Bangladeshi Street Food Calorie Estimation Using Attention Based Improved YoloV8 & Machine Learning Techniques

Aparup Dhar[1], MD Tamim Hossain[1*], Pritom Barua[1]

[1*]Department of Computer Science and Engineering, Premier University, Chattogram, 4000,Bangladesh.

*Corresponding author(s). E-mail(s): thossain3333@gmail.com;
Contributing authors: iaparup@gmail.com; pritombarua71@gmail.com;

Abstract

With the rise of obesity calorie tracking has become an important task for anyone wanting to lead a healthy lifestyle or looking to follow through with a dietary regiment. However, manual tracking is often inconvenient, time-consuming, and prone to human error, highlighting the need for automated solutions. In this paper we propose an innovative approach for tackling this problem specifically targeting Bangladeshi street food. We first construct a diverse dataset composed of popular street foods found in Bangladesh. Then we implement an effective calorie estimation system focused specifically on Bangladeshi street food by improving upon the state of the art vision model YoloV8. Our modified model achieves better results in terms of classification and segmentation with a slight trade-off in computational complexity when compared to base variant. Pairing this with a machine learning regression model we achieve an impressive 6.94 mean absolute error (MAE), 11.03 root mean squared error (RMSE) and 96.0% R^2 Score in calorie estimation making our system highly effective and accurate in calculating real word food calories.

Keywords: Deep learning, Attention Module, Street Food, Calorie Estimation, Machine Learning, Regression

1 Introduction

Obesity is a major public health problem affecting a large portion of the worlds population. It is a disease which is characterized by accumulation of excessive or abnormal amounts of body fat causing possible negative effects on the health of an individual. At a global scale obesity rates have doubled since the year 1990, while as of 2022 approximately 890 million adults can be classified as obese [1]. As such obesity is now considered and regarded as a global epidemic which goes beyond traditional social, economic boundaries as it disproportionately affects both lower and middle-income countries.

As a developing country Bangladesh is currently facing this problem. The country was once synonymous with food insecurity but now deals with a concurrent under-nutrition and over-nutrition situation. Reports from Bangladesh Demographic and Health Survey (BDHS) found that 24% of women and 16% of men are overweight or obese which is a staggering threefold increase since the year 2000 [5]. In urbanized areas, where the majority of people lead a sedentary lifestyle and regularly consume street food on a daily basis obesity rates surpasses 30% [6]. Processed snacks, fried foods, and sugary drinks now account for 40% of urban calorie consumption [7].

Effective obesity prevention requires accurate calorie monitoring helping users to keep track of their food intake. The Centers for Disease Control and Prevention (CDC) recommends keeping track of food and beverage intake in-order to ensure amount of calories do not exceed the body's daily needs [8].

A lot of progress has been made on research relating to food calorie estimation. With the advancements in computer vision and deep learning technologies novel methods have been applied in-order to achieve accurate calorie measurements. Although many solutions exist most are focused on western cuisine, only providing a constant caloric value for food items without considering actual portions of the food. As of now no work specifically address this problem in the context of Bangladeshi food, specifically street food which is consumed by a major part of the working class population.

In this paper we introduce a solution to accurately estimate the calorie content of Bangladeshi street food from a singular image which considers the actual food portion and weight by using coin as a reference object to determine real world dimensions of food. Additionally, we modify the base YoloV8 state of the art object detection, segmentation model by introducing Coordinate Convolution and Convolutional Block Attention Module to gain improved performance on our dataset constructed specifically for this research based on the most popular street foods of Bangladesh.

2 Literature Review

Recent years have seen extensive research on food calorie estimation, with many approaches leveraging convolutional neural networks (CNNs) for food detection and classification. Jiang et al. [10] proposed a deep CNN using a Region Proposal Network from Faster R-CNN and VGGNet for feature extraction and classification. Although efficient due to a regression module for localization, their assumption of a constant 400g weight per item undermines accuracy due to variable portion sizes.

Mask R-CNN [12] has also been widely adopted for object detection and segmentation in calorie estimation, with studies [13], [14] combining segmentation with linear regression for calorie prediction. However, these systems often depend on strict image capture protocols (e.g., fixed height), limiting usability.

To address scaling issues, Akpro et al. [15] introduced chopsticks as a reference object to estimate food volume. Despite being practical, this limits applicability to East Asian cuisine and requires manual food classification. SCalE [16] estimated volume using distance-based calculations, pixel-to-centimeter ratios, and K-means clustering for segmentation. However, the system's reliance on user-specific measurements (e.g., arm length) and camera angles reduces flexibility and introduces potential errors.

A novel method by [17] utilized a mobile device's microphone and speaker with MLS signals for non-invasive vertical distance measurement. This two-image (top and side view) approach performed well but was sensitive to noise and required food to be in containers.

Steinbrener et al. [18] proposed using monocular video and smartphone inertial data for metric volume estimation, combining Mask R-CNN outputs with LSTM-based processing. While achieving competitive accuracy, the method's reliance on video input and lack of real-world robustness evaluation are noted limitations.

Finally, NIRSCam [19] introduced a mobile near-infrared sensing system for calorie estimation, outperforming image-based methods in accuracy and environmental robustness. It uses commercial LEDs for NIRS and three ML models to estimate macronutrients. However, the need for external hardware and high computational complexity may hinder widespread adoption.

3 Methodology

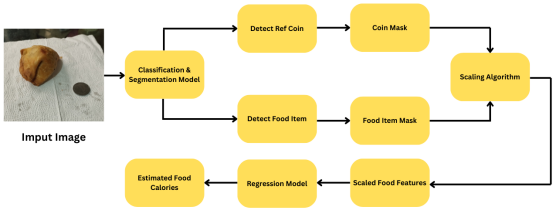


Fig. 1: Proposed System For Calorie Estimation

The proposed food calorie estimation system comprises three main components. First is the

classification and segmentation module, which detects and identifies food items while segmenting them from the background. Next, the scaling algorithm adjusts the food item dimensions based on a reference object to determine real-world measurements. Finally, the calorie estimation model employs machine learning regression techniques to predict calorie values using the extracted features. Each component is detailed below.

3.1 Classification & Segmentation Model

The proposed system takes a single RGB image as input and performs classification and instance segmentation on each detected food item. Instance segmentation generates binary masks, which are later used for feature extraction. For this task, we build upon the state-of-the-art YOLOv8 model [20], an evolution of the widely-used YOLOv5 architecture.

YOLOv8 introduces several key improvements over its predecessor, most notably anchor-free detection, which enables direct object localization, leading to faster training, more precise detection, and better performance. It also features an enhanced backbone based on a refined CSP-Net architecture [21], improving feature extraction capabilities. As a single-stage object detection and segmentation model, YOLOv8 performs predictions in one pass, making it more efficient and less computationally intensive than two-stage models like Mask R-CNN, as used in [13], [14].

Studies [22], [23] have shown that YOLOv8 outperforms Mask R-CNN in terms of segmentation precision, recall, and inference speed—making it ideal for real-time applications. In our implementation, we enhance the base YOLOv8 by replacing the standard "C2f" module in the head with a custom "C2f_CD" module, incorporating Coordinate Convolutional layers [24] and a Convolutional Block Attention Module (CBAM) [25]. These additions aim to improve detection of irregularly shaped food items and those with context-dependent appearances. Further architectural details are discussed in the following section.

3.2 Scaling Algorithm

To accurately estimate food calories, it is essential to determine the real-world dimensions of food items. However, images captured from varying angles and distances can distort these dimensions, leading to significant errors in calorie estimation. To address this, our system incorporates a reference object of known size to calculate a scaling factor that converts image-based measurements into real-world units.

We propose the use of a standard coin as the reference object. After evaluating alternatives, such as teaspoons (which vary widely in design despite having a standard volume), photographic reference scales (accurate but impractical for casual users), and context-specific objects like chopsticks, hands, or thumbs (which suffer from cultural or individual variability), the coin emerged as the most balanced solution. It offers accessibility, standardization, and ease of use in everyday scenarios.

In our implementation, we use a Bangladeshi 5 Taka coin, which has a known diameter of 25.5 mm. The system detects the coin in the image, computes its pixel dimensions, and derives a scaling factor. This factor is then applied to the segmented food items to convert extracted features into real-world measurements. The specific methodology for calculating this scaling factor is detailed in the following section.

$$\text{Height of Food Item} = F_h \text{ px}$$
$$\text{Width of Food Item} = F_w \text{ px}$$
$$\text{Diameter of Ref Coin} = D_c = 25.5 \text{ mm}$$

Scale factors for height and width are given by:

$$\text{Scale Factor (Height)} \quad S_h = \frac{D_c}{F_h} \tag{1}$$
$$\text{Scale Factor (Width)} \quad S_w = \frac{D_c}{F_w} \tag{2}$$

Thus, the overall scale factor is:

$$S_f = \frac{S_h + S_w}{2} \tag{3}$$

3.3 Estimation / Regression Model

The final step in the system is calorie estimation, which is performed using a machine learning regression model. Once the binary mask of

the food item is obtained from the segmentation model, we extract key features—height, width, area, perimeter, and class label—using OpenCV's contour feature functions [31]. These features are then scaled using the previously calculated scaling factor to reflect real-world dimensions. The scaled features are organized into a structured data frame and passed into the regression model, which outputs a continuous value representing the estimated calorie content of the food item.

4 Dataset Description

For the proposed system we require two datasets. One for training the classification, segmentation model and a separate dataset with food items along side reference object coin for constructing our regression dataset. Both datasets are separately discussed in detail below.

4.1 Classification/Segmentation Dataset

Although many datasets exist for Western and Asian foods, to our knowledge there is no dataset focused on Bangladeshi cuisine that includes instance segmentation masks alongside a reference object. To address this, we created a dataset based on popular Bangladeshi street foods [32], [33], comprising six classes: Singara, Somusa, Puri, Peaju, Beguni, and a reference object class for the coin. Images were collected both from the web and onsite, ensuring diversity. Web images offer a wide variety of environments and contexts, while physical data captures real-world challenges like poor lighting, occlusions, and unusual angles, improving the generalization ability of the segmentation and classification model.

(a)

(b)

Fig. 2: Sample Images From Dataset (a) Images Collected From Web (b) Images Collected Onsite

The base dataset contains 1,847 images, which was expanded through augmentation to 3,885 images. These are split 80% for training (3,146 images), 10% for validation (370 images), and 10% for testing (369 images). The total number of instances per class is shown in Table-1, and sample images are presented in Figure-2.

Table 1: Class Instance Distribution

Class	Train	Test	Valid
Singara	467	172	136
Somusa	567	185	173
Puri	415	149	144
Peaju	764	271	287
Beguni	603	207	203
Coin	469	162	161

4.2 Estimation / Regression Dataset

For the regression dataset, we prepared a separate set of images which were passed through the trained model to extract features like height, width, area, perimeter, and class. The dependent variable, calories, is calculated by multiplying the food's weight (measured in grams with a digital scale) by its caloric density. The calorie calculation formula is shown below.

$$C = W \times D \tag{4}$$

where:

C = Calories
W = Weight of food (grams)
D = Caloric density of food (kcal per grams)

The caloric density of every food item used in the research is determined according to an online nutritional database [26]. The specific values are shown in Table-2 below. Caloric density has been considered for standard variants of each food item. For example, in the case of Singara, commonly used ingredients such as potato and peas have been considered as a baseline. Alternative variants using meat or other ingredients have not been considered in this analysis.

Each food item was placed beside the reference coin and photographed from 10 different

Table 2: Calorie Density Values By Class

Class	Calorie Density
Singara	2.61
Somusa	2.11
Puri	2.44
Peaju	1.18
Beguni	1.48

angles to capture visual variations. Although the appearance changes, the calorie content remains constant. Figure-3 illustrates example images of a single food item from multiple views.

Fig. 3: Example of Multiple Images Taken From Different Angles of a Singular Food Item

After passing each image through our scaling algorithm, we obtain the features, which are appended with caloric values in a comma-separated values (CSV) file. After preprocessing, we obtain our regression dataset comprising 644 records. The distribution of classes in the regression dataset is shown below in Table-3.

Table 3: Sample Image Distribution

Class	Records
Singara	128
Somusa	131
Puri	116
Peaju	133
Beguni	136

5 Preprocessing Steps

The following preprocessing steps were applied to improve the quality of our data.

5.1 Image preprocessing

Image preprocessing enhances quality and standardizes input for deep learning. Two primary steps are applied:

5.1.1 Image Augmentation

To improve generalization and mitigate overfitting, we augment only training data via horizontal flips and 90° rotations. This increased training samples from 1,847 to 3,146 images.

5.1.2 Image Resizing

All images are resized to 640×640 pixels to ensure consistent input shape, reduce computational load, and avoid feature distortion.

5.2 Data Preprocessing

To prepare data for regression modeling, the following preprocessing steps are applied:

5.2.1 Encoding

We use one-hot encoding, where each class is represented as a binary vector. Through this we convert the categorical class names into a numeric format the model can understand.

5.2.2 Feature Scaling

Min-max normalization ensures that features with larger magnitudes (e.g., area,perimeter) do not dominate the learning process.

5.2.3 Z-Score Normalization

To eliminate outliers from mispredictions or segmentation errors, we apply Z-score filtering with a threshold of 2.

6 Model Architecture

We improved the base YoloV8 model by introducing an improved C2f_CD modular using Coordinate Convolution[24] in the backbone and using Convolutional Block Attention Module (CBAM) in neck of model.

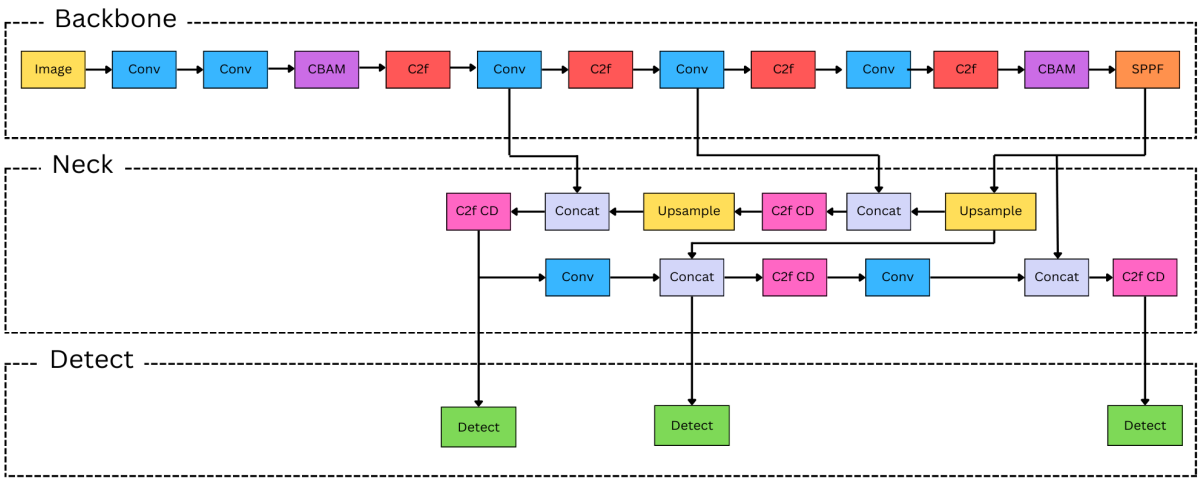


Fig. 4: Improved YoloV8 Model Architecture

6.1 C2f_CD Module

While the C2f module in YOLOv8 outperforms the older C3 module from YOLOv5 in efficiency, it still struggles with tasks where spatial positioning is critical. This is especially relevant for Bangladeshi street food classification and segmentation, where items often appear in consistent regions with subtle visual differences. Challenges like occlusion, overlap, or clustered arrangements also highlight the need for better spatial understanding—something standard convolutions lack. Coordinate Convolution addresses this by adding normalized x and y positional channels to the input tensor, enabling the model to learn spatially-aware features and improving performance in tasks that rely on precise location cues.As a result Coordinate convolution enables networks to better correlate features with their positions, enhancing their ability to handle transformations, localization, and context dependent patterns.

Given an input tensor X of shape (C_{in}, H, W), Coordinate convolution introduces explicit coordinate channels:

6.1.1 Horizontal Coordinate Map

Encodes the normalized x-coordinates across the width.

$$X_{\text{coord},x}(i,j) = \frac{2i}{W-1} - 1 \tag{5}$$

6.1.2 Vertical Coordinate Map

Encodes the normalized y-coordinates across the height.

$$X_{\text{coord},y}(i,j) = \frac{2j}{H-1} - 1 \tag{6}$$

These coordinate channels are concatenated to the input tensor:

$$X' = \text{concat}(X, X_{\text{coord},x}, X_{\text{coord},y}) \tag{7}$$

where X' has shape $(C_{\text{in}} + 2, H, W)$.

The standard convolution operation is then applied:

$$Y = \sigma\left(W * X' + b\right) \tag{8}$$

As a result Coordinate convolution enables networks to better correlate features with their positions, enhancing their ability to handle transformations, localization, and context dependent patterns.

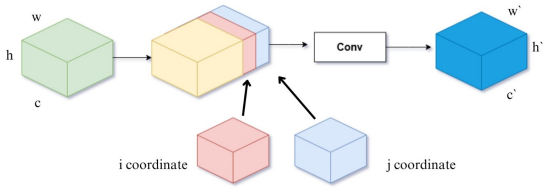


Fig. 5: Structure of Coordinate Convolution Module

We modify the base C2f module by replacing its standard convolution in the bottleneck with coordinate convolution. This adjustment enhances the model's spatial awareness, aligning better with the needs of Bangladeshi street food classification and segmentation.

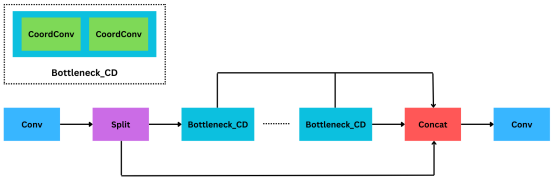


Fig. 6: Structure of C2f_CD Module

6.2 Convolutional Block Attention Module (CBAM) Module

The Convolutional Block Attention Module (CBAM) helps the model focus on essential information by filtering out irrelevant background noise. This is especially useful for Bangladeshi street food, where items often share similar appearances but differ in texture, shading, or partial occlusion. By highlighting unique patterns, CBAM enhances the model's ability to distinguish between visually similar classes. It operates in two stages: channel attention, which captures inter-channel relationships, followed by spatial attention, which pinpoints key areas in the feature maps—together improving feature refinement and recognition accuracy.

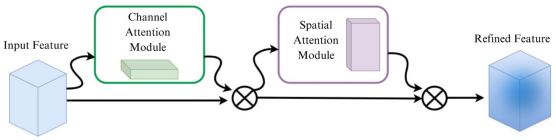


Fig. 7: Structure of CBAM module

6.2.1 Channel Attention

Channel attention uses global average and max pooling, followed by a shared MLP and sigmoid activation:

$$M_c = \sigma\left(\text{MLP}(\text{AvgPool}(X)) + \text{MLP}(\text{MaxPool}(X))\right) \tag{9}$$

The attention is applied as:

$$X' = M_c \odot X \tag{10}$$

The structure of channel attention module is represented in figure-8.

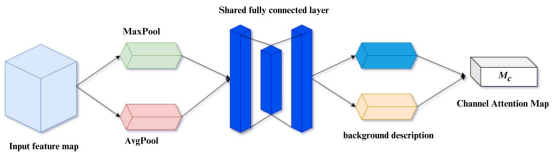


Fig. 8: Channel Attention module in CBAM

6.2.2 Spatial Attention

Spatial attention applies channel-wise pooling (average and max), followed by a 7×7 convolution and sigmoid activation:

$$M_s = \sigma\left(f^{7\times7}\left([\text{AvgPool}(X'), \text{MaxPool}(X')]\right)\right) \tag{11}$$

The final output is:

$$Y = M_s \odot X' \tag{12}$$

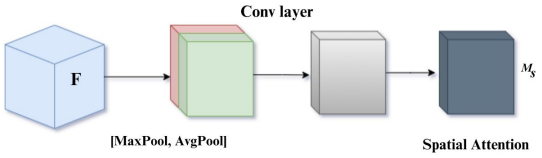


Fig. 9: Spatial Attention module in CBAM

7 Performance Metrics

Bangladeshi street food calorie estimation the performance metrics can be grouped into two categories. Metrics representing classification and segmentation performance and metrics representing regression or estimation performance.

7.1 Classification & Segmentation Metrics

7.1.1 Precision

Precision measures the accuracy of positive predictions by calculating the proportion of correctly identified positive cases relative to the total number of positive predictions made.

7.1.2 Recall

Recall, also known as sensitivity or true positive rate measures the ability to identify actual positive cases by calculating the proportion of correctly predicted positive instances relative to the total number of actual positive cases.

7.1.3 Mean Average Precision (mAP)

The Mean Average Precision measure is a standard method for evaluating the accuracy of predicted bounding boxes or masks against ground truth annotations. It allows us to analyze the models accuracy across various threshold values.

7.1.4 Giga Floating-Point Operations per Second (GFLOPs)

Giga Floating-Point Operations per Second (GFLOPs) is a metric used to measure the computational performance of a deep neural network model. It represents the number of floating-point operations performed by either the GPU or CPU per second.

7.2 Regression Metrics

7.2.1 Mean Absolute Error (MAE)

The Mean Absolute Error is the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset.

7.2.2 Root Mean Squared Error (RMSE)

Root Mean Squared Error represents the square root of the average squared difference between the original and predicted values in the dataset. It provides an estimate of the standard deviation of residuals.

7.2.3 R^2 Score

R-squared score is a statistical measure that indicates how much of the variation of a dependent variable is explained by an independent variable in a regression model.

8 Hyperparameters

The hyperparameters for the all YoloV8 models along with the experiment environment are discussed below.

Table 4: Hyperparameters of Model

Hyperparameter	Value
Epochs	125
Batch	32
Optimizer	AdamW
Initial Learning Rate	0.001

All experiments were conducted on Kaggle [31], which offers a cloud-based environment for training and evaluating deep learning models. We utilized an NVIDIA Tesla P100 GPU with 3,584 CUDA cores and 16GB HBM2 memory, running CUDA 12.1 and PyTorch 2.4. For all tests, we used the lightweight YoloV8n (Nano) chosen for its speed and efficiency, aligning with our real-time processing needs.

9 Results

Table-5 reveals a clear gap between Mask R-CNN and YoloV8, with the former trailing notably in mAP50-90. Adding CBAM or CoordConv alone to YoloV8n yields mixed results—CBAM slightly boosts box precision (+0.2%) but hurts mAP50-90, while CoordConv underperforms across all metrics. However, combining both leads to consistent gains: +0.9% precision, +0.8% recall, +1.0% mAP@50, and +0.6% mAP@50-95 over the base model. Mask metrics also improve, with +0.6% precision, +0.7% recall, +0.9% mAP@50, and a modest +0.1% mAP@50-95. Overall, the enhanced YoloV8n outperforms its base and other leading models in classifying and segmenting Bangladeshi street foods.

Table 5: Performance Metrics Comparison

Model	Bounding Box				Mask			
	P	R	mAP50	mAP50-95	P	R	mAP50	mAP50-95
Mask RCNN (ResNet50, C4)	90.1%	81.6%	90.1%	76.7%	89.4%	79.2%	89.3%	74.4%
Mask RCNN (ResNet50, DC5)	91.2%	82.4%	91.1%	78.2%	91.2%	81.9%	91.2%	77.7%
Mask RCNN (ResNet50, FPN)	90.5%	81.7%	90.5%	77.6%	90.8%	82.4%	90.8%	78.5%
YoloV8n	93.5%	89.2%	93.1%	86.2%	91.7%	87.4%	91.6%	81.1%
YoloV8n + CBAM	93.7%	88.9%	93.1%	85.8%	91.5%	86.8%	91.7%	80.6%
YoloV8n + CoodConv	93.4%	87.8%	92.5%	85.9%	91.7%	86.2%	91.3%	80.9%
YoloV8n + CBAM + CoodConv	**94.4%**	**90%**	**94.1%**	**86.8%**	**92.3%**	**88.1%**	**92.5%**	**81.2%**

9.1 Bounding Box Precision Recall Curves

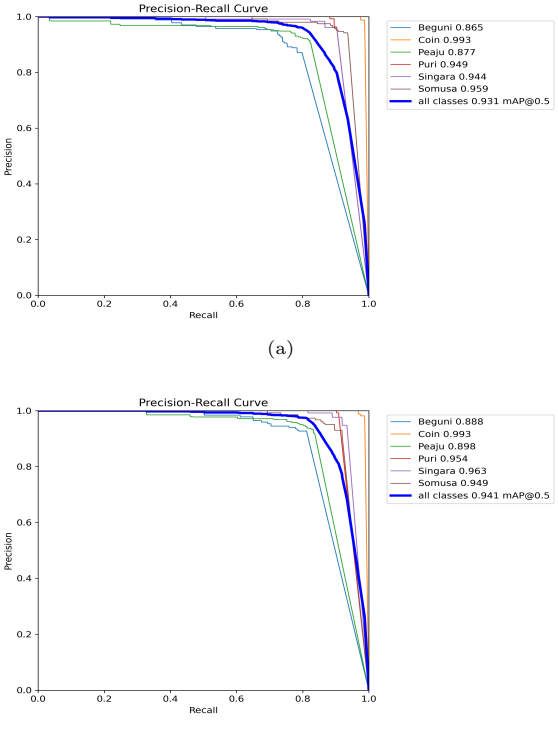


Fig. 10: Bounding Box Precision Recall Curve (a) Improved YoloV8n (b) Base YoloV8n

The Precision-Recall (PR) curves in the figure compare Improved YOLOv8n (a) with Base YOLOv8n (b) for food classification. The improved model achieves a slightly higher mAP@0.5 of 94.1% versus 93.1%, indicating an overall boost in precision and recall. Class-wise gains are also evident: Beguni improves from 86.5% to 88.8%, and Peaju from 87.7% to 89.8%. While some classes like Coin retain consistent high performance (99.3%), others show minor shifts—Singara improves to 96.3% from 94.4%, while Somusa drops slightly to 94.9% from 95.9%.

9.2 Mask Precision Recall Curves

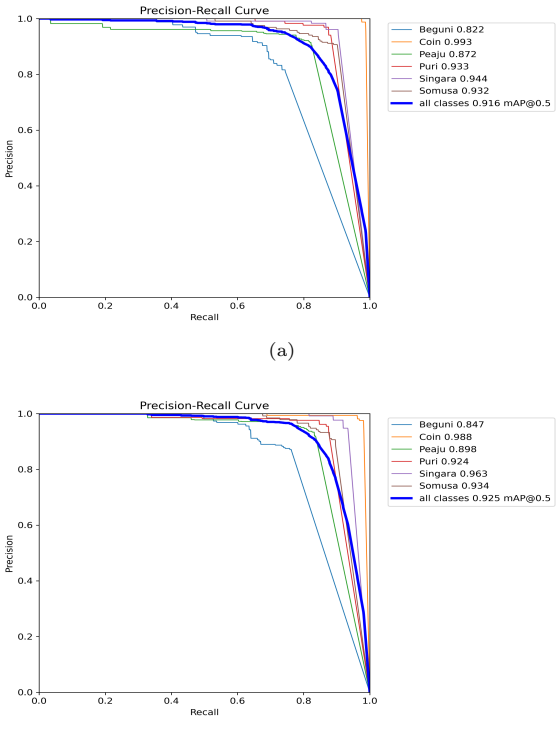
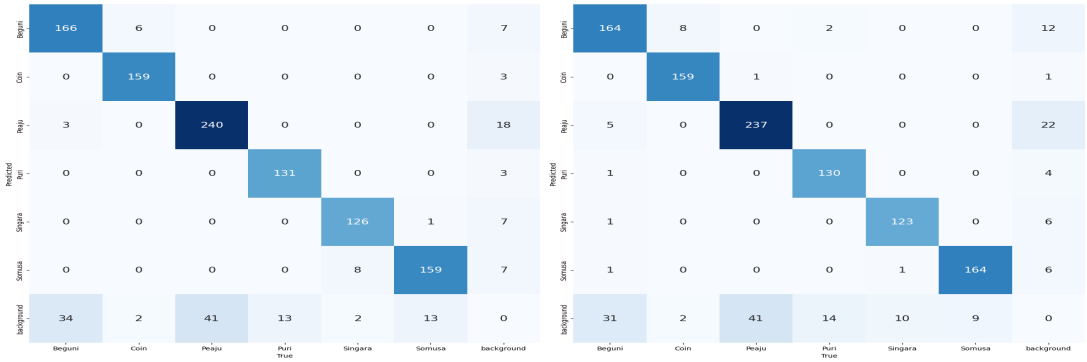


Fig. 11: Mask Precision Recall Curve (a) Improved YoloV8n (b) Base YoloV8n

The mask Precision-Recall (PR) curves illustrate segmentation performance for both the



(a) (b)

Fig. 12: Confusion Matrix Comparison(a) Improved YoloV8n (b) Base YoloV8n

improved YOLOv8n (a) and base YOLOv8n (b). The base model achieves a strong mAP@0.5 of 91.6%, with class-wise precision-recall scores like Beguni (82.2%), Coin (99.3%), Peaju (87.2%), Puri (93.3%), Singara (94.4%), and Somusa (93.2%) showing solid consistency. The improved YOLOv8n slightly outperforms it with a mAP of 92.5%, reflecting marginal but consistent gains across most categories: Beguni (84.7%), Coin (98.8%), Peaju (89.8%), Puri (92.4%), Singara (96.3%), and Somusa (93.4%). Overall, the improved model demonstrates more balanced, reliable segmentation across all classes.

9.3 Confusion Matrices Comparison

Comparison of the confusion matrices reveals that the improved YOLOv8n substantially reduces background hallucinations and false detections, indicating better discrimination between food items and irrelevant scene elements. It also improves intra-class separation, lowering confusion between visually similar items like Singara vs. Somusa or Beguni vs. Peaju. The improved model shows fewer misclassifications overall, evidenced by stronger diagonal dominance, confirming higher accuracy and robustness.

9.4 Computational Complexity Comparison

Compared to Mask R-CNN variants, YOLOv8n models show much lower computational complexity, reflected in fewer GFLOPs, parameters, faster inference, and smaller model size.

Table 6: Comparison of Models by Computational Complexity and Size

Model	GFLOPs	Params (M)	Time (ms)	Size (MB)
M-RCNN (R50, C4)	822	34	248.24	420.51
M-RCNN (R50, DC5)	344	166	149.83	2060.00
M-RCNN (R50, FPN)	447	44	58.16	526.23
YoloV8n (Base)	12.0	3.2	3.8	6.80
YoloV8n (Imp)	16.3	3.6	5.0	7.63

Mask R-CNN GFLOPs range from 344 to 822, with inference times up to 248 ms, making them unsuitable for real-time use. In contrast, base YOLOv8n has just 12 GFLOPs, 3.2M parameters, and a 3.8 ms inference time, making it far more efficient. Its model size is only 6.8 MB, compared to over 500 MB for Mask R-CNN. The improved YOLOv8n sees a slight rise in GFLOPs to 16.3 and parameters but still maintains fast inference around 5 ms, preserving real-time capability.

9.5 Rregression / Estimation Model Results

The performance evaluation of various machine learning models for calorie estimation shows that ensemble and non-linear models outperform traditional linear regression. Random Forest stands out with the lowest MAE of 6.94, RMSE of 11.03, and the highest R^2 of 96%, indicating accurate predictions with minimal errors.

Table 7: Performance Metrics of Machine Learning Models

Model	MAE	MSE	R^2 Score
Linear Regression	12.40	304.40	91.0%
KNN	7.62	140.50	95.0%
Decision Tree C	8.19	199.07	94.0%
Random Forest	6.94	121.80	96.0%
Gradient Boost	7.84	147.84	95.0%
AdaBoost	8.10	142.75	95.0%

This highlights the advantage of ensemble learning in capturing complex patterns and reducing overfitting. K-Nearest Neighbors, Gradient Boost, and AdaBoost also perform well, each achieving around 95% R^2, demonstrating their ability to model underlying data relationships. The Decision Tree performs moderately with 94% R^2 but has a higher RMSE (14.10), suggesting more susceptibility to overfitting. Linear Regression fares worst, with the highest MAE (12.40) and RMSE (17.44), showing its limitations in modeling non-linear dependencies typical in calorie estimation. Overall, ensemble methods like Random Forest and boosting approaches are best suited for accurate calorie prediction, balancing bias and variance while effectively handling complex food feature interactions.

10 Conclusion

This paper presents a novel approach for accurately estimating calories in Bangladeshi street food, addressing the lack of region-specific nutritional data. We created a diverse dataset of 3,885 images sourced both online and onsite, covering popular local street foods to ensure real-world relevance. Building on this, we developed an improved YoloV8 model that outperforms the base YoloV8 and other leading object detectors, delivering higher accuracy even in cluttered scenes typical of street food environments. To enhance calorie prediction, we employ a regression-based method that significantly lowers error margins, making the system not only accurate but also practical for real-world applications like mobile apps and automated dietary tracking.

11 Future Work

Looking ahead, a major challenge is scaling food items without relying on a reference object like a coin or plate. Future work will explore techniques such as depth estimation from images to infer true food sizes without needing a reference. While our dataset covers many popular Bangladeshi street foods, it lacks numerous traditional and regional dishes. Expanding the dataset to include more cuisine-specific items will improve the system's versatility and reach. Additionally, optimizing the model for computational efficiency is vital to enable smooth performance on low-end mobile devices, increasing accessibility and impact, especially in areas with limited access to high-end technology.