# Methodology

This work uses a thorough technique that includes data collection, dataset preparation, and the training of models in order to achieve reliable food recognition and the calculation of calorie values. The main goal is to look at how well the YOLO V8 model can identify foods and then estimate how many calories they contain. A thoughtfully designed series of actions has been developed to accomplish this purpose. The ECUSRFD dataset, which includes data on 19 different food classes, was first chosen as the foundation for the project. A selection of 10 food groups was taken from this dataset and used as the basis for the next steps. The inclusion of a broad variety of food varieties was then ensured by collecting photographs from other platforms, including Google and Roboflow, to create a diversified and vast dataset. The YOLO V8 model was then trained using this combined dataset. The parts that follow in this approach will lead you through each stage in detail, highlighting the methods used to collect the datasets, build up the model, and assess the model's performance.

## ECUSTFD Datasets

The ECUSTFD dataset, which consists of 2978 pictures altogether and spans 19 different food categories, serves as the basis for the analysis done in this work. Notably, this dataset is enhanced with annotations in XML format, giving each food item in the image’s exact spatial information. This important component of the dataset makes it easier to train the YOLO V8 model and accurately assess how well it recognizes and locates food items.

The current study adopted a planned strategy by extracting 1689 photos that cover 10 distinct food classes in order to achieve a concentrated and representative sample of food varieties. these food classes including apple, banana, egg, orange, kiwi, tomato, mango, lemon, lychee, and peach present a wide variety of food items typically found in everyday situations. The extraction procedure reduces the study's scope and supports the objective of testing the model's performance over a wide range of identifiable food categories.

After the relevant fraction was extracted, the dataset was carefully divided into training and testing sets. This partitioning maintained a 90% training to 10% testing ratio, ensuring there was enough data for both the model's learning and assessment stages. Table 1 provides comprehensive information on the sample distribution for the train and test sets within every food class.

Table 1: Class Distribution of ECUSTFD Dataset

| **Food Class** | **Total Samples** | **Train Samples** | **Test Samples** |
| --- | --- | --- | --- |
| Apple | 74 | 67 | 7 |
| Banana | 169 | 153 | 16 |
| Egg | 190 | 172 | 18 |
| Orange | 127 | 115 | 12 |
| Kiwi | 190 | 172 | 18 |
| Tomato | 285 | 258 | 27 |
| Mango | 232 | 210 | 22 |
| Lemon | 127 | 115 | 12 |
| Lychee | 84 | 76 | 8 |
| Peach | 211 | 191 | 20 |

## Self-Collected Dataset

A rigorously constructed self-compiled collection was used as the study's second dataset in addition to the ECUSTFD dataset. The Roboflow dataset and Google search results were only two of the many sources used to gather a total of 2708 photos that each fit into one of ten distinct cuisine categories. By including a wide range of culinary variants, this carefully chosen collection of photographs from various sources seeks to increase the model's potential to be applied in real-world situations.

The YOLO V8 model had to be trained to precisely detect and locate food items, which required a human annotation step as part of the dataset development process that went beyond simple image gathering. The bounding box coordinates were manually drawn on each of the 2708 photos, and they were then saved in a structured Yolo format. This thorough annotating procedure helps the model's object localization accuracy, which is essential for efficient food detection and precise calorie estimate.

The carefully prepared and annotated dataset was then strategically partitioned to create training and testing subsets. This partitioning assures a sizeable amount of data for both the model training and assessment stages by adhering to a ratio of 90% for training and 10% for testing. Table 2 provides a breakdown of the class distribution for each food type for both the train and test groups.

Table 2: Class Distribution of Self-Collected Dataset

| **Food Class** | **Total Samples** | **Train Samples** | **Test Samples** |
| --- | --- | --- | --- |
| Apple | 558 | 535 | 23 |
| Banana | 254 | 229 | 25 |
| Egg | 211 | 182 | 29 |
| Orange | 352 | 299 | 53 |
| Kiwi | 286 | 97 | 189 |
| Tomato | 379 | 41 | 338 |
| Mango | 333 | 287 | 46 |
| Lemon | 266 | 220 | 46 |
| Lychee | 225 | 206 | 19 |
| Peach | 126 | 110 | 16 |

The careful curation, labelling, and division of this self-collected dataset set provide the basis for thorough model training and evaluation, which is described in more detail in the following sections.

## Model Architecture

The YOLOv8 (You Only Look Once version 8) model design is the core of our food recognition method. YOLOv8 is known for being able to detect goods in real-time, which makes it a great choice for our job of recognizing food. The structure of the model is based on a single deep neural network that can predict both the coordinates of the bounding box and class probabilities of multiple items in a target image at the same time.

The YOLOv8 design is made up of several steps. The first step is to use a convolutional neural network (CNN) built on the CSPDarknet53 backbone to extract feature maps from the input picture. These feature maps, which we'll call F, go through a number of changes as they move through the detection layers. Each of these layers is in charge of predicting things at different sizes and shapes. The model forecasts the bounding box values (bx, by, bw, and bh) and class probabilities (P(C)) for a set of anchor boxes that have already been chosen.

The prediction at each layer can be expressed mathematically in the following way.

Where:

* (bx, by) shows where the center of the bounding box is expected to be.
* (bw, bh) are the same as the surrounding box's width and height.
* (t\_x, t\_y, t\_w, t\_h) are the bounding box change parameters that the network thinks will happen.
* (c\_x, c\_y) are the coordinates of the cell where the surrounding box is located.
* (p\_w, p\_h) stand for the width and height of the anchor box.
* (t\_c) is what the network thinks the objectness score will be.
* The symbol stands for the sigmoid activation function.

The training process of the YOLOv8 model involves the optimization of a composite loss function, which encompasses many components such as objectness loss, classification loss, and bounding box regression loss. The utilization of this particular loss function guarantees that the predictions made by the model undergo iterative refinement during the training procedure, hence facilitating the network's ability to precisely locate and identify food items.

Our food identification model is able to analyze data in real-time, locate food items precisely, and handle a wide range of food classes thanks to the strength of YOLOv8. With the help of this architecture, our model is able to accurately identify several food items in a picture and predict their appropriate classification labels and bounding box locations, which serves as the foundation for further food recognition and calorie estimate.

## Caloric Estimation

The calorie estimation method, which is a key part of this study, was carefully made to correctly measure the number of calories in well-known foods. This way of estimating was based on a series of steps meant to translate object recognition into a full understanding of nutritional value. To start figuring out how many calories a food has, the size of its bounding box was estimated. This area was used as a starting point for other measurements and calculations. The area estimate used the model's accurate predictions of the bounding box to get an exact idea of how big the food is in the image. After figuring out the area, the estimation method went on to figure out the amount of the known food item. This volumetric estimate was based on the derived area and helped figure out how much room the food took up in three dimensions. By relating the food's two-dimensional surface to its depth, the volume estimate showed where the food was in space. After figuring out how much food there was, the next step was to use a conversion factor based on how many calories the food had per 100 grammes. This conversion factor was based on nutritional information and made it possible to convert volume to calories. By multiplying the extracted amount by the calculated number of calories per 100 grammes, the calorie calculation method got a good idea of how much energy was in the food.

## Evaluation Measure

The efficacy of the food recognition model that has undergone training is assessed using a comprehensive set of assessment metrics, which collectively offer valuable insights into many facets of the model's performance. The metrics employed in this study include accuracy, precision, recall, F1-score, precision-recall (PR) curve analysis, and the Intersection over Union (IOU) score.

**Accuracy:** The metric of accuracy is a fundamental measure that measures the ratio of accurately predicted food items to the total number of predictions generated by the model. In mathematical terms, accuracy may be defined as the quotient obtained by dividing the number of correct predictions by the total number of forecasts made.

**Precision:** Precision is a metric that evaluates the model's capacity to limit the occurrence of false positives. It is calculated by dividing the number of properly recognized food items (true positives) by the total number of positive predictions, which includes both true positives and false positives.

**Recall:** Recall, which is sometimes referred to as sensitivity or true positive rate, evaluates the model's ability to correctly identify all pertinent events. The calculation involves determining the proportion of accurate positive predictions in relation to the overall count of positive events.

**F1-Score:** The F1-score is a metric that aims to find a compromise between accuracy and recall by calculating the harmonic mean. This measure effectively evaluates the model's ability to correctly forecast both true positives and false negatives.

**PR-Curve:** The Precision-Recall (PR) Curve illustrates the relationship between precision and recall at different categorization thresholds in a visual manner. The graphic depiction illustrates the model's capacity to accurately categorize positive events while decreasing the occurrence of false positives.

**IOU:** The Intersection over Union (IOU) Score, sometimes referred to as the Jaccard index, serves as a metric for measuring the degree of overlapping across the predicted bounding boxes and the ground truth bounding boxes. The metric is computed by dividing the amount of overlap between the predicted and ground truth bounding boxes by the combined area of both boxes.

Through a combined consideration of various evaluation factors, our assessment encompasses a full range of model performance. Accuracy is a metric that offers a broad perspective on the overall correctness of a system or model. On the other hand, precision and recall are measures that focus on the specific details of false positives and false negatives. The F1-score is a metric that combines precision and recall, providing a unified measure. The PR curve provides valuable information on classification thresholds, while the IOU score measures the spatial alignment between predicted and genuine bounding boxes. This comprehensive assessment guarantees a comprehensive evaluation of the model's capacity to identify food items and precisely estimate their calorie contents.

# Results and Discussion

The technique and assessment are completed by the Results and Discussion section, which reveals the findings of our thorough inquiry into food recognition and calorie estimate utilizing the YOLOv8 model. We offer a thorough review of the model's performance in this part, utilizing the already described evaluation measures. The model's accuracy, precision, recall, F1-score, PR curve, and IOU score are examined in detail, along with its advantages and disadvantages. We analyze the consequences of our findings and consider their larger ramifications for the area of food recognition and associated applications through thorough comparisons and perceptive interpretations.

## Environmental Setup

The Ultra Latics framework for dataset administration and the first training API were used to train the YOLOv8 model, which was then improved upon in the Google Colab setting. The model training procedure was made easier by using Python 3.11 and the necessary modules. The training process was accelerated by the publicly accessible GPU resources, resulting in effective convergence of the model. This coordinated arrangement made sure that the YOLOv8 model for identifying foods and estimating calories could be developed and tested without any problems. Numerous hardware, software and libraries were used to support the training process and the list of all utilized resources are available in Table 3.

Table 3: Specifications used resources in Training Environment

| **Hardware** | |
| --- | --- |
| GPU | Freely Available |
| **Software** | |
| Python | 3.11 |
| Cuda | 11.0 |
| cuDNN | 8.0 |
| **Libraries** | |
| **Library** | **Version** |
| TensorFlow | 2.5.0 |
| Keras | 2.4.3 |
| NumPy | 1.21.0 |
| Matplotlib | 3.4.3 |
| Pillow | 8.2.0 |
| Scikit-learn | 0.24.2 |
| OpenCV | 4.5.2 |

# Model Training and Evaluation with ECUSTFD Dataset

The Adam optimizer was used to speed up convergence while the YOLO V8 model was subsequently trained using a precisely chosen dataset of 1529 training samples. A learning rate of 0.0007 was used to coordinate the training process in order to strike a compromise between stability and quick convergence. This made it possible for the model to gradually improve its comprehension of the dataset's food item identification. A total of 100 training iterations were performed in an effort to get the best model performance. A batch of 72 images was shown throughout each cycle to allow the model to simultaneously integrate data from several examples. This batch-wise method sped up not just the training but also the computation of gradients and parameter updates, improving the effectiveness of the optimization procedure. The internal representations of the model were iteratively modified during training to better match the supplied training examples. The model was given the ability to precisely detect and identify food items contained in the input photos thanks to this adaptive learning, which was powered by the interaction of the optimizer and the carefully selected learning rate. By keeping an eye on the loss function's decrease, which shows a gradual alignment of predictions with real ground truth annotations, the model's convergence was measured. The performance of the model during training is shown in Figure 1(a-d) which represents the precision, recall, f1-scorfe and PR curve during the training of the model.

|  |  |
| --- | --- |
| (a) | (b) |
| (C) | (d) |

Figure 1: Evaluation curves of the model during training on ECUSTFD Dataset

Following the model training procedure, an evaluation was carried out using a set of 160 test samples that had been kept aside. The YOLO V8 model, which was trained on the samples of the ECUSTFD dataset, achieved a great test accuracy score of 94.3%, demonstrating its effectiveness in identifying food items in the test dataset. The model's accuracy, recall, and F1 score were also examined with a 0.50% confidence score threshold in mind. The F1-score was determined to be 0.9489%, the recall was 0.958%, and precision was 0.967%. The model's capacity to reduce false positives, increase the recovery of real positives, and establish a general balance between accuracy and recall are all highlighted by these measures taken together. The alignment between anticipated and actual bounding boxes was assessed further using the Intersection over Union (IOU) score. An IOU score of 0.45% was displayed by the model, demonstrating precise localization of food items inside the pictures. Table 4 provides a concise summary of these assessment scores, giving a quick look of how the model performed across several measures.

Table 4: Evaluation Scores of the Model Trained on the ECUSTFD Dataset

|  |  |
| --- | --- |
| **Evaluation Measure** | **Score** |
| Accuracy | 0.94 |
| Precision | 0.96 |
| Recall | 0.95 |
| F1 Score | 0.94 |

Additionally, Table 5 provides per-class assessment ratings that show the best results attained for classes including egg, orange, lemon, and litchi. The peach class, on the other hand, showed much lower results, suggesting how difficult it is to accurately recognize it. Figure 2 illustrates the distribution of samples that the model accurately predicted for each class, providing more precise information about the model's performance. It is noteworthy that the graphic shows that 16 of the 33 samples in the peach class were incorrectly categorized. This highlights the difficulties with class-specific identification.

Table 5: Per Class Accuracy of Model on ECUSTFD Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Food Class | Accuracy | Precision | Recall | F1-Score |
| Apple | 1.00 | 0.978 | 1.00 | 0.881 |
| Banana | 1.00 | 0.995 | 1.00 | 0.989 |
| Egg | 1.00 | 0.995 | 1.00 | 0.990 |
| Orange | 1.00 | 0.995 | 1.00 | 0.989 |
| Kiwi | 0.83 | 0.995 | 0.908 | 0.952 |
| Tomato | 0.93 | 0.903 | 0.926 | 0.916 |
| Mango | 0.82 | 0.943 | 0.802 | 0.890 |
| Lemon | 1.00 | 0.995 | 0.997 | 0.998 |
| Lychee | 1.00 | 0.995 | 1.00 | 0.982 |
| Peach | 0.85 | 0.887 | 0.950 | 0.866 |

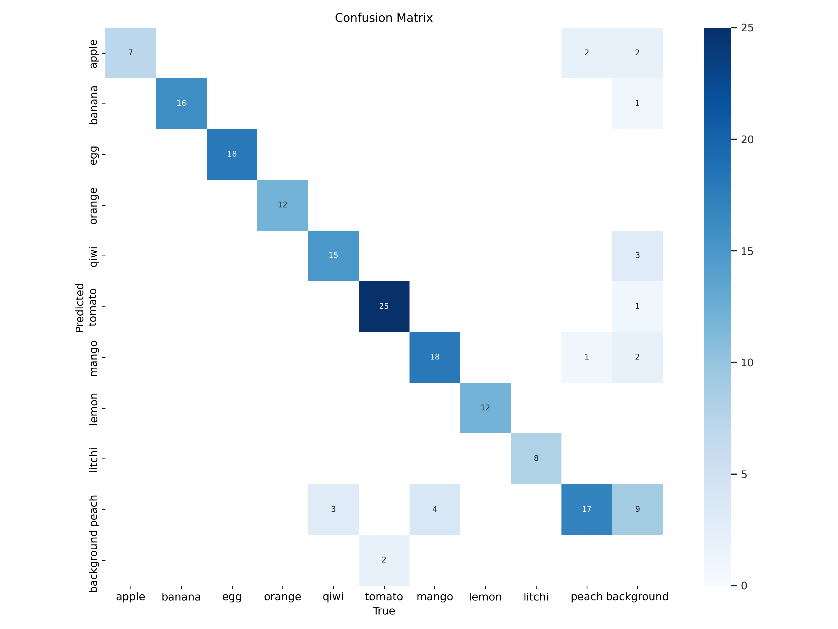


Figure 2: Confusion Matrix of the Model trained on ECUSTFD Dataset

In conclusion, Table 5 and Figure 2 shows that the best classes for food recognition in the ECUSTFD dataset are egg, orange, lemon, and litchi. Peach, on the other hand, poses a significant challenge as seen by its rate of misclassification. These observations help to elucidate the model's advantages and disadvantages in relation to various dietary groups.

# Model Training and Evaluation with Self-Collected Dataset

Real-world situations are very different, so an alternative training effort was done on a self-collected dataset that was designed to test the model's skill even more. This dataset, which included training examples from a wide range of sources like Roboflow and Google Search, showed how the flexibility of the model. Using the same YOLO V8 design, this self-collected dataset was carefully annotated to make the training process easier. Each of the n samples experienced a process of human annotation, which helped with accurate object localization and identification. The training process was the same as for the ECUSTFD dataset. An Adam optimizer with a learning rate of 0.007 was used. The model was trained iteratively over 100 epochs, with 71 images in each batch. The chosen batch size made it possible to make good use of the GPU's resources, which helped speed up convergence and parameter changes.

After the careful training procedure, a separate set of 784 test samples were used to evaluate the performance of YOLO V8 model that had been trained on the samples of self-collected dataset. A test accuracy score of 75.6% was given to the model's performance, which shows that it was able to recognize and classify food items in the test dataset. Also, the model's accuracy, recall, and F1-score were looked at, with a confidence score level of 0.50% as the main target. The model had a precision of 0.77%, a recall of 0.70%, and an F1-score of 0.66%. Together, these numbers show that the model was able to reduce false positives, increase true positives, and find a balance between precision and recall. The evaluation process included the Intersection over Union (IOU) score, which is a way to measure how well the real and projected bounding boxes match up. The model had an IOU threshold of 0.45%, which shows a good level of accuracy in locating objects. In Table 6, the results of these evaluation measures are summed up in a way that is easy to understand and gives a full picture of the model's success across a wide range of measurement parameters.

Table 6: Evaluation Scores of the Model Trained on the Self-Collected Dataset

|  |  |
| --- | --- |
| **Evaluation Measure** | **Score** |
| Accuracy | 0.75 |
| Precision | 0.77 |
| Recall | 0.70 |
| F1 Score | 0.66 |

Table 7 also shows the per-class rating scores, which show that the self-collected dataset has both good and difficult categories. In particular, the tomato, kiwi, and egg classes had lower performance scores, which shows how hard it is to recognize these things correctly. Figure 3 adds to the quantitative analysis by showing clearly how the expected samples for each class are spread out. It's interesting to see that a lot of tomato samples were wrongly labelled as apples, and some egg samples were given the wrong name of "background." This complexity points to class-specific problems and mistakes that can happen during recognition.

Table 7: Per Class Accuracy of Model on Self-Collected Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Food Class | Accuracy | Precision | Recall | F1-Score |
| Apple | 1.00 | 0.147 | 1.00 | 0.256 |
| Banana | 1.00 | 0.995 | 0.889 | 0.939 |
| Egg | 0.48 | 0.839 | 0.241 | 0.374 |
| Orange | 1.00 | 0.953 | 1.00 | 0.975 |
| Kiwi | 0.37 | 0.748 | 0.361 | 0.486 |
| Tomato | 0.10 | 0.604 | 0.063 | 0.114 |
| Mango | 0.61 | 0.464 | 0.522 | 0.491 |
| Lemon | 1.00 | 0.995 | 1.00 | 0.997 |
| Lychee | 1.00 | 0.995 | 1.00 | 0.997 |
| Peach | 1.00 | 0.995 | 1.00 | 0.997 |

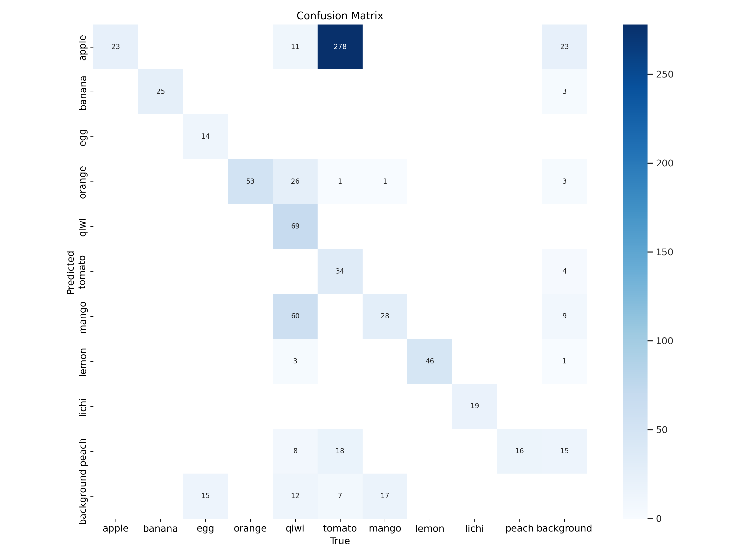


Figure 3: Confusion Matrix of the Model trained on Self-Collected Dataset

Together, the findings from Table 7 and Figure 3 show that the apple, banana, orange, lemon, litchi, and peach classes have the best recognition performance in the self-collected dataset. On the other hand, apple, kiwi, and egg are the most difficult groups to correctly identify. With these results, we can better understand the model's strengths and weaknesses across the different food groups in the self-collected dataset.

## Comparative Analysis and Discussion

When the performance of the two models that were trained on both datasets are compared, interesting things about the models' skills in different areas become clear. Accuracy, dataset variety, class-specific identification, and general efficiency are all parts of these factors.

* **Comparing accuracy:** the model trained on the ECUSTFD dataset had a score of 0.95%, which was much better than the model learned on the self-collected dataset, which had a score of 0.75. This difference shows that the first model is better at recognizing food than the second model. It also shows that the ECUSTFD dataset is useful for training a more accurate food recognition model.
* **Diversity in the dataset:** The ECUSTFD dataset was carefully gathered in a controlled setting, which made sure that the lighting, background, and picture quality were all the same. On the other hand, the self-collected collection was made up of pictures from many different sources, which reflected how different real-world situations are. Because of this, the model that was trained on the second dataset was exposed to a wider range of context changes, which made it better able to adapt to settings that aren't predictable.
* **Class-Specific Recognition:** The model that was trained on the self-collected dataset was able to recognize specific classes with a optimal score of 100% accuracy. But it wasn't able to correctly identify three classes: egg, kiwi, and tomato. The ECUSTFD dataset-trained model, on the other hand, had an average accuracy of 90% across all classes, with only a few classes scoring above 70%. This difference shows how good the model is at certain classes in the self-collected dataset and how well it does across the board in the ECUSTFD dataset.
* **Overall Model Performance:** The ECUSTFD dataset was used to train the model, and it had an accuracy score of about 0.90 for all classes, which shows that it did well across a wide range of food groups. On the other hand, the self-collected dataset-trained model was good at recognizing some classes but had trouble with others, giving it an average accuracy score of 0.75. While the first one was known all the time, the second one was only good at a few classes. This made for a tricky mix between their skills and weaknesses.

In conclusion, the comparison study shows how the two models have different strengths. The ECUSTFD dataset-based model stands out because it is very accurate and can recognize a wide range of things. The self-collected dataset-based model, on the other hand, is good at specialized recognition but has problems in some areas. This study shows how important it is to have a variety of datasets, controlled environments, and a complex relationship between accuracy and selectivity when building and analyzing food recognition models.