


Article

Utilizing RT-DETR Model for Fruit Calorie Estimation from Digital Images

Shaomei Tang * and Weiqi Yan * 

School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology,
Auckland 1010, New Zealand

* Correspondence: xry4889@aut.ac.nz (S.T.); weiqi.yan@aut.ac.nz (W.Y.)

Abstract: Estimating the calorie content of fruits is critical for weight management and maintaining overall health as well as aiding individuals in making informed dietary choices. Accurate knowledge of fruit calorie content assists in crafting personalized nutrition plans and preventing obesity and associated health issues. In this paper, we investigate the application of deep learning models for estimating the calorie content in fruits from digital images, aiming to provide a more efficient and accurate method for nutritional analysis. We create a dataset comprising images of various fruits and employ random data augmentation techniques during training to enhance model robustness. We utilize the RT-DETR model integrated into the ultralytics framework for implementation and conduct comparative experiments with YOLOv10 on the dataset. Our results show that the RT-DETR model achieved a precision rate of 99.01% and mAP50-95 of 94.45% in fruit detection from digital images, outperforming YOLOv10 in terms of F1- Confidence Curves, P-R curves, precision, and mAP. Conclusively, in this paper, we utilize a transformer architecture to detect fruits and estimate their calorie and nutritional content. The results of the experiments provide a technical reference for more accurately monitoring an individual's dietary intake by estimating the calorie content of fruits.

Keywords: RT-DETR; YOLOv10; deep learning; calorie; mAP



Citation: Tang, S.; Yan, W. Utilizing RT-DETR Model for Fruit Calorie Estimation from Digital Images. *Information* **2024**, *15*, 469. <https://doi.org/10.3390/info15080469>

Academic Editors: Nikolaos Mitianoudis and Ilias Theodorakopoulos

Received: 28 June 2024

Revised: 28 July 2024

Accepted: 5 August 2024

Published: 7 August 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Nowadays, obesity has emerged as a significant global health issue due to its association with an increased risk of diseases such as heart disease, diabetes, and hypertension [1]. An effective method to prevent obesity is through controlling the calorie intake in foods [2]. In daily diets, fruits and vegetables play a crucial role as primary sources of nutrition. However, a great number of individuals lack understanding regarding the calorie and nutritional content of various foods, necessitating a method to help them easily comprehend the calorie content of their food intake [3]. With the advancement of technology, various artificial intelligence systems have been investigated to help people understand the daily calorie intake of fruits and vegetables, aiding them in better diet control, such as the research of Begum et al. [4].

In this project, we propose a Transformer-based deep learning model to calculate the calories in fruits. Transformer architecture [5] was initially devised for tasks related to natural language processing (NLP) but has been so successful that deep learning models based on it have flourished and exhibited exceptional performance across various computer vision tasks, notably in object detection. The framework we used for the real-time detection of objects utilizing the transformer architecture is Real-Time Detection Transformer (RT-DETR) [6], which has achieved impressive accuracy in real-time object detection. The motivation of our project is to utilize the features of the RT-DETR model to create a system that is able to detect fruits in real time using a camera feed and estimate their calorie content. By automating these processes, we streamline workflows, improve efficiency, and provide users with valuable insights into their dietary habits.

This project aims to explore the capabilities of the RT-DETR model in fruit detection and calorie estimation, evaluate its performance compared to existing methods like YOLOv10, and showcase its potential for practical use in dietary monitoring and nutrition analysis. Hence, the research query addressed in this paper is how well deep learning technology performs in estimating the calorie content of fruits. Three hypotheses are introduced to guide our research:

Hypothesis 1: *The RT-DETR model will achieve higher accuracy and precision across various performance metrics in fruit detection compared to the YOLOv10 model.*

Hypothesis 2: *The RT-DETR model will produce lower loss values in the detection of fruits compared to the YOLOv10 model.*

Hypothesis 3: *The RT-DETR model will provide a better user experience and effectiveness in practical applications compared to the YOLOv10 model.*

2. Materials and Methods

In this section, we describe the methods we used to achieve relevant results. We will provide a literature review of relevant background studies, followed by a description of the methods used in this project.

2.1. Transformer

Transformer architecture [5] is specifically crafted for processing data of sequence, such as the words in a sentence. It processes incoming sequences and converts them into other sequences. It utilizes self-attention exclusively to calculate the input and output representations, eliminating the need for sequence-aligned RNNs (Recurrent Neural Networks) [7] or convolution. The architecture comprises an encoder and a decoder, illustrated in Figure 1.

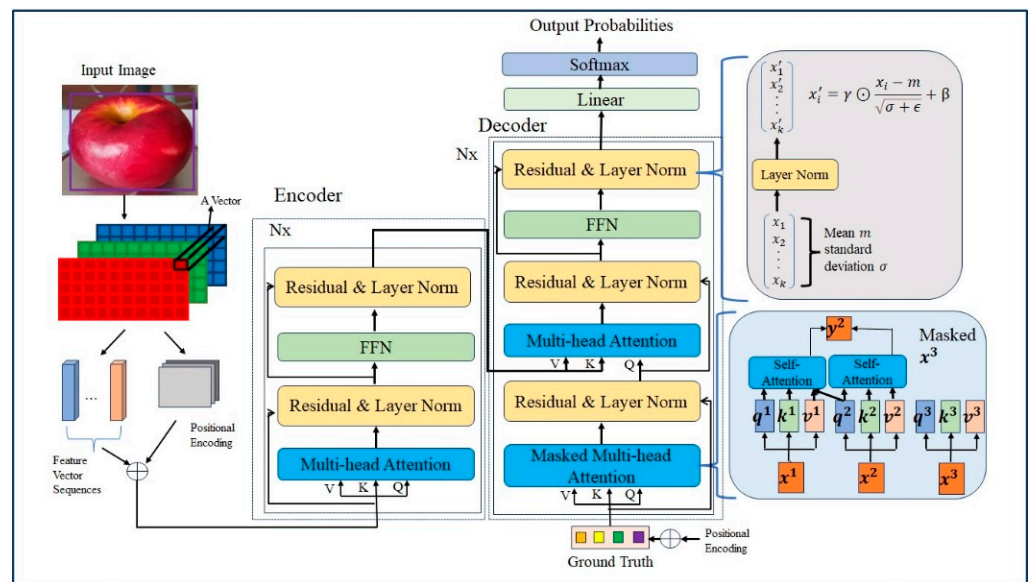


Figure 1. The transformer architecture.

In Figure 1, the encoder consists of N (equal to 6) identical layers; each contains two sub-layers: multi-head self-attention and a fully connected feed-forward network (FFN). These sub-layers include residual connections [8] and layer normalization [9]. The multi-head attention mechanism involves projecting queries, keys, and values through linear transformations and then concatenating the resulting attention results. Each sub-layer output is computed as the layer normalization of the sum of the sub-layer input and its output.

In the first sub-layer, multi-head self-attention is employed, where attention is calculated for a set of queries concurrently by using a scaled dot-product mechanism. The output is obtained by multiplying the softmax function of the scaled dot product of the query and key with the value matrix. This process is repeated for each head and then concatenated and linearly transformed.

The second sub-layer is a position-wise fully connected feedforward neural network (FFN) that provides linear transformations. It applies a nonlinear activation function after the linear transformation to introduce complexity.

The decoder also consists of N layers, which were split into sub-layers. The first sub-layer utilizes masked multi-headed attention to ensure that the predictions for each position depend only on previous positions. The second sub-layer is an encoder–decoder multi-head attention, where the input comes from the encoder’s output and the decoder’s previous layer output. This layer facilitates information exchange between the encoder and decoder.

2.2. DETR

Transformer initially was designed for natural language processing, which has applications in visual object detection tasks owing to its robust modeling capability and efficient parallel computing [5]. Visual object detection from digital images involves identifying objects within images or videos, determining their positions, and assigning corresponding categories.

Compared to convolutional neural networks (CNNs), transformer-based backbone networks offer advantages such as larger receptive fields, more adaptable weight settings, and better global modeling capabilities [10]. In 2020, Google introduced the Vision Transformer (ViT) (URL: https://github.com/google-research/vision_transformer, accessed on 4 August 2024), which proved successful in image classification, marking a significant milestone in applying transformer models to computer vision (CV) [10].

Another notable development is the DETR model (URL: <https://github.com/facebookresearch/detr>, accessed on 4 August 2024), which employs an end-to-end transformer architecture to transform object detection into a sequence-to-sequence problem [11]. DETR simplifies the detection process by eliminating the need for manually crafted components like Non-Maximum Suppression (NMS) or anchor generation, achieving good performance. However, DETR suffers from drawbacks such as slow training, high computational overhead, and poor performance in detecting small objects [11]. To address these limitations, variants of DETR models have been proposed, each targeting specific challenges: PnP-DETR introduces a “poll and pool” sampling module to adaptively sample features of different granularity, balancing computational overhead and performance [12]. Deformable DETR reduces computational overhead by altering the attention mechanism calculation method, leveraging deformable convolutions to improve small object detection performance [13]. Sparse DETR further reduces computational costs by selectively updating only a portion of encoder tokens, maintaining detection performance [14]. Conditional DETR decouples appearance and position features to speed up convergence by learning conditional space queries [15]. The Anchor DETR model shows a new object query design by using anchor points to guide optimization and accelerate convergence [12]. DAB-DETR builds on Anchor DETR by introducing 4D reference points to further accelerate convergence [16]. DN-DETR addresses slow convergence by stabilizing training with noisy ground truth and query inputs [17].

These methods aim to expedite DETR convergence and have demonstrated effectiveness in experiments conducted on the COCO dataset. Through innovative approaches targeting different aspects of the detection process, these variants contribute to advancing the performance and efficiency of object detection models.

2.3. Our Methods

2.3.1. Dataset

Data Collection

Due to the seasonal variations in data collection, the neural network in this project is trained by utilizing the fruit categories covering the fruits most likely to be encountered when using the calorie detection system: Royal Gala Apple, Rose Apple, Granny Smith Apple, Ambrosia Apple, JAZZ Apple, Orange, and Kiwifruit.

We created a dataset comprising 1866 images of various fruits for fruit detection. By using a camera, we captured videos of each fruit from multiple angles at equidistant distances and the images were obtained by extracting frames from the videos. The dataset consists of seven classes of local fruit products. To identify fruit categories and estimate their calorie content, each fruit was classified into weight categories of equal intervals, resulting in a total of 22 categories. For example, for Royal Gala Apples, weight categories were defined as follows: “Royal_Gala_Apple 1” for weights up to 140 g, “Royal_Gala_Apple 2” for weights between 140 g and 180 g, and “Royal_Gala_Apple 3” for weights exceeding 180 g.

Due to the equidistant capture method, the images of fruits of different weights have varying dimensions, allowing the deep learning model to estimate calorie content based on image dimensions. The calorie and nutrient composition data for the 7 classes of fruits were sourced from “The Concise Food Composition Tables” [18]. These datasets are utilized for energy estimation during the fruit detection process. Additionally, by starting with these 1866 fruit images as a foundation, we employed various data augmentation techniques to generate a dataset comprising 4478 images. This dataset was subsequently partitioned into training, validation, and testing sets in the proportions of 88%, 8%, and 4% (100% in total), respectively. Specifically, the dataset includes 3918 images for the training, 374 images for the validation, and 186 images for the testing.

Data Preprocessing

To ensure neural networks for tasks like image classification and object detection are trained effectively, it is crucial to resize images to match the input layer’s predetermined size. This is the reason why convolutional layers in neural networks analyze images pixel by pixel and the interactions with neighboring pixels to identify features. Given the use of the ultralytics framework in this project, we have standardized image dimensions to 640 by 640 pixels, compatible with ultralytics specifications.

Data augmentation plays a key role in diversifying training data, enhancing model generalization and resilience. This involves transforming inputs to enlarge the scale and variety of annotated training datasets [19]. In this paper, various data augmentation methods were employed: (1) horizontal flipping with a 50% probability; (2) no rotation, clockwise rotation, and anti-clockwise rotation in 90-degree increments; (3) random cropping of 0% to 20% of image size; (4) random rotation between -15 and $+15$ degrees; (5) horizontal shearing between -10° and $+10^\circ$, and vertical shearing between -10° and $+10^\circ$; (6) random brightness adjustment between -15% and $+15\%$; (7) random grayscale application to a subset of the training set with a 15% probability. If these data augmentation techniques are applied to the images, we can obtain results similar to Figure 2. It is important to apply the augmentation only to the training dataset, not the validation or testing datasets.

Furthermore, in the training process, we utilized some default hyperparameters of the framework, including an initial and final learning rate of 0.01. The framework offers a variety of optimizers, such as SGD (URL: <https://github.com/CU-UQ/SGD>, accessed on 4 August 2024), Adam, Adamax, AdamW, NAdam, RAdam, and RMSProp. For our experiments, we opted to allow the model to automatically select the most suitable optimizer. Moreover, in order to enhance the robustness of this proposed model, we utilized random data augmentation from the ultralytics framework during model training. Specifically, the data augmentation approaches include the random Affine transformation which involves translating, shearing, rotating, and scaling the image based on specified

parameter values. In this transformation, we set the parameters as follows: degrees: 30.0, translate: 0.1, shear: 10, and scale: 0.5. Random 4-image mosaic augmentation combines four random images into a single mosaic image. HSV augmentation stands for hue, saturation, and value, collectively representing a color space used for describing colors. Figure 3 demonstrates the combined enhancement effect when we set the HSV parameters to hsv_h : 0.015, hsv_s : 0.7, and hsv_v : 0.4. We used the Albumentations toolkit to incorporate the techniques Blur, MedianBlur, ToGray, and CLAHE. Figure 4 showcases the effects of employing these augmentation techniques.

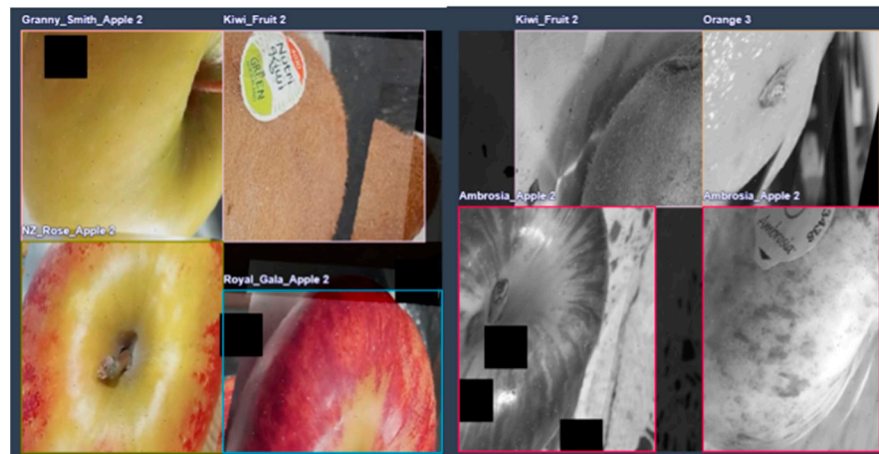


Figure 2. The images used in data augmentations.

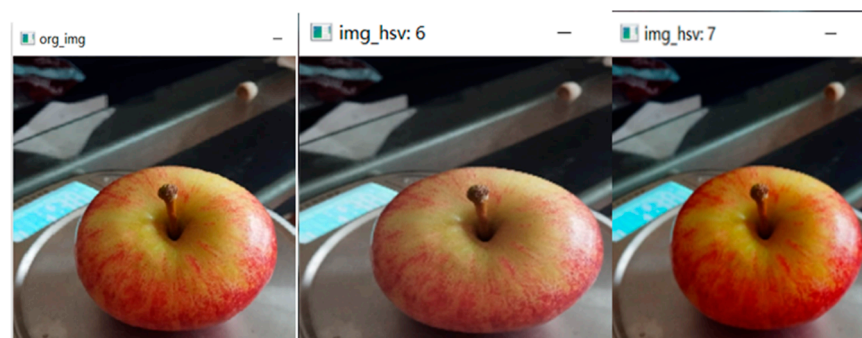


Figure 3. Applying HSV augmentation randomly.

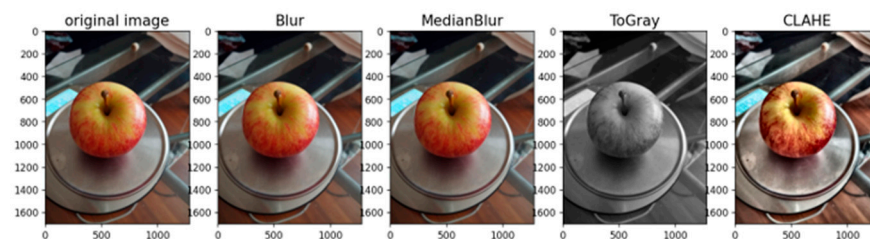


Figure 4. The effects of four augmentation techniques from the Albumentations Library.

2.3.2. RT-DETR Architecture

In this project, a DETR model named RT-DETR is employed for fruit detection and calorie estimation from digital images. RT-DETR [6] not only resolves the “two sets of thresholds” issue but also significantly improves practical utility, streamlining deployment processes. These advancements enable RT-DETR to meet real-time detection demands, leading to widespread adoption in practical applications. The architecture of RT-DETR is illustrated in Figure 5. In terms of structure, RT-DETR consists of three blocks: backbone network, neck network, and decoder.

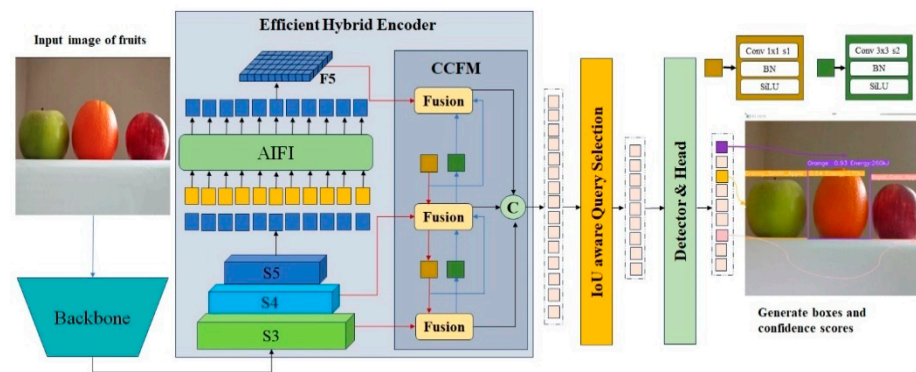


Figure 5. The architecture of RT-DETR.

Backbone. The backbone network of RT-DETR takes use of ResNet50, ResNet101 [8,10,20], and HGNet-v2 [21]. RT-DETR extracts outputs at three scales, S_3 , S_4 , and S_5 , from the backbone network. In this project, we trained the RT-DETR model on our dataset by using the HGNetv2 backbone.

Hybrid Encoder. For the neck network, RT-DETR employs a solitary layer of Transformer encoder, exclusively processing the S_5 features outputted from the backbone network, as shown in Figure 2, called AIFI (i.e., the Attention-based Intra-scale Feature Interaction) module. The mathematical operations of AIFI are represented as follows:

$$Q = K = V = Flatten(S_5) \quad (1)$$

$$F_5 = Reshape(Attn(Q, K, V)) \quad (2)$$

where *Attn* means the multi-head self-attention; *Reshape* is utilized to revert the feature's shape back to that of S_5 .

The two-dimensional S_5 features undergo flattening into a vector before being passed to the AIFI module. The computational process entails multi-head self-attention and FFN (Feed-Forward Network). Subsequently, the output is reshaped back into two dimensions, represented as S_5 , for further "cross-scale feature fusion." According to the RT-DETR research team, the decision of RT-DETR to only process the final S_5 feature through AIFI is based on two considerations: (1) Previous DETR models, such as Deformable DETR, concatenated features from multiple scales into one long sequence vector. While this approach facilitates ample interaction between features at different scales, it also leads to significant computational overhead and time consumption. RT-DETR is considered one of the primary reasons for the slow computation speed of the existing DETR models. (2) In RT-DETR, compared to shallower features like S_3 and S_4 , the S_5 features possess deeper, more advanced, and enhanced semantic information. These semantic features offer greater value and utility for Transformers to distinguish between different objects. In contrast, shallow features lack significant semantic information and are less effective.

The RT-DETR demonstrates that applying the encoder only to the S_5 features can significantly reduce computational complexity, improve computation speed, and maintain model performance.

IoU-aware query selection. IoU-aware query selection is introduced to guide the model during training. This approach enhances the classification by assigning higher scores to features with high IoU [22] scores and lower scores to those with low IoU scores. This improves the quality of initial object queries for the decoder, thereby enhancing detection performance.

To address the latency issues caused by NMS (Non-Maximum Suppression) [23] in current real-time detectors, RT-DETR introduces a real-time end-to-end detector. This detector comprises two key critical enhancements. Firstly, a hybrid encoder is designed to efficiently process multiscale features. Secondly, IoU-aware query selection enhances the

initialization of object queries. The combination of these improved components enhances the performance of our detector in real-time scenarios.

Decoder. RT-DETR supports flexible tuning of inference speed by employing varying numbers of decoder layers and eliminating the necessity for retraining, thus enabling the model to adapt to various real-time scenarios.

2.4. Operating Environment

The operating environment required in this paper includes Microsoft Windows 10 or later versions, Python 3.10 or later versions for the programming language, PyTorch 2.1.0 or later versions for the deep learning framework, and the ultralytics object detection framework and CUDA 11.7 or later versions for accelerated computing.

2.5. Evaluation Method

We evaluated the effectiveness of RT-DETR by conducting a comparative analysis with YOLOv10 on our dataset since YOLOv10 is the cutting-edge object detector constructed upon CNN architecture. The training performance of YOLOv9 and YOLOv8 [22] is also used for comparison, as they are high-performance versions of the YOLO models.

YOLO (You Only Look Once) has become mainstream in the field of real-time object detection due to its effective balance between detection speed and performance. The YOLO series [24] includes YOLOv1 through YOLOv10. YOLOv10 [25] is proposed after exploring various aspects of the YOLO series, aiming to further broaden the performance-efficiency frontier of YOLO through advancements in post-processing techniques and model design.

Typically, during training, YOLO uses a one-to-many label assignment approach, where a single ground truth object is matched with multiple positive samples. This method requires using NMS (Non-Maximum Suppression) during inference to choose the most promising positive prediction. This process reduces the inference speed and makes the performance dependent on NMS hyperparameters, preventing YOLO from achieving truly optimal end-to-end deployment. To address this, YOLOv10 introduces a training approach that eliminates the need for Non-Maximum Suppression, utilizing a dual-label assignment strategy with consistent matching criteria instead. Departing from the one-to-many assignment paradigm, YOLOv10 adopts a one-to-one matching scheme, which avoids the need for NMS during inference but introduces the drawback of weaker supervision, hampering both model accuracy and training convergence rate [26]. Thus, YOLOv10 adds another one-to-one head, retaining the same structure as the original one-to-many branch and adopting the same optimization objectives but utilizing one-to-one matching for label assignment. In the training procedure, dual heads are optimized together with the model, enabling the backbone and neck networks to take advantage of the extensive supervisory information offered by the one-to-many assignments. At inference time, the one-to-many head is discarded, and predictions are made using the one-to-one output head, allowing end-to-end deployment with no added computational burden, achieving high efficiency and competitive performance.

Additionally, YOLOs exhibit significant computational redundancy, with inefficient parameter utilization and suboptimal efficiency. YOLOv10 implements a comprehensive model design from the perspectives of both efficiency and accuracy.

Efficiency-driven model design. Firstly, YOLOv10 employs a lightweight architecture for the classification head, reducing the overhead without significantly affecting performance. Secondly, it decouples spatial downsampling and channel increase operations to achieve more efficient downsampling, reducing computational cost and parameter count while maximizing preservation and downsampling to achieve strong performance with less delay. Lastly, YOLOv10 uses a rank-guided block design strategy instead of applying the same module design for all stages. It introduces a compact inverted block (CIB) structure, using affordable depthwise convolutions for spatial integration and cost-effective pointwise convolutions for channel mixing. A module allocation strategy guided by ranks is employed to achieve optimal efficiency while preserving competitive capability, adapting

a compact module design to reduce complexity in stages showing redundancy without compromising performance.

Accuracy-driven model design. YOLOv10 uses large-kernel depthwise convolutions in the deep stages of the CIB [25] for small model scales to expand the receptive field and enhance model capability. Furthermore, it introduces an effective Partial Self-Attention (PSA) [25] module design, where a portion of features is fed into NPSA [25] blocks comprising multi-head self-attention (MHSA) [5] and feed-forward networks (FFN) [5] after the lowest-resolution Stage 4. The two segments are subsequently combined and integrated using a 1×1 convolution. This approach incorporates the global modeling ability of self-attention into YOLOs while keeping the computational cost low, effectively enhancing model capability and improving performance.

3. Results

In this section, we provide a detailed analysis of the results produced by the models we trained. All training results refer to the model being trained on our dataset for 100 epochs, with a batch size of 4. The performance metrics include precision, P-R curves, and loss values. Additionally, we conduct real-time experiments to evaluate the detection performance of the model.

3.1. Benchmark Tests

We utilize four metrics to evaluate the performance of the model: precision, recall, mAP50, and mAP50-95. Precision refers to the capability of this proposed model to correctly identify and classify only visual objects that are pertinent to the given task. Recall assesses the fraction of all true positive samples that the model can identify. mAP is short for mean average precision, indicating the mean precision across different classes. mAP50 denotes the value of mAP at a 50% threshold of IoU. In formal terms, the average precision (AP) for a specific class is derived from the region under the precision–recall curve [27].

$$AP = \int p(r)dr \quad (3)$$

The term *mAP* is utilized to calculate the mean values of AP across all classes.

$$mAP = \frac{1}{nc} \sum AP \quad (4)$$

where *nc* is the total count of classes.

The term mAP50-95 is a stricter evaluation metric as it computes the value of mAP across the range of 50–95% IoU thresholds (e.g., from 0.5 to 0.95, with increments of 0.05, i.e., 0.5, 0.55, 0.6 . . . , 0.95), and then takes the average.

Table 1 displays the best performance of these three models across the four metrics for 100 epochs. The RT-DETR model has higher precision, recall, and mAP50 compared to YOLOv8, YOLOv9, and YOLOv10, with values of 99.01%, 99.20%, and 99.17%, respectively. Furthermore, the RT-DETR model achieves higher mAP50-95 than YOLOv8 and YOLOv9; however, it is slightly lower than that of YOLOv10, with values of 94.45% and 95.35%, respectively. In addition, the training time for RT-DETR is shorter than that of YOLOv9 but is three times longer than YOLOv10 and five times longer than YOLOv8.

Table 1. Performance values of YOLOv8, YOLOv9, YOLOv10, and RT-DETR.

	Precision	Recall	mAP50	mAP50-95	Training Time
YOLOv8	94.57%	95.17%	97.87%	93.01%	54 min
YOLOv9	96.63%	91.32%	98.56%	94.64%	9 h 23 min
YOLOv10	96.27%	97.11%	99.07%	95.35%	3 h 11 min
RT-DETR	99.01%	99.20%	99.17%	94.45%	6 h 35 min

3.2. F1–Confidence Curves

The F1 curve represents the harmonic mean of precision and recall. It ranges from 0 to 1, where a value of 1 indicates optimal performance and 0 indicates poor performance. The F1 score curve shows how the F1 score changes at different thresholds. It is mathematically represented by Equations (5) and (6) [27]:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5}$$

$$F_1 = 2TP / (2TP + FN + FP) \tag{6}$$

where precision measures the accuracy of detections by indicating the proportion of predicted bounding boxes that correspond to ground truth objects. It reflects how many of the predicted objects are correct. Recall evaluates the capacity of the model to detect ground truth objects by indicating the proportion of actual objects that are accurately identified. Figures 6a and 6b depict the F1–confidence curves for the YOLOv10 and RT-DETR, respectively.

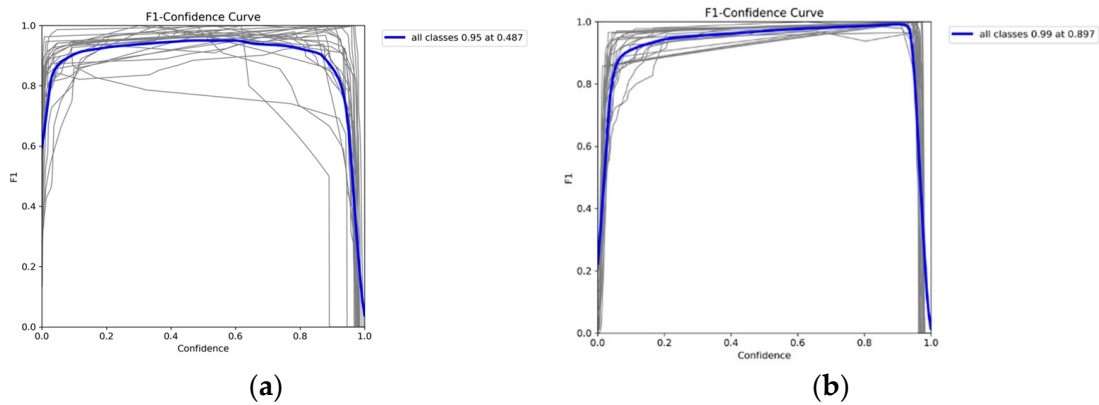


Figure 6. F1–confidence curves for YOLOv10 (a) and RT-DETR (b). The graylines are our F1–confidence curves with different parameters and datasets.

(1) The maximum F1 score for the YOLOv10 is 0.95, while RT-DETR achieves 0.99, indicating an improvement of 0.04.

(2) The region beneath the F1–confidence curve provides a summary of performance across all thresholds. A greater region indicates superior model performance. The results show that the region beneath the curve of the RT-DETR model is larger than that of the YOLOv10 model.

3.3. P–R Curves

The P–R curve showcases the trade-off between precision and recall. We aim at the curve to approach the point (1.0,1.0) by indicating maximum precision and recall, thus maximizing the area under the mAP curve as close to 1.0 as possible.

Figures 7a and 7b illustrate the P–R curves for YOLOv10 and RT-DETR, respectively. We see that the curve of the RT-DETR model is higher than that of YOLOv10. Furthermore, the curve of the RT-DETR model is closer to the upper right corner compared to YOLOv10. In addition, the RT-DETR model has a higher AUC than YOLOv10, indicating that the RT-DETR model exhibits better performance.

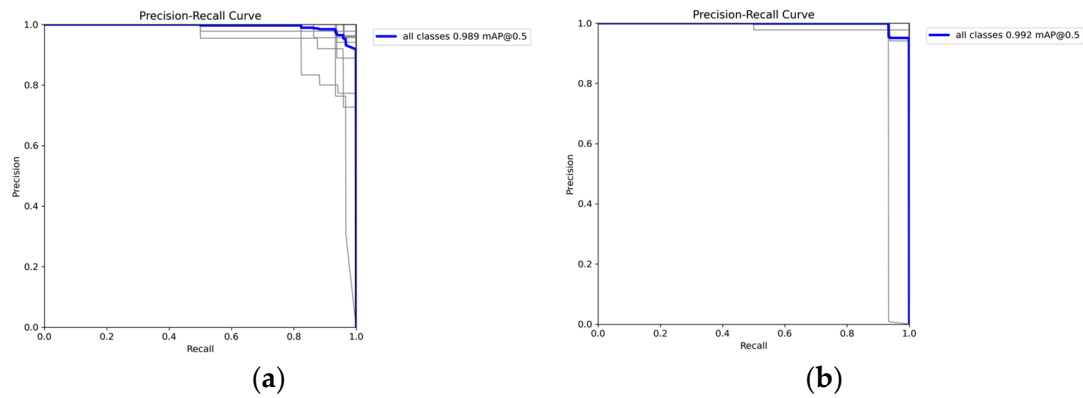


Figure 7. The P-R curves for YOLOv10 (a) and RT-DETR (b). The graylines indicate the different parameters and datasets for testing our proposed models.

3.4. Loss Values

We use three loss functions to measure the extent to which the model’s predictions deviate from the ground truth, aiming to extensively evaluate the performance of the proposed model.

- **GIoU loss (Localization Loss)/box_loss:** This loss function calculates the difference between predicted bounding boxes and ground truth bounding boxes.
- **Cls_loss (Classification loss):** The model uses classification loss to measure the accuracy of classification.
- **L1 loss (Feature Point Loss):** The model utilizes it to directly measure the absolute differences between the predicted and ground truth bounding box coordinates.
- **df1_loss (Distribution Focal Loss):** It aims to enhance the model’s accuracy, especially in complex object detection scenarios by focusing more on hard-to-detect objects. Essentially, it adjusts how the model weighs errors differently depending on their difficulty, helping the model to better estimate object categories. It is a loss metric utilized by YOLO models but not applied in RT-DETR.

Note: For the regression loss, we use GIoU loss (be presented as bos_loss in YOLO models) as the main loss metric for comparison, as GIoU loss takes correlation into account compared to L1 and L2.

Tables 2 and 3 show that, overall, the GIoU loss (Localization Loss) and Cls_loss (Classification Loss) of the RT-DETR model are smaller than those of YOLOv10, YOLOv9, and YOLOv8 during both the training and validation phases. Specifically, during the training phase, the GIoU loss and Cls_loss for the RT-DETR model are 0.04 and 0.1, respectively, while YOLOv10 has values of 0.34 and 0.31. In the validation phase, the GIoU loss and Cls_loss for the RT-DETR model are 0.09 and 0.21, compared to 0.65 and 0.37 for YOLOv10.

Table 2. Loss values for YOLOv10, YOLOv9, and YOLOv8.

	Train			Val		
	box_loss	cls_loss	df1_loss	box_loss	cls_loss	df1_loss
YOLOV8	0.2	0.28	0.89	0.36	0.26	0.98
YOLOV9	0.36	0.85	1.17	0.42	0.34	1.11
YOLOv10	0.34	0.31	1.75	0.65	0.37	1.79

Table 3. Loss values for RT-DETR.

	Train			Val		
	giou_loss	cls_loss	l1_loss	giou_loss	cls_loss	l1_loss
RT-DETR	0.04	0.1	0.05	0.09	0.21	0.18

3.5. Real-Time Detection

We used multiple fruits with varying weights for real-time prediction. Additionally, the juice content of fruits of equal weight to the detected fruits was also reflected in the experimental results.

Figure 8a–e shows the detection outcomes generated by using RT-DETR (left side) and YOLOv10 (right side) models in diverse environments. Overall, the RT-DETR model has a higher detection accuracy than the YOLOv10 model.

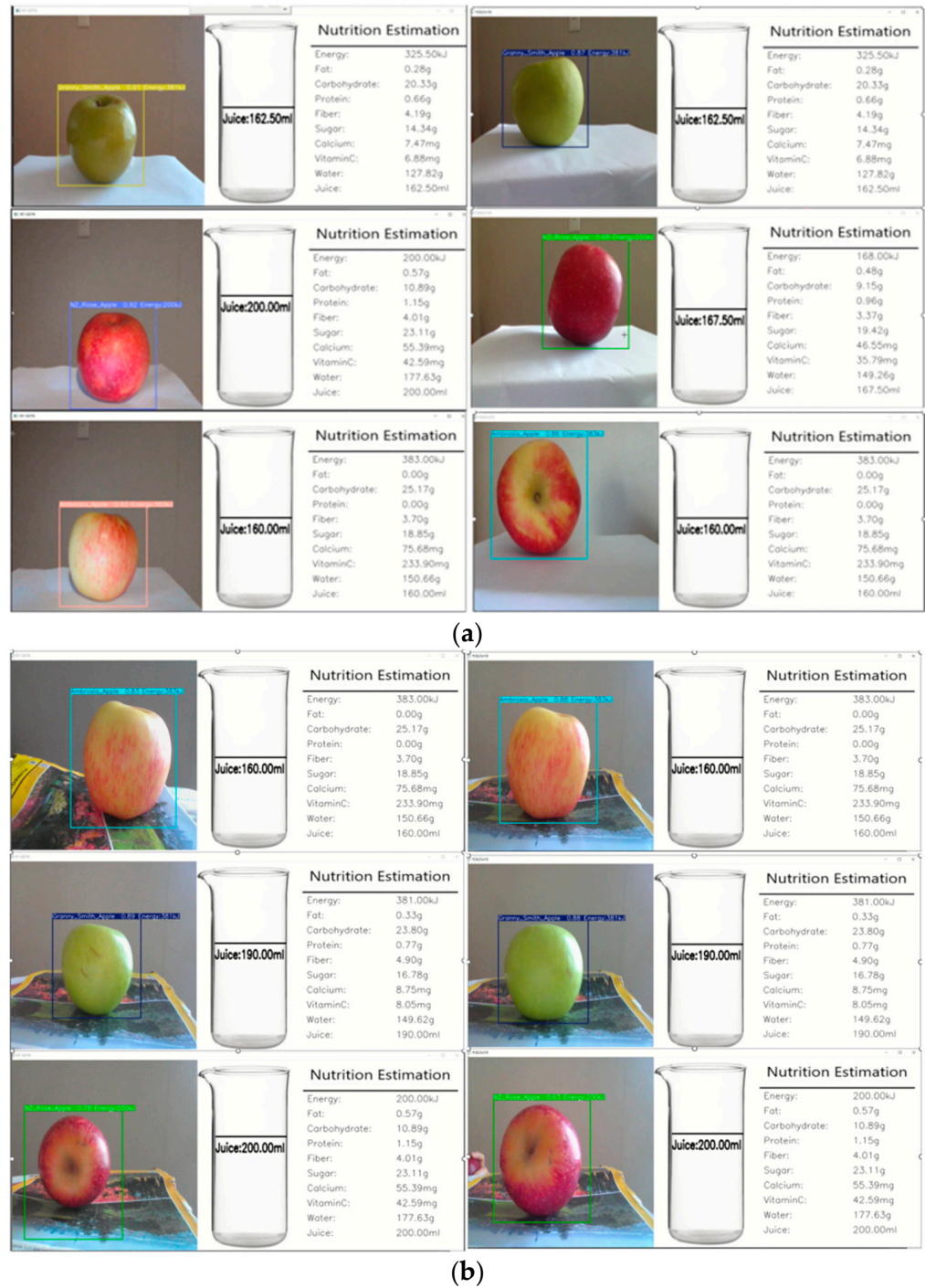
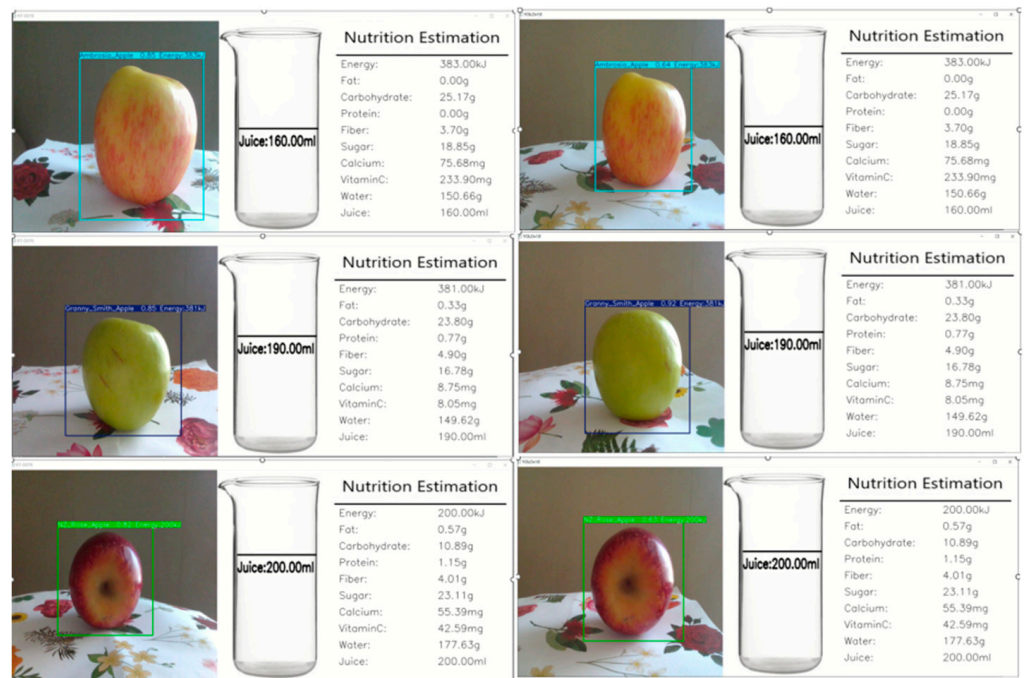
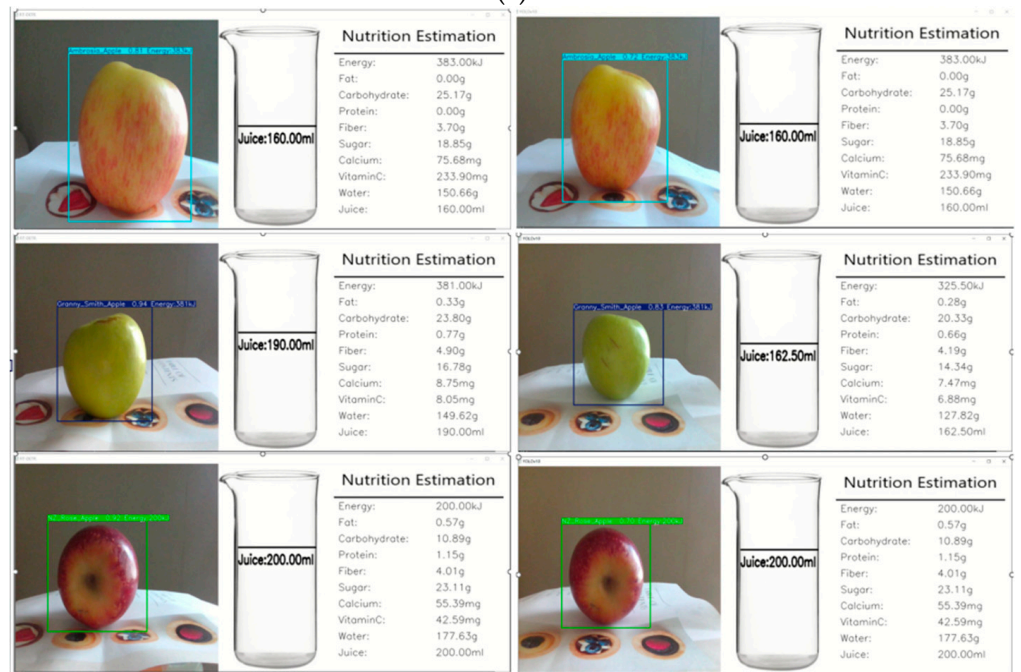


Figure 8. Cont.



(c)



(d)

Figure 8. Cont.

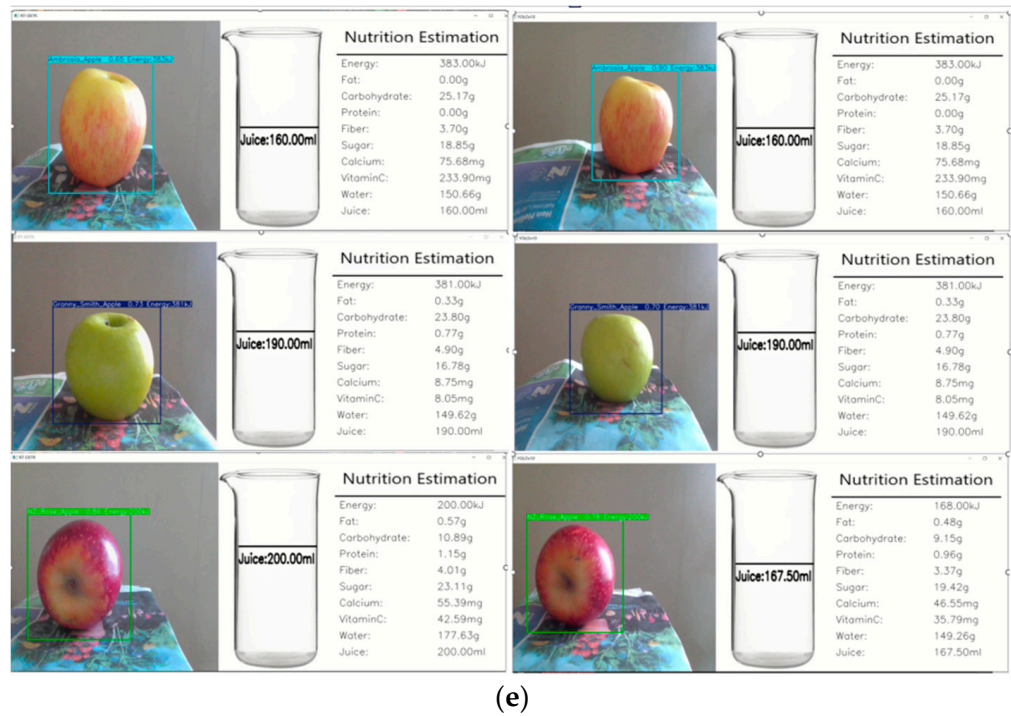


Figure 8. (a–e) Prediction of RT-DETR model (left) and YOLOv10 model (right) in various backgrounds.

Additionally, while it might be due to insufficient sample diversity, the two models have calorie estimation errors. For instance, Figure 9a shows that an NZ Rose apple originally containing 264 kJ of energy was detected by RT-DETR as having 200 kJ (left) and by YOLOv10 models as having 136 kJ (right). Figure 9b shows that both the RT-DETR model and the YOLOv10 model detected an NZ Rose apple, which originally contained 136 kJ of energy, as having 200 kJ.

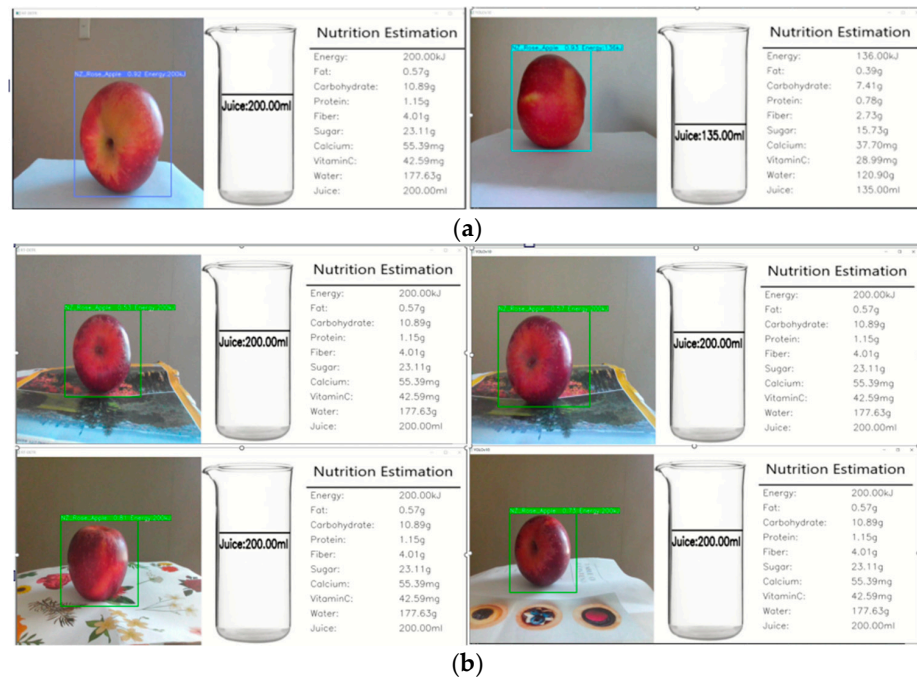


Figure 9. (a,b) Calorie estimation error for the RT-DETR model and the YOLOv10 model in different environments.

Furthermore, the two models occasionally exhibit detection errors. For instance, Figure 10a,b shows that the RT-DETR and YOLOv10 models both misidentified an Ambrosia apple or a Rose apple as a Gala apple.

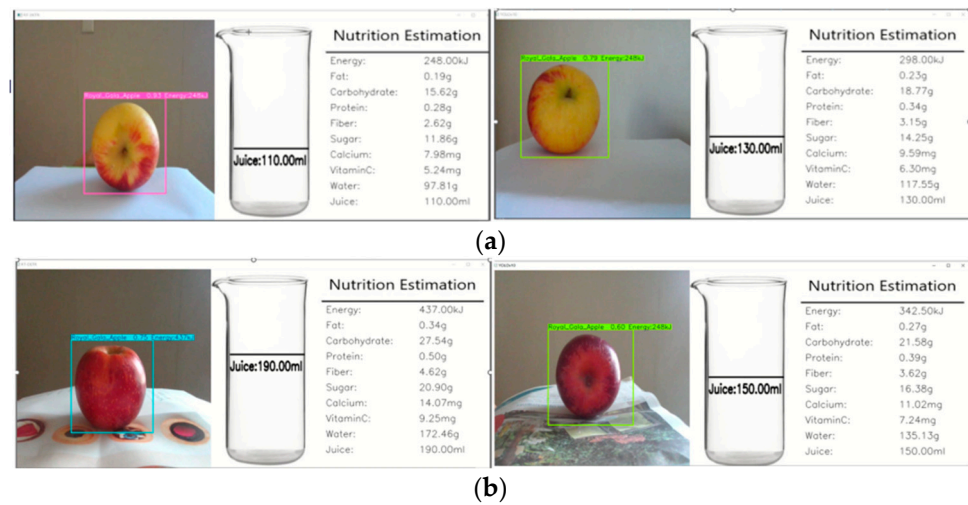


Figure 10. (a,b) Both RT-DETR and YOLOv10 models incorrectly detected an Ambrosia apple or a Rose apple as a Gala apple. (a) The case in misidentification for an Ambrosia apple (b) The case in misidentification for an Rose apple.

4. Discussion

This project aims to investigate the application of deep learning models for estimating the calorie content in fruits. To achieve this goal, we generated a dataset from videos captured by ourselves. We trained the TR-DETR model on our dataset and performed detection in a real-time environment. We will discuss the results, explore the limitations of the project, and examine potential future directions based on the training and real-time detection outcomes and the three hypotheses.

4.1. Will the RT-DETR Model Overall Outperform YOLOv10 across Various Performance Metrics?

Table 1 presents the performance metrics for four object detection models: YOLOv8, YOLOv9, YOLOv10, and RT-DETR. It highlights that RT-DETR generally outperforms the YOLO models in precision (99.01%), recall (99.20%), and mAP50 (99.17%), indicating better detection accuracy. However, YOLOv10 has a slightly higher mAP50-95 score than the RT-DETR model, with values of 95.35% and 94.45%, respectively, showing marginally better performance across different IoU thresholds.

Figure 6 shows that the RT-DETR model has a higher peak F1 score than YOLOv10, indicating superior optimal performance. Additionally, the larger area under the F1-confidence curve for RT-DETR compared to YOLOv10 suggests that RT-DETR consistently performs better across various confidence levels.

As shown in Figure 7, the P-R curve of the RT-DETR model is higher and closer to the upper right corner (1.0, 1.0) compared to YOLOv10, indicating superior precision and recall across different levels. Additionally, RT-DETR has a higher area under the curve (AUC) than YOLOv10, demonstrating overall better performance.

Additionally, for the real-time detection performance, Figure 8a–e shows that, overall, the RT-DETR model achieves higher precision than the YOLOv10 model across various backgrounds.

Based on our results, the first hypothesis is supported that the RT-DETR model overall outperforms YOLOv10 across various performance metrics.

4.2. Will the RT-DETR Model Produce Lower Loss Values in the Detection of Fruits Compared to the YOLOv10 Model?

In Tables 2 and 3, we see that the RT-DETR model consistently demonstrates lower GIoU loss (localization loss) and Cls_loss (classification loss) compared to YOLOv10, YOLOv9, and YOLOv8. This indicates that RT-DETR is more effective in both accurately localizing objects and classifying them.

Furthermore, during the training phase, RT-DETR's GIoU loss and Cls_loss are notably lower (0.04 and 0.1) than those of YOLOv10 (0.34 and 0.31). This suggests that RT-DETR achieves better performance and converges faster in terms of localization and classification than YOLOv10 during training.

Lastly, in the validation phase, RT-DETR continues to show lower GIoU loss and Cls_loss (0.09 and 0.21) compared to YOLOv10 (0.65 and 0.37). This indicates that RT-DETR maintains its superior performance and generalizes better to unseen data.

According to the above analysis, the second hypothesis is confirmed that the RT-DETR model exhibits lower loss values for fruit detection when compared to the YOLOv10 model.

4.3. Will the RT-DETR Model Provide a Better User Experience and Effectiveness in Practical Applications Compared to the YOLOv10 Model

Table 1 shows that the training time for RT-DETR is shorter than that of YOLOv9 but is three times longer than YOLOv10 and five times longer than YOLOv8. Additionally, in real-time detection, the RT-DETR model is also slower than the YOLOv10 model. Based on these, the third hypothesis is not supported that the RT-DETR model provides a better user experience and effectiveness in practical applications compared to the YOLOv10 model.

4.4. Detection Errors

Figures 9 and 10 show that both the TR-DETR and YOLOv10 models exhibit calorie estimation errors and occasional detection errors. These issues are likely attributable to insufficient sample diversity in the training data, which may affect the models' performance and accuracy.

4.5. Limitations and Future Work

This research project has several limitations. The uniformity of our dataset, stemming from fruits typically being displayed in supermarkets based on similar sizes, leads to a lack of diversity, potentially hampering comprehensive training. Moreover, the current focus solely on estimating individual fruit calorie content limits the extension to estimating total calorie content across multiple fruit types. Although the RT-DETR-L model outperforms YOLOv10, YOLOv9, and YOLOv8 models in real-time detection, its longer training time and less smooth performance during real-time detection call for further enhancements. Additionally, false detections during real-time detection underscore the necessity for precision improvement.

Our future work can focus on collecting more diverse samples to improve detection precision. Additionally, exploring the incorporation of fruit weight as a parameter in model training to estimate calorie content based on weight could be beneficial. Furthermore, the detection and estimation of total calories from multiple fruits simultaneously would broaden the scope of the project. Lastly, there is still room for improvement in the detection precision of our proposed model. Currently, our efforts are directed towards integrating CNN and Transformer architectures to achieve optimal results. Looking ahead, it is promising that Transformers may entirely replace CNNs in the realm of computer vision as further fine-tuning progresses.

5. Conclusions

This project aims to assess deep learning models for estimating fruit calorie content, focusing on the transformer-based RT-DETR model compared to the CNN-based YOLOv10 model. The RT-DETR model generally outperforms the YOLOv10 model in accuracy and

loss metrics, showing superior performance in detection precision and classification. Its higher F1 scores and larger areas under the precision–recall curves highlight its consistent superiority.

However, the RT-DETR model faces practical challenges, including longer training times and slower real-time detection compared to the YOLOv10 model, impacting its usability. Both RT-DETR and YOLO models also exhibit calorie estimation and detection errors, likely due to limited sample diversity in the training data, which affects their generalization ability.

Future research should focus on increasing dataset diversity to improve detection precision and reduce errors. Incorporating additional parameters, like fruit weight, could refine calorie estimation further. Additionally, exploring new model architectures, such as integrating CNN and Transformer techniques, may enhance both accuracy and efficiency.

In summary, while the RT-DETR model demonstrates strong performance, further development is needed to enhance its practical application and real-time performance. Addressing these limitations and exploring advanced methodologies will be key to improving the effectiveness of deep learning models in fruit calorie estimation.

Author Contributions: Conceptualization, S.T. and W.Y.; methodology, S.T.; software, S.T.; validation, S.T.; formal analysis, S.T.; investigation, S.T.; resources, S.T. and W.Y.; data collection, S.T.; writing—original draft preparation, S.T.; writing—review and editing, S.T. and W.Y.; visualization, S.T.; supervision, W.Y.; project administration, W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mansoor, S.; Jain, P.; Hassan, N.; Farooq, U.; Mirza, M.A.; Pandith, A.A.; Iqbal, Z. Role of Genetic and Dietary Implications in the Pathogenesis of Global Obesity. *Food Rev. Int.* **2022**, *38* (Suppl. S1), 434–455. [\[CrossRef\]](#)
2. Rolls, E.T. Understanding the Mechanisms of Food Intake and Obesity. *Obes. Rev.* **2007**, *8* (Suppl. S1), 67–72. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Veni, S.; Krishna Sameera, A.; Samuktha, V.; Anand, R. A Robust Approach Using Fuzzy Logic for the Calories Evaluation of Fruits. In Proceedings of the IEEE International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 26–27 August 2021; pp. 1–6.
4. Begum, N.; Goyal, A.; Sharma, S. Artificial Intelligence-Based Food Calories Estimation Methods in Diet Assessment Research. In *Artificial Intelligence Applications in Agriculture and Food Quality Improvement*; IGI Global: Hershey, PA, USA, 2022; p. 15.
5. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 2017; pp. 6000–6010.
6. Li, E.; Wang, Q.; Zhang, J.; Zhang, W.; Mo, H.; Wu, Y. Fish Detection under Occlusion Using Modified You Only Look Once v8 Integrating Real-Time Detection Transformer Features. *Appl. Sci.* **2023**, *13*, 12645. [\[CrossRef\]](#)
7. Manaswi, N.K. *RNN and LSTM. Deep Learning with Applications Using Python*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 115–126.
8. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 27–30.
9. Yu, R.; Wang, Z.; Wang, Y.; Li, K.; Liu, C.; Duan, H.; Ji, X.; Chen, J. LaPE: Layer-adaptive Position Embedding for Vision Transformers with Independent Layer Normalization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023.
10. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image Is Worth 16×16 Words: Transformers for Image Recognition at Scale. In Proceedings of the International Conference on Learning Representations, Virtual Only Conference, 3–7 May 2021.
11. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In Proceedings of the ECCV 2020, Lecture Notes in Computer Science, Glasgow, UK, 23–28 August 2020.

12. Wang, Y.; Zhang, X.; Yang, T.; Sun, J. Anchor DETR: Query Design for Transformer-Based Object Detection. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-22), Virtual Only Conference, 22 February–1 March 2022.
13. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In Proceedings of the International Conference on Learning Representations, Virtual Only Conference, 3–7 May 2021.
14. Roh, B.; Shin, J.; Shin, W.; Kim, S. Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity. In Proceedings of the International Conference on Learning Representations, Virtual Only Conference, 25–29 April 2022.
15. Meng, D.; Chen, X.; Fan, Z.; Zeng, G.; Li, H.; Yuan, Y.; Sun, L.; Wang, J. Conditional DETR for Fast Training Convergence. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 3631–3640.
16. Liu, S.; Li, F.; Zhang, H.; Yang, X.; Qi, X.; Su, H.; Zhu, J.; Zhang, L. DAB-DETR: Dynamic Anchor Boxes Are Better Queries for DETR. In Proceedings of the International Conference on Learning Representations, Virtual Only Conference, 25–29 April 2022.
17. Li, F.; Zhang, H.; Liu, S.; Guo, J.; Ni, L.M.; Zhang, L. DN-DETR: Accelerate DETR Training by Introducing Query DeNoising. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *46*, 2239–2251. [[CrossRef](#)] [[PubMed](#)]
18. Lister, C. Easily Accessible Food Composition Data: Accessing Strategic Food Information for Product Development and Marketing. *Food N. Z.* **2018**, *18*, 17–19.
19. Shorten, C.; Khoshgoftaar, T.M. A Survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [[CrossRef](#)]
20. He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J.; Li, M. Bag of Tricks for Image Classification with Convolutional Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 558–567.
21. Yao, T.; Li, Y.; Pan, Y.; Mei, T. HGNet: Learning Hierarchical Geometry from Points, Edges, and Surfaces. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 21846–21855.
22. Yan, W.Q. *Computational Methods for Deep Learning: Theory, Algorithms, and Implementations*; Springer: Berlin/Heidelberg, Germany, 2023.
23. Jiang, S.; Xu, T.; Li, J.; Huang, B.; Guo, J.; Bian, Z. IdentifyNet for Non-Maximum Suppression. *IEEE Access* **2019**, *7*, 148245–148253. [[CrossRef](#)]
24. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
25. Wu, W.; Liu, A.; Hu, J.; Mo, Y.; Xiang, S.; Duan, P.; Liang, Q. EUAVDet: An Efficient and Lightweight Object Detector for UAV Aerial Images with an Edge-Based Computing Platform. *Drones* **2024**, *8*, 261. [[CrossRef](#)]
26. Zong, Z.; Song, G.; Liu, Y. Detrs with Collaborative Hybrid Assignments Training. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 1–6 October 2023; pp. 6748–6758.
27. Padilla, R.; Passos, W.L.; Dias, T.L.B.; Netto, S.L.; da Silva, E.A.B. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics* **2021**, *10*, 279. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.