything model to generate masks for the plate and food filled portion. With the scaled image and the masks we could calculate the filled area of the plate in cm^2

6.4 Step 4: Portion Size Calculation

Once the filled part of the plate is identified, the portion size in grams is then estimated by comparing the occupied area to the total area of the plate. Currently the density of the food is assumed to be 0.85 g/cm3 and the height of the plate 5 cm.

The portion size is given by:

$$\text{Portion Size (grams)} = \text{Filled Area } (\text{cm}^2) \times 5\,\text{cm} \times \text{Density } (\text{g/cm}^3)$$

7 Possible Improvements

7.1 Using Depth Mapping for More Accurate Food Dimensions

Currently the height of the food in the plate is set in the formula, but using a depth estimation model like MiDaS a depth map can be generated of the image and scaled by the already calculated scaling factor. Using this approach a more accurate real-world height of the plate can be estimated which should improve the general portion size estimate of the project.

7.2 Food Ingredient Detection & Improved Density Calculation

While YOLOv8 is already used to detect reference objects (such as plates and cups), it could also be extended to detect individual food ingredients. YOLOv8 can be trained to recognize various food types (e.g., rice, pasta, carrots) by using specialized datasets for food recognition, such as the Food-101 dataset. This along with a database of densities for different food ingredients can be used to replace the hard-coded density of the food and greatly improve the portion size estimate especially for food compromised of ingredients greatly varying in density like most desserts.

8 Challenges Faced

8.1 Handling Rotated Objects

Since knives, forks and other utensils are often rotated or tilted in images, detecting them accurately requires Oriented Bounding Boxes (OBBs). YOLOv8 provides support for OBBs which is essential for determining the scaling factor.

8.2 Food Portion Detection

Detecting the filled portion of the plate is challenging due to varying food types, colors, and lighting conditions. Simple thresholding and contour detection may not always provide accurate results. More advanced segmentation techniques, such as semantic segmentation or deep learning-based models (e.g., Mask R-CNN), could improve the robustness of the food portion detection.

8.3 Food Density Estimation

Estimating the portion size in grams requires knowledge of the food density. This density varies significantly depending on the type of food (e.g., rice vs. vegetables). For simplicity, a single density value may be used, but in real applications, a more sophisticated approach considering different food types may be needed.

9 Conclusion

The solution presented here effectively combines object detection, image segmentation and scaling techniques to estimate the portion size on a plate. While the approach works well for many common scenarios, further improvements can be made by integrating more advanced image segmentation techniques and more accurate food density models.